

Le package `lymath` : des formules plus sémantiques

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

Version 1.1.0-beta développée et testée sur Mac OS X.

Christophe BAL

2020-06-27

Table des matières

1	Introduction	6
2	Comment lire cette documentation ?	6
3	A propos des macros	6
3.1	Règles de nommage	6
3.1.1	Les macros de même « type »	6
3.1.2	Les formes « négatives » des macros	7
3.1.3	Les macros en mode <code>displaystyle</code>	7
3.1.4	Les macros standards redéfinies	7
3.2	Les arguments : deux conventions à connaître	7
3.2.1	Avec un nombre fixé d'arguments	7
3.2.2	Avec un nombre variable d'arguments	7
3.2.3	Important	7
4	Couleurs	7
5	Quelques modifications générales	8
5.1	Deux séparateurs d'arguments par défaut	8
5.2	Espace et point-virgule avec l'option <code>french</code> de <code>babel</code>	8
5.3	Espace et fractions	8
5.4	Espace et racines n-ièmes d'un réel	8
5.5	Espace après la négation logique	8
5.6	Sommes et produits en mode ligne	8
6	Logique et fondements	9
6.1	Différents types de comparaisons « standard »	9
6.1.1	Définir quelque chose	9
6.1.2	Indiquer une identité	9
6.1.3	Une égalité à vérifier ou non, une hypothèse, une condition	9

6.1.4	Une égalité indiquant le choix d'une valeur ou l'application d'une relation . . .	9
6.1.5	Une égalité indiquant l'équation d'une courbe	10
6.1.6	Différents types d'inéquations	10
6.1.7	Des formes négatives aussi pour les inéquations	10
6.1.8	Une table récapitulative	10
6.1.9	Textes utilisés	10
6.2	Équivalences et implications	11
6.2.1	Des symboles supplémentaires	11
6.2.2	Une table récapitulative	11
6.2.3	Équivalences et implications verticales	11
6.3	Tables des décorations possibles des opérateurs	12
6.4	Des versions alternatives du quantificateur \exists	12
6.5	Détailler un raisonnement simple	13
6.5.1	Version pour le lycée et après	13
6.5.2	Version pour les collégiens	17
6.5.3	De courts commentaires	19
6.5.4	Un mini hack très utile pour des « <i>étapes alignées</i> »	21
6.5.5	Un conseil de mise en forme	21
6.6	Détailler un « vrai » raisonnement	22
6.6.1	Un tableau pour le post-bac	22
6.6.2	Un tableau pour le collège et le lycée	24
7	Ensembles et applications	25
7.1	Différents types d'ensembles	25
7.1.1	Ensembles versus accolades	25
7.1.2	Ensembles pour la géométrie	25
7.1.3	Ensembles probabilistes	25
7.1.4	Ensembles pour l'algèbre générale	26
7.2	Ensembles classiques en mathématiques et en informatique théorique	26
7.2.1	La liste complète	26
7.2.2	Ensembles classiques suffixés	26
7.3	Des suffixes à la carte	27
7.4	Intervalles	27
7.4.1	Intervalles réels - Notation française (?)	27
7.4.2	Intervalles réels - Notation américaine	28
7.4.3	Intervalles discrets d'entiers	28
7.5	Unions et intersections	28
7.6	Cardinal, image et compagnie	29
7.7	Application totale, partielle, injective, surjective et/ou bijective	29
8	Géométrie	30
8.1	Points et lignes	30
8.1.1	Points	30
8.1.2	Lignes	30
8.1.3	Droites parallèles ou non	31
8.2	Vecteurs	31
8.2.1	Les écrire	31
8.2.2	Norme	32
8.2.3	Produit scalaire	32
8.2.4	Produit vectoriel	32

8.2.5	Plan – Déterminant de deux vecteurs	34
8.3	Coordonnées	36
8.4	Nommer un repère	37
8.5	Arcs circulaires	39
8.6	Angles	39
8.6.1	Angles géométriques « intérieurs »	39
8.6.2	Angles orientés de vecteurs	39
9	Analyse	40
9.1	Constantes et paramètres	40
9.1.1	Constantes classiques	40
9.1.2	Constantes latines personnelles	40
9.2	La fonction valeur absolue	40
9.3	Fonctions nommées spéciales	41
9.3.1	Sans paramètre	41
9.3.2	Avec un paramètre	41
9.3.3	Toutes les fonctions nommées en plus	41
9.4	Des notations complémentaires pour des suites spéciales	41
9.5	Calcul différentiel	42
9.5.1	Les opérateurs ∂ et d	42
9.5.2	Dérivations totales d'une fonction – Version longue mais polymorphe	42
9.5.3	Dérivations totales d'une fonction – Version courte pour les écritures standard	43
9.5.4	L'opérateur de dérivation totale	43
9.5.5	Dérivations partielles	43
9.5.6	L'opérateur de dérivation partielle	44
9.6	Calcul intégral	44
9.6.1	Intégrales multiples	44
9.6.2	Un opérateur d'intégration clés en main	45
9.6.3	L'opérateur crochet	45
9.7	Tableaux de variation et de signe	46
9.8	Comparaison asymptotique de suites et de fonctions	50
9.8.1	Les notations \mathcal{O} et \mathcal{o}	50
9.8.2	La notation Ω	50
9.8.3	La notation Θ	51
10	Probabilité	51
10.1	Probabilité « simple »	51
10.2	Probabilité conditionnelle	51
10.3	Espérance	52
10.4	Arbres pondérés	52
11	Arithmétique	55
11.1	Opérateurs de base	55
11.2	Fractions continuées	55
11.2.1	Fractions continuées standard	55
11.2.2	Fractions continuées généralisées	55
11.2.3	Comme une fraction continuée isolée	56
11.2.4	L'opérateur \mathcal{K}	56

12 Algèbre	57
12.1 Polynômes, séries formelles et compagnie	57
12.1.1 Polynômes et fractions polynômiales	57
12.1.2 Séries formelles et leurs corps de fractions	57
12.1.3 Polynômes de Laurent et séries formelles de Laurent	57
12.2 Matrices	58
12.2.1 Calculs expliqués des déterminants 2×2	58
12.2.2 Quelques exemples pour bien démarrer	59
13 Historique	62
14 Toutes les fiches techniques	68
14.1 Quelques modifications générales	68
14.1.1 Deux séparateurs d'arguments par défaut	68
14.1.2 Espace et fractions	68
14.1.3 Espace et racines n-ièmes d'un réel	68
14.1.4 Espace après la négation logique	68
14.1.5 Sommes et produits en mode ligne	68
14.2 Logique et fondements	68
14.2.1 Les textes pour les opérateurs de « comparaison algébrique » et de logique . . .	68
14.2.2 Les opérateurs de « comparaison algébrique »	69
14.2.3 Les opérateurs de logique	70
14.2.4 Les opérateurs de logique « verticaux »	71
14.2.5 Des versions alternatives du quantificateur \exists	72
14.2.6 Détailler un raisonnement simple	72
14.2.7 Détailler un raisonnement simple – Mise en forme du texte	73
14.2.8 Détailler un « vrai » raisonnement via un tableau	73
14.2.9 Détailler un « vrai » raisonnement via un tableau - Textes utilisés	74
14.3 Ensembles et applications	74
14.3.1 Ensembles versus accolades	74
14.3.2 Ensembles pour la géométrie	74
14.3.3 Ensembles probabilistes	75
14.3.4 Ensembles pour l'algèbre générale	75
14.3.5 Ensembles classiques	75
14.3.6 Des suffixes à la carte	76
14.3.7 Intervalles réels - Notation française (?)	76
14.3.8 Intervalles réels - Notation américaine	77
14.3.9 Intervalles discrets d'entiers	77
14.3.10 Unions et intersections	78
14.3.11 Cardinal, image et compagnie	78
14.3.12 Application totale, partielle, injective, surjective et/ou bijective	78
14.4 Géométrie	78
14.4.1 Points	78
14.4.2 Lignes	79
14.4.3 Droites parallèles ou non	79
14.4.4 Vecteurs	79
14.4.5 Norme	79
14.4.6 Produit scalaire	80
14.4.7 Produit vectoriel	80
14.4.8 Plan – Déterminant de deux vecteurs	81

14.4.9	Coordonnées	82
14.4.10	Nommer un repère	82
14.4.11	Arcs circulaires	83
14.4.12	Angles géométriques « intérieurs »	83
14.4.13	Angles orientés de vecteurs	84
14.5	Analyse	84
14.5.1	Constantes classiques	84
14.5.2	Constantes latines personnelles	84
14.5.3	Sans paramètre	84
14.5.4	Avec un paramètre	85
14.5.5	Des suites spéciales	85
14.5.6	Calcul différentiel	85
14.5.7	Dérivations totales	86
14.5.8	Opérateur de dérivation totale	86
14.5.9	Dérivations partielles	86
14.5.10	Opérateur de dérivation partielle	87
14.5.11	Le symbole d'intégration standard	87
14.5.12	Fonctionnelle d'intégration	87
14.5.13	L'opérateur crochet	87
14.5.14	Les notations \mathcal{O} et \mathcal{o}	88
14.5.15	La notation Ω	88
14.5.16	La notation Θ	88
14.6	Probabilité	88
14.6.1	Probabilité « simple »	88
14.6.2	Probabilité conditionnelle	88
14.6.3	Espérance	88
14.6.4	Arbres pondérés	88
14.7	Arithmétique	89
14.7.1	Opérateurs de base	89
14.7.2	Fractions continuées standard	89
14.7.3	Fractions continuées généralisées	89
14.7.4	Comme une fraction continuée isolée	89
14.7.5	L'opérateur \mathcal{K}	89
14.8	Algèbre	89
14.8.1	Polynômes, séries formelles et compagnie	89
14.8.2	Calculs expliqués des déterminants 2×2	90

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  $f'(x) = \cos(x^2)$  sur  $[a ; b]$  , nous avons :  
 $$ 
```

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

```
Sachant que  $\text{sder}\{f\}(x) = \cos(x^2)$  sur  $\text{intervalC}\{a\}\{b\}$ , nous avons :  
 $$ 
```

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

1. La mise en forme dans votre document devient consistante.
2. Il est facile de changer une mise en forme sur l'ensemble d'un document ou localement via certaines options.
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 (*voir la section dédiée 14*). Les exemples se présentent comme ci-dessous et sont généralement très courts.

```
 $$$$$$ 
```

$$\int_a^b \cos(x^2) dx = [f(x)]_a^b$$

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 suivent les règles suivantes.

1. Un nom de base explicite est choisi comme par exemple `\dotproduct` pour « *produit scalaire* » en anglais ou `\set` pour « *ensemble* » en anglais.
2. Si besoin, on spécialise du point de vue sémantique avec un préfixe et/ou un suffixe. Voici deux exemples.
 - (a) Dans `\vdotproduct`, le préfixe `v` est pour `v`-ecteur car cette macro s'utilise avec des noms de vecteurs `u` et `v` et non directement des vecteurs `\vect{u}` et `\vect{v}`.

- (b) Dans `\setproba`, le suffixe `proba` est pour proba-bilité car cette macro sert à écrire des ensembles munis d'une probabilité¹.
3. Si l'on propose différentes mises en forme pour une même signification sémantique alors ceci se fera via des versions étoilées et/ou par le biais d'option(s) comme dans `\dotproduct[r]` pour obtenir des produits scalaires utilisant des chevrons (*r est pour r-after soit « chevron » en anglais*).

3.1.2 Les formes « négatives » des macros

Les formes « négatives » des macros auront un nom préfixé par la lettre `n` en référence à `n-ot`. C'est l'usage dans le monde L^AT_EX comme par exemple pour `\neq`.

3.1.3 Les macros en mode `displaystyle`

Les macros évitant d'avoir à taper `\displaystyle` auront un nom préfixé par la lettre `d` comme par exemple dans `\dintegrate`.

3.1.4 Les macros standards redéfinies

Certaines macros comme `\frac` sont un peu revues par `lymath`. Dans ce cas, les versions standard restent accessibles en utilisant le préfixe `std` ce qui donne ici la macro `\stdfrac`.

3.2 Les arguments : deux conventions à connaître

3.2.1 Avec un nombre fixé d'arguments

Dans ce cas, c'est la syntaxe L^AT_EX usuelle qui sera à utiliser comme dans `\dotprod{u}{v}`.

3.2.2 Avec un 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 `lymath` comme étant formé des quatre arguments `x`, `y`, `z` et `t`.

3.2.3 Important

Certaines macros pourront demander des arguments non utilisés pour la mise en forme. Pourquoi cela? Tout simplement pour permettre des copier-coller lors de la rédaction : voir par exemple le calcul du déterminant de deux vecteurs du plan pour vérifier leur colinéarité (*ceci est présenté dans la section 8.2.5*).

4 Couleurs

Certaines macros utilisent de la couleur. Les choix faits sont tels que l'impression en noir et blanc ne soit pas impactée.

1. Ce choix est assumé même si on obtient un nom faisant penser à « régler ... » au lieu de « ensemble de type ... ».

5 Quelques modifications générales

5.1 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.

5.2 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!

5.3 Espace et fractions

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.

```
\frac{2}{3} = \stdfrac{2}{3}
```

$$\frac{2}{3} = \frac{2}{3}$$

```
\dfrac{2}{3} = \stdfrac{2}{3}
```

$$\frac{2}{3} = \frac{2}{3}$$

5.4 Espace et racines n-ièmes d'un réel

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

```
\sqrt{2} = \stdsqrt{2}
```

$$\sqrt{2} = \sqrt{2}$$

```
\sqrt[n]{45} = \stdsqrt[n]{45}
```

$$\sqrt[n]{45} = \sqrt[n]{45}$$

5.5 Espace après la négation logique

`\neg` a été redéfinie pour ajouter un peu d'espace après le symbole. Le comportement par défaut se retrouve en utilisant la macro `\stdneg`. Voici un exemple.

```
\neg A = \stdneg A
```

$$\neg A = \neg A$$

```
\neg\neg A = \stdneg \stdneg A
```

$$\neg\neg A = \neg\neg A$$

5.6 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.


```

 $\sum_{k=0}^n 2^k$ 
= \sum_{k=0}^n 2^k

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

```

$$\sum_{k=0}^n 2^k = \sum_{k=0}^n 2^k$$

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

6 Logique et fondements

6.1 Différents types de comparaisons « 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 éducative.

6.1.1 Définir quelque chose

L'exemple suivant montre deux façons de rédiger une égalité signifiant une définition (*la section 6.1.9 explique comment est défini le texte « déf »*).

```

 $f(x) \stackrel{\text{def}}{=} x^3 + 1$ 
 $f(x) \stackrel{\text{def}}{:=} x^3 + 1$ 

```

$$f(x) \stackrel{\text{def}}{=} x^3 + 1$$

$$f(x) \stackrel{\text{def}}{:=} x^3 + 1$$

6.1.2 Indiquer une identité

L'exemple suivant montre deux façons de rédiger des identités avec une notation symbolique non standard (*la section 6.1.9 explique comment est défini le texte « id »*).

```

 $(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2ab$ 
 $(a + b)^2 \stackrel{\text{id}}{:=} a^2 + b^2 + 2ab$ 

```

$$(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2ab$$

$$(a + b)^2 \stackrel{\text{id}}{:=} a^2 + b^2 + 2ab$$

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

Se reporter à la section 6.1.9 pour savoir comment sont définis les textes « cons », « cond » et « hyp ».

```

 $(a + b)^3 \stackrel{\text{test}}{=} a^3 + b^3 + 3ab$ 
 $(a + b)^3 \stackrel{\text{neq id}}{\neq} a^3 + b^3 + 3ab$ 

 $x \stackrel{\text{neq hyp}}{\neq} 0$  ou
 $x \stackrel{\text{neq cond}}{\neq} 0$  ou
 $x \stackrel{\text{eq cons}}{=} 0$ 

```

$$(a + b)^3 \stackrel{?}{=} a^3 + b^3 + 3ab$$

$$(a + b)^3 \stackrel{\text{id}}{\neq} a^3 + b^3 + 3ab$$

$$x \stackrel{\text{hyp}}{\neq} 0 \text{ ou } x \stackrel{\text{cond}}{\neq} 0 \text{ ou } x \stackrel{\text{cons}}{=} 0$$

6.1.4 Une égalité indiquant le choix d'une valeur ou l'application d'une relation

La section 6.1.9 permet de savoir comment les textes « choix » et « appli » sont définis.

```
$x \geqcond 4$ implique
$x^2 \geqcons 16$.
```

$x \overset{\text{cond}}{\geq} 4$ implique $x^2 \overset{\text{cons}}{\geq} 16$.
Donc $x \overset{\text{choix}}{=} 123$ donne $123^2 \overset{\text{appli}}{\geq} 16$.

```
Donc $x \eqchoice 123$ donne
$123^2 \eqappli 16$.
```

6.1.5 Une égalité indiquant l'équation d'une courbe

La section 6.1.9 permet de savoir comment les texte « *graph* » est défini (la macro `\setgeo` est présentée dans la section 7.1.2).

```
$M \in \setgeo{C}: y \eqplot x^2 + 3$
donne
$y_M \eqappli x_M^2 + 3$.
```

$M \in \mathcal{C} : y \overset{\text{graph}}{=} x^2 + 3$ donne $y_M \overset{\text{appli}}{=} x_M^2 + 3$.

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.

```
$x \leqtest x^2$ ou $x \lesscons x^2$ ou
$x \geqhyp 1$ ou $x \gtrcond 2$.
```

$x \overset{?}{\leq} x^2$ ou $x \overset{\text{cons}}{<} x^2$ ou $x \overset{\text{hyp}}{\geq} 1$ ou $x \overset{\text{cond}}{>} 2$.

6.1.7 Des formes négatives aussi pour les inéquations

Tous les opérateurs de comparaison ont une forme négative qui s'obtient en préfixant le nom de l'opérateur par `n`. Voici quelques exemples d'utilisation.

```
$x \nlesshyp 3$ ou
$y \nleqtest 4$ ou
$z \ngeqcons 5$
```

$x \overset{\text{hyp}}{\not\leq} 3$ ou $y \overset{?}{\not\leq} 4$ ou $z \overset{\text{cons}}{\not\geq} 5$

6.1.8 Une table récapitulative

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

6.1.9 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 »
\textopchoice donne « choix »
\textopcond donne « cond »
\textopcons donne « cons »
\textopdef donne « déf »
```

```
\textophyp donne « hyp »
\textopid donne « id »
\textopplot donne « graph »
\textoptest donne « ? »
```

6.2 Équivalences et implications

6.2.1 Des symboles supplémentaires

Exemple 1 – 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 \Leftarrow ², ainsi que des versions négatives. Voici un exemple d'utilisation.

<code>\$(A \implies B)</code> <code>\iff (B \liesimp A)\$</code>	$(A \Rightarrow B) \iff (B \Leftarrow A)$
<code>\$(A \implies B)</code> <code>\niff (A \nimplies B)\$</code>	$(A \Rightarrow B) \niff (A \nRightarrow B)$

Exemple 2 – Des opérateurs décorés

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

<code>\$(A \iffappli B \niffchoix C)\$</code>	$A \overset{\text{appli}}{\iff} B \overset{\text{choix}}{\niff} C$
<code>\$(A \impliescond B \nimpliescons C)\$</code>	$A \overset{\text{cond}}{\implies} B \overset{\text{cons}}{\nimplies} C$
<code>\$(A \liesimphyp B \nliesimptest C)\$</code>	$A \overset{\text{hyp}}{\Leftarrow} B \overset{?}{\nLeftarrow} C$

6.2.2 Une table récapitulative

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

6.2.3 Équivalences et implications verticales

À quoi cela sert-il ?

Dans la section 6.5.1 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 (*tous les cas possibles ont été indiqués*). Bien entendu le préfixe `v` est pour vertical.

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

$\$ \backslash \text{existsone } x \backslash \text{in } E \$ \text{ pour}$ $\backslash \text{og il existe un seul } \$x \$ \backslash \text{fg.}$	$\exists ! x \in E$ pour « il existe un seul x ».
$\$ \backslash \text{existmulti}\{\leq 1\} y \backslash \text{in } F \$ \text{ pour}$ $\backslash \text{og il existe au plus un } \$y \$ \backslash \text{fg.}$	$\exists_{\leq 1} y \in F$ pour « il existe au plus un y ». $\exists_{=1} \stackrel{\text{déf}}{=} \exists !$
$\$ \backslash \text{existmulti}\{=1\} \backslash \text{eqdef } \backslash \text{existsone} \$$	

Versions négatives

$\$ \backslash \text{nexistsone} \$ \text{ pour}$ $\backslash \text{og il n'existe pas un unique } \backslash \text{fg.}$	$\nexists !$ pour « il n'existe pas un unique ».
$\$ \backslash \text{existmulti}\{\neq 1\} \backslash \text{eqdef } \backslash \text{nexistsone} \$$	$\exists_{\neq 1} \stackrel{\text{déf}}{=} \nexists !$
$\$ \backslash \text{nexistmulti}\{>4\} \$ \text{ pour}$ $\backslash \text{og il n'existe pas plus de quatre } \backslash \text{fg.}$	$\nexists_{>4}$ pour « il n'existe pas plus de quatre ». \nexists vient de <code>amssymb</code> .
$\$ \backslash \text{nexists} \$ \text{ vient de } \backslash \text{verb+amssymb+}.$	

6.5 Détailler un raisonnement simple

6.5.1 Version pour le lycée et après

Exemple 1 – Avec les réglages par défaut

L'environnement `explain` permet de détailler les étapes principales d'un calcul ou d'un raisonnement simple en s'appuyant sur la macro `\explnext` dont le nom vient de « *expl-ain next step* » soit « *expliquer la prochaine étape* » en anglais³. On dispose aussi de `\explnext*` pour des explications descendantes et/ou montantes⁴.

Ci-dessous se trouve un exemple, très farfelu vers la fin, où l'on utilise les réglages par défaut. Notons au passage que ce type de présentation n'est sûrement pas bien adaptée à un jeune public pour lequel une 2^e façon de détailler des calculs et/ou un raisonnement simple est proposée plus bas dans la section 6.5.2.

3. Cet environnement utilise aussi le package `witharrows` qui est très sympathique pour expliquer des étapes de calcul.

4. Les explications données ne doivent pas être trop longues car ce serait contre-productif.

```

\begin{explain}
(a + b)^2
\explnext{On utilise  $x^2 = x \cdot x$ .}
(a + b)(a + b)
\explnext*{Double développement depuis la parenthèse gauche.}%
{Double factorisation pas facile.}
a^2 + a b + b a + b^2
\explnext*{}%
{Commutativité du produit.}
a^2 + 2 a b + b^2
\explnext*{Commutativité de l'addition.}%
{}
a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{aligned}
 &(a + b)^2 \\
 = &\quad \{ \textit{On utilise } x^2 = x \cdot x. \} \\
 &(a + b)(a + b) \\
 = &\quad \left\{ \begin{array}{l} \downarrow \textit{Double développement depuis la parenthèse gauche.} \downarrow \\ \uparrow \quad \quad \textit{Double factorisation pas facile.} \quad \uparrow \end{array} \right\} \\
 &a^2 + ab + ba + b^2 \\
 = &\quad \{ \uparrow \textit{Commutativité du produit.} \uparrow \} \\
 &a^2 + 2ab + b^2 \\
 = &\quad \{ \downarrow \textit{Commutativité de l'addition.} \downarrow \} \\
 &a^2 + b^2 + 2ab
 \end{aligned}$$

Remarque. Il faut savoir que la mise en forme est celle d'une formule ce qui peut rendre service comme dans l'exemple suivant.

```

Un calcul avec un placement pouvant être
utile :
\begin{explain}
(a + b)^2
\explnext{Identité remarquable.}
a^2 + b^2 + 2 a b
\end{explain}

```

Un calcul avec un placement pouvant être
utile : $(a + b)^2$
= $\{ \textit{Identité remarquable.} \}$
 $a^2 + b^2 + 2ab$

Avec un retour à la ligne, il faudra donc si besoin gérer l'espacement vertical.

```

Mon calcul pas trop proche.

\medskip
\begin{explain}
(a + b)^2
\explnext{Identité remarquable.}
a^2 + b^2 + 2 a b
\end{explain}

```

Mon calcul pas trop proche.
 $(a + b)^2$
= $\{ \textit{Identité remarquable.} \}$
 $a^2 + b^2 + 2ab$

Remarque. Voici des petites choses à connaître sur les macros `\explnext` et `\explnext*`.

1. `\expltxt` est utilisée par `\explnext` pour mettre en forme le texte d'explication.
2. `\expltxtup` et `\expltxtdown` sont utilisées par `\explnext*` décorer les textes d'explication juste avant leur mise en forme finale via `\expltxtupdown`.
3. `\explnext` et `\explnext*` utilisent la macro constante `\expltxtspacein` pour l'espacement entre le symbole et la courte explication. Par défaut, cette macro vaut `2em`.

Exemple 2 – Utiliser un autre symbole globalement

L'environnement `explain` possède plusieurs options dont l'une est `ope` qui vaut `{=}` par défaut. Ceci permet de faire ce qui suit sans effort.

<pre>\begin{explain}[ope = \viff] x^2 + 10 x + 25 = 0 \explnext{Identité remarquable.} (x + 5)^2 = 0 \explnext{\$P^2 = 0\$ si et seulement si \$P = 0\$.} x = -5 \end{explain}</pre>	$x^2 + 10x + 25 = 0$ $\Updownarrow \quad \{ \textit{Identité remarquable.} \}$ $(x + 5)^2 = 0$ $\Updownarrow \quad \{ P^2 = 0 \textit{ si et seulement si } P = 0. \}$ $x = -5$
---	---

Exemple 3 – Juste utiliser des symboles

Si l'argument obligatoire de la macro `\explnext` est vide alors seul le symbole est affiché (*ne pas oublier les accolades vides*). Voici un court exemple de ceci.

<pre>\begin{explain}[ope = \viff] a^2 = b^2 \explnext{} a = \pm b \end{explain}</pre>	$a^2 = b^2$ \Updownarrow $a = \pm b$
---	--

Exemple 4 – Utiliser un autre symbole localement

La macro `\explnext` possède un argument optionnel qui utilise par défaut celui de l'environnement. En utilisant cette option, on choisit alors localement le symbole à employer. Voici un exemple d'utilisation complètement farfelu bien que correct.

<pre>\begin{explain}[ope = \viff] 0 \leq a < b \explnext[\vimplies]{% {Croissance de \$x^2\$ sur \$\mathbb{R}_+\$}.} a^2 < b^2 \explnext{} a^2 - b^2 < 0 \explnext{Identité remarquable.} (a - b)(a + b) < 0 \explnext[\vimplies]{} a \neq b \end{explain}</pre>	$0 \leq a < b$ $\Downarrow \quad \{ \textit{Croissance de } x^2 \textit{ sur } \mathbb{R}_+. \}$ $a^2 < b^2$ \Updownarrow $a^2 - b^2 < 0$ $\Updownarrow \quad \{ \textit{Identité remarquable.} \}$ $(a - b)(a + b) < 0$ \Downarrow $a \neq b$
---	--

Exemple 5 – Choisir la mise en forme des explications

Pour la mise en forme des explications à double sens, la macro `\explnext` fait appel à la macro `\expltxt`. Par défaut, le package utilise la définition suivante.

```
\newcommand\expltxt[1]{%
  \text{\color{blue}\footnotesize \{\,\,\{\itshape #1\}\,\,\}}%
}
```

Pour la mise en forme des explications à sens unique, la macro `\explnext*` fait appel aux macros `\expltxtup`, `\expltxtdown` et `\expltxtupdown`. Par défaut le package utilise les définitions suivantes.

```
\newcommand\expltxtup[1]{%
  $\uparrow$ #1 $\uparrow$%
}

\newcommand\expltxtdown[1]{%
  $\downarrow$ #1 $\downarrow$%
}

\newcommand\expltxtupdown[2]{%
  \displaystyle\footnotesize\color{blue}%
  \left\{\,\,%
    \genfrac{}{}{0pt}{}{%
      \text{\itshape\expltxtdown{\samesizeas{#1}{#2}}}%
    }{%
      \text{\itshape\expltxtup{\samesizeas{#2}{#1}}}%
    }%
  \,\,\right\}%
}
```

Nous allons expliquer comment obtenir l'affreux exemple ci-dessous montrant que l'on peut adapter si besoin la mise en forme (*ne pas oublier de passer en mode texte via `\text`*).

<pre>\begin{explain} (a + b) (a + b) \explnext{Se souvenir de \$P\cdot\$ P = P^2\$.}% (a + b)^2 \explnext*{Id. Rm. - Dév.}% {Id. Rm. - Facto.} a^2 + 2 a b + b^2 \end{explain}</pre>	$ \begin{aligned} & (a + b)(a + b) \\ = & \quad \Downarrow \textit{Se souvenir de } P \cdot P = P^2. \quad \Uparrow \\ & (a + b)^2 \\ = & \quad \left\langle \begin{array}{c} \Downarrow \textit{Id. Rm. - Dév.} \Downarrow \\ \hline \Uparrow \textit{Id. Rm. - Facto.} \Uparrow \end{array} \right\rangle \\ & a^2 + 2ab + b^2 \end{aligned} $
--	--

La mise en forme a été obtenue en utilisant le code L^AT_EX suivant où la macro `\samesizeas{#1}{#2}` rend le texte `#1` aussi large que `#2` en ajoutant des espaces supplémentaires tout en centrant le résultat final si besoin.


```

\newcommand\myexpltxt[2]{%
  \text{\color{#1} \footnotesize \itshape \bfseries #2}%
}

\renewcommand\expltxt[1]{%
  \myexpltxt{gray}{\Downarrow$ #1 $\Uparrow$}%
}

\renewcommand\expltxtup[1]{%
  \myexpltxt{orange}{\Uparrow$ #1 $\Uparrow$}%
}

\renewcommand\expltxtdown[1]{%
  \myexpltxt{red}{\Downarrow$ #1 $\Downarrow$}%
}

\renewcommand\expltxtupdown[2]{%
  \displaystyle\color{blue!20!black!30!green}%
  \genfrac{\langle}{\rangle}{1pt}{}{%
    \expltxtdown{\samesizeas{#1}{#2}}%
  }{%
    \expltxtup{\samesizeas{#2}{#1}}%
  }%
}

```

6.5.2 Version pour les collégiens

L'environnement `explain` avec l'option `style = ar`⁵ utilise des flèches pour indiquer les explications (*ar est pour ar-row soit « flèche » en anglais*). Dans ce cas d'utilisation, la macro `\explnext*` permet d'avoir une flèche unidirectionnelle, vers le haut ou le bas au choix, ou bien d'écrire deux indications dont l'une est montante et l'autre descendante.

Il existe aussi l'option `style = sar` lorsque la toute 1^{re} étape n'est pas expliquée (*s est pour s-short soit « court » en anglais*). Attention car forcément ceci nécessite au tout début de l'environnement l'usage de la macro `\explnext` sans aucun contenu !

Exemple 1 – Flèche à double sens ou sans flèche

```

\begin{explain}[style = ar]
  (a + b)^2
  \explnext{Identité remarquable}
  a^2 + 2 a b + b^2
  \explnext{}
  a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{aligned}
 &(a + b)^2 \\
 &= a^2 + 2ab + b^2 \quad \text{↕ Identité remarquable} \\
 &= a^2 + b^2 + 2ab
 \end{aligned}$$

Exemple 2 – Des flèches unidirectionnelles

Ce qui suit est juste là comme démo. car les explications y sont un peu farfelues.

5. Cet environnement utilise aussi le package `witharrows`.

```

\begin{explain}[style = ar]
  (a + b)^2
    \explnext*{Via  $P^2 = P \cdot P$ .}
    {Via  $P \cdot P = P^2$ .}
  (a + b) (a + b)
    \explnext*{Double développement.}%
    {Double factorisation (pas simple).}
  a^2 + a b + b a + b^2
    \explnext*{Commutativité du produit.}%
    {}
  a^2 + 2 a b + b^2
    \explnext*{}%
    {Commutativité de l'addition.}
  a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{array}{ll}
 (a + b)^2 & \\
 = (a + b)(a + b) & \left. \begin{array}{l} \text{Via } P^2 = P \cdot P. \\ \text{Via } P \cdot P = P^2. \end{array} \right\} \\
 = a^2 + ab + ba + b^2 & \left. \begin{array}{l} \text{Double développement.} \\ \text{Double factorisation (pas simple).} \end{array} \right\} \\
 = a^2 + 2ab + b^2 & \left. \begin{array}{l} \text{Commutativité du produit.} \\ \text{Commutativité de l'addition.} \end{array} \right\} \\
 = a^2 + b^2 + 2ab &
 \end{array}$$

Exemple 3 – Ne pas expliquer le tout début

L'environnement étoilé `explain` avec l'option `style = sar` débute différemment la mise en forme. Bien entendu ici le tout premier doit avoir un argument vide !

```

\begin{explain}[style = sar]
  (a + b) (a + b)
    \explnext{}
  (a + b)^2
    \explnext{Identité remarquable.}
  a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{array}{ll}
 (a + b)(a + b) = (a + b)^2 & \\
 = a^2 + b^2 + 2ab & \left. \begin{array}{l} \\ \end{array} \right\} \text{Identité remarquable.}
 \end{array}$$

Exemple 4 – Choisir son symbole

Voici comment faire où l'implication finale est juste là pour la démonstration (*on notera une petite bidouille un peu sale à faire pour avoir un alignement à peu près correct*).

```

\begin{explain}[style = ar, ope = \iff]
  a^2 + 2 a b + b^2 = 0
  \explnext{}
  (a + b)^2 = 0
  \explnext[\:\implies]%
    {$P^2 = 0$ ssi $P = 0$.}

  a + b = 0
\end{explain}

```

$$\begin{aligned}
 & a^2 + 2ab + b^2 = 0 \\
 \iff & (a + b)^2 = 0 \quad \left. \vphantom{(a + b)^2 = 0} \right\} P^2 = 0 \text{ ssi } P = 0. \\
 \implies & a + b = 0
 \end{aligned}$$

Avec la version courte, on obtient ce qui suit.

```

\begin{explain}[style = sar, ope = \iff]
  a^2 + 2 a b + b^2 = 0
  \explnext{}
  (a + b)^2 = 0
  \explnext[\:\implies]%
    {$P^2 = 0$ ssi $P = 0$.}

  a + b = 0
\end{explain}

```

$$\begin{aligned}
 a^2 + 2ab + b^2 = 0 & \iff (a + b)^2 = 0 \\
 & \implies a + b = 0 \quad \left. \vphantom{(a + b)^2 = 0} \right\} P^2 = 0 \text{ ssi } P = 0.
 \end{aligned}$$

6.5.3 De courts commentaires

Exemple 1 – Sans alignement

Il est possible d'ajouter de petits commentaires via `\comthis` où `comthis` est pour `com-ment this` soit « *commenter ceci* » en anglais.

```

\begin{explain}
  (a + b)^2
  \comthis{Forme facto.}
  \explnext*{Id.Rq. -- Dév.}%
    {Id.Rq. -- Facto.}

  a^2 + 2 a b + b^2
  \comthis{Forme dév.}
\end{explain}

```

$$\begin{aligned}
 & (a + b)^2 \quad [\textit{Forme facto.}] \\
 = & \quad \left\{ \begin{array}{l} \downarrow \textit{Id.Rq.} - \textit{Dév.} \downarrow \\ \uparrow \textit{Id.Rq.} - \textit{Facto.} \uparrow \end{array} \right\} \\
 & a^2 + 2ab + b^2 \quad [\textit{Forme dév.}]
 \end{aligned}$$

Remarque. La mise en forme du texte des commentaires est fait via la macro personnalisable `\explcom`. Quant à l'espacement ajouté entre le texte et son commentaire il est défini par la macro `\expltxtspacein` qui est égale à 2em par défaut.

Exemple 2 – Tout aligner

Il peut être utile d'aligner tous les commentaires. Ceci s'obtient via l'option `com = al` où `al` est pour aligné (par défaut `com = nal` avec le préfixe *n* pour *n-on*).

<pre> \begin{explain}[com = al] (a + b)^2 \comthis{Forme facto.} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$(a + b)^2 \quad [\textit{Forme facto.}]$ $= \left\{ \begin{array}{c} \downarrow \textit{Id.Rq - Dév.} \downarrow \\ \uparrow \textit{Id.Rq - Facto.} \uparrow \end{array} \right.$ $a^2 + 2ab + b^2 \quad [\textit{Forme dév.}]$
---	---

Exemple 3 – Le meilleur des deux mondes

Dans d'autres situations, utiliser les deux types d'alignement peut faire sens. Ceci s'obtient via l'option `com = al` et l'emploi de la macro étoilée `\comthis*` à chaque fois que l'on souhaite "coller" un commentaire le plus à gauche possible.

<pre> \begin{explain}[com = al] (a + b) (a + b) \comthis{Forme facto.} \explnext{Via \$x^2 = x \cdot x\$.} (a + b)^2 \comthis*{Au passage...} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$(a + b)(a + b) \quad [\textit{Forme facto.}]$ $= \quad \{ \textit{Via } x^2 = x \cdot x. \}$ $(a + b)^2 \quad [\textit{Au passage...}]$ $= \left\{ \begin{array}{c} \downarrow \textit{Id.Rq - Dév.} \downarrow \\ \uparrow \textit{Id.Rq - Facto.} \uparrow \end{array} \right.$ $a^2 + 2ab + b^2 \quad [\textit{Forme dév.}]$
--	--

Remarque. Si l'alignement n'est pas activé, les macros `\comthis*` et `\comthis` auront toutes les deux le même effet.

Exemple 4 – Ceci marche aussi avec le style « fléché »

Voici ce que donne le mode mixte lorsque des flèches sont utilisées pour les explications. Il semble moins pertinent ici de mixer les modes « alignement » et « non alignement » mais chacun pris séparément peut avoir son utilité.

<pre> \begin{explain}[style = ar, com = al] (a + b) (a + b) \comthis{Forme facto.} \explnext{Via \$x^2 = x \cdot x\$.} (a + b)^2 \comthis*{Au passage...} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$(a + b)(a + b) \quad [\textit{Forme facto.}]$ $= (a + b)^2 \quad [\textit{Au passage...}]$ $= a^2 + 2ab + b^2 \quad [\textit{Forme dév.}]$
$\begin{array}{l} \text{ } \end{array} \left. \begin{array}{l} \text{ } \end{array} \right\} \textit{Via } x^2 = x \cdot x.$ $\begin{array}{l} \text{ } \end{array} \left. \begin{array}{l} \text{ } \end{array} \right\} \textit{Id.Rq - Dév.} \quad \left. \begin{array}{l} \text{ } \end{array} \right\} \textit{Id.Rq - Facto.}$	

Remarque. Bien entendu il est impossible de commenter le tout début en mode fléché court.

6.5.4 Un mini hack très utile pour des « étapes alignées »

Vous pouvez écrire très facilement des calculs ou raisonnement simples alignés comme suit sans trop vous fatiguer.

```
\begin{explain}[style = sar]
  (a + b) (a + b)
  \explnext{}
  (a + b)^2
  \explnext{}
  a^2 + b^2 + 2 a b
  \comthis{Pourquoi ?}
  \explnext{}
  a^2 + 2 a b + b^2
\end{explain}
```

$$\begin{aligned}(a + b)(a + b) &= (a + b)^2 \\ &= a^2 + b^2 + 2ab \quad [\text{Pourquoi ?}] \\ &= a^2 + 2ab + b^2\end{aligned}$$

On a accès à une autre mise en forme (*ceci peut rendre aussi service*).

```
\begin{explain}[style = ar]
  (a + b) (a + b)
  \explnext{}
  (a + b)^2
  \explnext{Pourquoi ?}
  a^2 + b^2 + 2 a b
  \explnext{}
  a^2 + 2 a b + b^2
\end{explain}
```

$$\begin{aligned}(a + b)(a + b) \\ &= (a + b)^2 \\ &= a^2 + b^2 + 2ab \quad \left. \vphantom{a^2 + b^2 + 2ab} \right\} \text{Pourquoi ?} \\ &= a^2 + 2ab + b^2\end{aligned}$$

Enfin dans le cadre de calculs à faire expliquer par des élèves, ce qui suit peut être utile.

Donner les justifications J1, J2 et J3.

```
\medskip
\begin{explain}
  (a + b) (a + b)
  \explnext{J1}
  (a + b)^2
  \explnext{J2}
  a^2 + b^2 + 2 a b
  \explnext{J3}
  a^2 + 2 a b + b^2
\end{explain}
```

Donner les justifications J1, J2 et J3.

$$\begin{aligned}(a + b)(a + b) \\ &= \{ J1 \} \\ (a + b)^2 \\ &= \{ J2 \} \\ a^2 + b^2 + 2ab \\ &= \{ J3 \} \\ a^2 + 2ab + b^2\end{aligned}$$

6.5.5 Un conseil de mise en forme

Voici un style de codage que nous trouvons très facile à relire et maintenir.

```

\begin{explain}[com = al]
  (a + b) (a + b)
  \comthis{Forme facto.}
  \explnext{Via  $x^2 = x \cdot x$ .}
%
  (a + b)^2
  \comthis*{Au passage...}
  \explnext*{Id.Rq - Dév.}%
  {Id.Rq - Facto.}
%
  a^2 + 2 a b + b^2
  \comthis{Forme dév.}
\end{explain}

```

$$\begin{aligned}
 & (a + b)(a + b) && [\textit{Forme facto.}] \\
 = & \{ \textit{Via } x^2 = x \cdot x. \} \\
 & (a + b)^2 && [\textit{Au passage...}] \\
 = & \left\{ \begin{array}{c} \downarrow \textit{Id.Rq - Dév.} \downarrow \\ \uparrow \textit{Id.Rq - Facto.} \uparrow \end{array} \right\} \\
 & a^2 + 2ab + b^2 && [\textit{Forme dév.}]
 \end{aligned}$$

6.6 Détailler un « vrai » raisonnement

6.6.1 Un tableau pour le post-bac

Exemple 1 – Le minimum avec les réglages par défaut

Prenons un exemple utile à la logique formelle en informatique théorique mais qui a complètement sa place en mathématiques plus classiques (*voir la section 6.6.2 pour un autre type de présentation plus adaptée à un public de collège ou de lycée*). Ci-dessous l’environnement `demoexplain` facilite la mise en page via l’environnement `tabular` utilisé en coulisse, et la macro étoilée `\explref*` permet d’indiquer une référence interne au raisonnement⁶. Dans cet exemple en deux morceaux, pour montrer au passage comment continuer la numérotation là où elle s’était arrêtée, on utilise « *m.p.* » comme abréviation de « *modus ponens* ».

```

\begin{demoexplain}
  \demostep
    Hypothèse & $$
  \demostep
    Axiome 1 & $A \implies B$
  \demostep
    m.p. sur
    \explref*{1} et \explref*{2}
    & $$
  \demostep
    \explref*{1} et \explref*{3}
    & $A \wedge B$
\end{demoexplain}

```

1	Hypothèse	A
2	Axiome 1	$A \implies B$
3	m.p. sur 1 et 2	B
4	1 et 3	$A \wedge B$

Il est possible de couper sa démonstration en morceaux en indiquant à l’environnement la valeur du 1^{er} numéro de justification via la clé `start` : la valeur spéciale `last` indique de continuer la numérotation à la suite.

6. Ceci permet de numéroter les indications jusqu’à 99 ce qui est bien au-delà des besoins pratiques.

```

\begin{demoexplain}[start = last]
  \demostep
    Axiome 3
    &  $(A \wedge B) \implies C$ 
  \demostep
    m.p. sur
    \explref*{4} et \explref*{5}
    &  $C$ 
\end{demoexplain}

```

5	Axiome 3	$(A \wedge B) \implies C$
6	m.p. sur 4 et 5	C

Exemple 2 – Référencer une indication

L'argument optionnel de `\demostep` permet de définir un label qui ensuite facilitera le référencement d'une justification de façon pérenne via la macro non étoilée `\explref`.

```

\begin{demoexplain}
  \demostep[demo-my-hyp]
    Hypothèse &  $A$ 
  \demostep[demo-use-axiom-1]
    Axiome 1 &  $A \implies B$ 
  \demostep
    m.p. sur
    \explref{demo-my-hyp}
    et
    \explref{demo-use-axiom-1}
    &  $B$ 
\end{demoexplain}

```

1	Hypothèse	A
2	Axiome 1	$A \implies B$
3	m.p. sur 1 et 2	B

Remarque. Prendre bien garde au fait que ce mécanisme utilise les macros `\label` et `\ref` de L^AT_EX. On travaille donc avec des références globalement au document compilé.

Exemple 3 – Indiquer ce que l'on cherche à faire

Les clés optionnelles `hyps` pour plusieurs hypothèses, `hyp` pour une seule hypothèse et `ccl` pour la conclusion permettent d'expliquer ce que l'on démontre et sous quel contexte.

```

\begin{demoexplain}[hyp =  $A$ , ccl =  $B$ ]
  \demostep
    Hypothèse &  $A$ 
  \demostep
    Axiome 1 &  $A \implies B$ 
  \demostep
    m.p. sur
    \explref*{1} et \explref*{2}
    &  $B$ 
\end{demoexplain}

```

Démonstration sous l'hypothèse : A

1	Hypothèse	A
2	Axiome 1	$A \implies B$
3	m.p. sur 1 et 2	B

Conclusion : B

Remarque. Aucune des clés `hyps`, `hyp` et `ccl` n'est obligatoire. Par contre il n'est pas possible d'utiliser à la fois les clés `hyps` et `hyp`.

6.6.2 Un tableau pour le collège et le lycée

Exemple 1 – Avec les réglages par défaut

L'environnement étoilé `demoexplain*` est différent de l'environnement `demoexplain` puisqu'il sert à indiquer trois choses et non juste deux comme le montre l'exemple suivant ⁷. Par contre, la syntaxe est très similaire. Notez au passage la possibilité d'utiliser `\newline` pour forcer un retour à la ligne dans une cellule.

```
\begin{demoexplain*}
  \demostep
    $ABC$ est un triangle
    \newline équilatéral
    & Dans un triangle équilatéral, les trois angles mesurent $60$\textdegree.
    & $\anglein{ABC} = 60$\textdegree
  \demostep
    Voir la conséquence \explref*{1} .
    & Simple calcul avec conversion en radians.
    & $\dfrac{1}{3} \anglein{ABC} = \dfrac{\pi}{9}$
\end{demoexplain*}
```

Réf.	Je sais que...	Propriété ou fait utilisé	Conséquence
1	ABC est un triangle équilatéral	Dans un triangle équilatéral, les trois angles mesurent 60° .	$\widehat{ABC} = 60^\circ$
2	Voir la conséquence 1 .	Simple calcul avec conversion en radians.	$\frac{1}{3} \widehat{ABC} = \frac{\pi}{9}$

Exemple 2 – Avec toutes les options

Le système de référence marche ici aussi. Par contre `demoexplain*` ne propose que `start` comme clé optionnelle avec le même fonctionnement que pour `demoexplain`.

```
\begin{demoexplain*}[start = last]
  \demostep[demo-first-geo-fact]
    $ABC$ est un triangle \newline équilatéral
    & Dans un triangle équilatéral, les trois angles mesurent $60$\textdegree.
    & $\anglein{ABC} = 60$\textdegree
  \demostep
    Voir la conséquence \explref{demo-first-geo-fact} .
    & Simple calcul avec conversion en radians.
    & $\dfrac{1}{3} \anglein{ABC} = \dfrac{\pi}{9}$
\end{demoexplain*}
```

Réf.	Je sais que...	Propriété ou fait utilisé	Conséquence
3	ABC est un triangle équilatéral	Dans un triangle équilatéral, les trois angles mesurent 60° .	$\widehat{ABC} = 60^\circ$
4	Voir la conséquence 3 .	Simple calcul avec conversion en radians.	$\frac{1}{3} \widehat{ABC} = \frac{\pi}{9}$

7. C'est pour cela qu'est proposé une version étoilée de l'environnement et non l'utilisation d'une option de l'environnement non étoilé.

7 Ensembles et applications

7.1 Différents types d'ensembles

7.1.1 Ensembles versus accolades

Exemple 1

<code>$\setgene{1 ; 3 ; 5}$</code>	$\{1;3;5\}$
---	-------------

Exemple 2

Dans l'exemple suivant on utilise l'option `sb` pour **s**-mall **b**-races soit « *petites accolades* » en anglais.

<code>$\setgene{\dfrac{1}{3} ; \dfrac{5}{7} ; \dfrac{9}{11}}$</code>	$\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$
<code>$\setgene*{\dfrac{1}{3} ; \dfrac{5}{7} ; \dfrac{9}{11}}$</code>	$\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$

7.1.2 Ensembles pour la géométrie

Exemple 1

<code>\setgeo{C}</code> , <code>\setgeo{D}</code> ou <code>\setgeo{d}</code>	\mathcal{C} , \mathcal{D} ou d
---	--------------------------------------

Remarque. Pour le moment, il n'est pas possible de taper `\setgeo{ABC}` avec plusieurs lettres.

Exemple 2 – Avec des indices

<code>$\setgeo*{C}{1}$</code> ou <code>$\setgeo*{C}{2}$</code>	\mathcal{C}_1 ou \mathcal{C}_2
---	------------------------------------

7.1.3 Ensembles probabilistes

Exemple 1

<code>\setproba{E}</code> ou <code>\setproba{G}</code>	\mathcal{E} ou \mathcal{G}
---	--------------------------------

Remarque. Pour le moment, il n'est pas possible de taper `\setproba{ABC}` avec plusieurs lettres.

Exemple 2 – Avec des indices

<code>$\setproba*{E}{1}$</code> ou <code>$\setproba*{E}{2}$</code>	\mathcal{E}_1 ou \mathcal{E}_2
---	------------------------------------

7.1.4 Ensembles pour l'algèbre générale

Exemple 1

\set{A} , \set{K} , \set{h} ou \set{k}	A , K , h ou k
---	------------------------

Remarque. Pour le moment, il n'est pas possible de taper \set{ABC} avec plusieurs lettres.

Exemple 2 – Avec des indices

$\set*[k]{1}$ ou $\set*[k]{2}$	k_1 ou k_2
--------------------------------	----------------

7.2 Ensembles classiques en mathématiques et en informatique théorique

7.2.1 La liste complète

Dans l'exemple suivant, \mathbb{P} désigne l'ensemble des nombres premiers, \mathbb{H} celui des quaternions, \mathbb{O} celui des octonions et \mathbb{F} un ensemble de nombres flottants (*notation à préciser suivant le contexte*).

\emptyset	
\mathbb{N} , \mathbb{Z} , \mathbb{P}	\mathbb{N} , \mathbb{Z} , \mathbb{P}
\mathbb{D} , \mathbb{Q} , \mathbb{R} , \mathbb{C}	\mathbb{D} , \mathbb{Q} , \mathbb{R} , \mathbb{C}
\mathbb{H} , \mathbb{O}	\mathbb{H} , \mathbb{O}
\mathbb{F}	\mathbb{F}

7.2.2 Ensembles classiques suffixés

L'ensemble \mathbb{R} nous permet de voir tous les cas possibles.

\mathbb{R}_n , \mathbb{R}_p , \mathbb{R}_s	\mathbb{R}_- , \mathbb{R}_+ , \mathbb{R}^*
\mathbb{R}_{sn} , \mathbb{R}_{sp}	\mathbb{R}_-^* , \mathbb{R}_+^*

Nous avons utilisé les suffixes **n** pour **n**-égatif, **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 \mathbb{C}_n 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 8 page suivante montre les associations autorisées entre ensembles classiques et suffixes.

TABLE 8 – Suffixes

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

7.3 Des suffixes à la carte

Exemple d'utilisation

Dans cet exemple, il faut savoir que le 2^e argument ne peut prendre que les valeurs **n**, **p**, **s**, **sn** ou **sp**.

```

 $\setsspecial{\CC}{n}$  ,
 $\setsspecial{\HH}{sp}$  ou
 $\setsspecial*{\setproba{P}}{n}$ 

```

\mathbb{C}_- , \mathbb{H}_+^* ou $\mathcal{P}_{\leq 0}$

7.4 Intervalles

7.4.1 Intervalles réels - Notation française (?)

Exemple 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**. Notez au passage que la macro utilisée résout un problème d'espacement vis à vis du signe = .

```
 $I = ]a ; b] = \intervalOC{a}{b}$ 
```

$I =]a ; b] =]a ; b]$

Exemple 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.

```

 $\displaystyle$ 
 $\intervalC{ \frac{1}{2} }{ 1^{2^3} } }$ 
=
 $[ \frac{1}{2} ; 1^{2^3} ]$ 
=
 $\intervalC*{ \frac{1}{2} }{ 1^{2^3} } }$ 

```

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

7.4.2 Intervalles réels - Notation américaine

Exemple

Dans cet exemple, la syntaxe fait référence à P-arenthèse. Cette notation est utilisée États Unis.

```
\intervalPC{a}{b} = \intervalOC{a}{b}$
et
\intervalP{a}{b} = \intervalO{a}{b}$.
```

$(a; b] =]a; b]$ et $(a; b) =]a; b[$.

7.4.3 Intervalles discrets d'entiers

Exemple

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

```
\ZintervalC{-1}{4}
\eqdef
\{ -1 ; 0 ; 1 ; 2 ; 3 ; 4 \}$

\ZintervalC{-1}{4}
= \ZintervalO{-2}{5}$.
```

$\llbracket -1; 4 \rrbracket \stackrel{\text{déf}}{=} \{-1; 0; 1; 2; 3; 4\}$
 $\llbracket -1; 4 \rrbracket = \llbracket -2; 5 \rrbracket$.

7.5 Unions et intersections

Exemple 1 – Des unions

Ci-dessous est utilisée la macro `\bigcup` proposée par le package `amssymb`.

```
$A \cup B$

$\dcup_{k=1}^n A_k$

$\bigcup_{k=1}^n B_k$

$\displaystyle \bigcup_{k=1}^n C_k$
```

$A \cup B$
 $\bigcup_{k=1}^n A_k$
 $\bigcup_{k=1}^n B_k$
 $\bigcup_{k=1}^n C_k$

Exemple 2 – Des unions disjointes

Ci-dessous sont utilisées les macros `\sqcup` et `\bigsqcup` proposée par le package `amssymb`.

```
$A \sqcup B$

$\dsqcup_{k=1}^n A_k$

$\bigsqcup_{k=1}^n B_k$

$\displaystyle \bigsqcup_{k=1}^n C_k$
```

$A \sqcup B$
 $\sqcup_{k=1}^n A_k$
 $\sqcup_{k=1}^n B_k$
 $\sqcup_{k=1}^n C_k$

Exemple 3 – Des intersections

Ci-dessous est utilisée la macro `\bigcap` proposée par le package `amssymb`.

<code>\$A \cap B\$</code>	$A \cap B$
<code>\$_{\displaystyle}\bigcap_{k=1}^n A_k\$</code>	$\bigcap_{k=1}^n A_k$
<code>\$_{\displaystyle}\bigcap_{k=1}^n B_k\$</code>	$\bigcap_{k=1}^n B_k$
<code>\$_{\displaystyle}\bigcap_{k=1}^n C_k\$</code>	$\bigcap_{k=1}^n C_k$

7.6 Cardinal, image et compagnie

Exemple 1 – Cardinal

<code>\$_{\displaystyle}\text{card} E = \text{card } E\$</code>	$\#E = \text{card } E$
---	------------------------

Exemple 2 – Image et compagnie

<code>\$_{\displaystyle}\ker f\$, \$_{\displaystyle}\text{dom } f\$, \$_{\displaystyle}\text{im } f\$ ou \$_{\displaystyle}\text{codom } f\$</code>	$\ker f$, $\text{dom } f$, $\text{im } f$ ou $\text{codom } f$
---	--

7.7 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 partielles⁸ (*on parle aussi d'applications, sans qualificatif, et de fonctions*).

Exemple 1 – Applications totales

<code>\$f: A \to B\$ est une application totale, c'est à dire définie sur \$A\$ tout entier.</code>
<code>\$i: C \hookrightarrow D\$ est une application totale injective.</code>
<code>\$s: E \twoheadrightarrow F\$ est une application totale surjective.</code>
<code>\$b: G \twoheadrightarrow H\$ est une application totale bijective.</code>

<code>$f : A \rightarrow B$ est une application totale, c'est à dire définie sur A tout entier.</code>
<code>$i : C \hookrightarrow D$ est une application totale injective.</code>
<code>$s : E \twoheadrightarrow F$ est une application totale surjective.</code>
<code>$b : G \twoheadrightarrow H$ est une application totale bijective.</code>

8. $a : E \rightarrow F$ est une application totale si $\forall x \in E, \exists ! y \in F$ tel que $y = a(x)$. Plus généralement, $f : E \rightarrow F$ est une application partielle 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 .

Exemple 2 – Applications partielles

`$f: A \pto B$` 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.

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

8 Géométrie

8.1 Points et lignes

8.1.1 Points

Exemple 1 – Sans indice

`\pt{I}`

I

Exemple 2 – Avec un indice

`$\pt*{I}{1}$` ou
`$\pt*{I}{2}$`

I_1 ou I_2

8.1.2 Lignes

Exemple 1 – Les droites

Dans l’exemple suivant, le préfixe g est pour g -éometrie tandis que p est pour p -oint.

`$\gline{A}{B}$` ,
`$\gline{\pt{A}}{\pt{B}}$` ou
`$\pgline{A}{B}$`

(AB) , (AB) ou (AB)

Exemple 2 – Les segments

Les macros `\segment` et `\psegment` ont un comportement similaire à `\gline` et `\pgline`.

`$\segment{A}{B}$` ,
`$\segment{\pt{A}}{\pt{B}}$` ou
`$\psegment{A}{B}$`

$[AB]$, $[AB]$ ou $[AB]$

Exemple 3 – Les demi-droites

Dans l'exemple suivant, le préfixe `h` est pour `h-alf` soit « *moitié* » en anglais.

$\backslash\mathrm{hgline}\{A\}\{B\}$, $\backslash\mathrm{hgline}\{\mathrm{pt}\{A\}\}\{\mathrm{pt}\{B\}\}$ ou $\backslash\mathrm{phgline}\{A\}\{B\}$	$[AB)$, $[AB)$ ou $[AB)$
---	---------------------------

Exemple 4 – D'autres demi-droites

Ce qui suit nécessite d'utiliser l'argument optionnel de `\gline` et `\pgline`. La valeur `OC` provient de `O-pened – C-losed` soit « *ouvert – fermé* » en anglais.

$\backslash\mathrm{gline}[OC]\{A\}\{B\}$, $\backslash\mathrm{gline}[OC]\{\mathrm{pt}\{A\}\}\{\mathrm{pt}\{B\}\}$ ou $\backslash\mathrm{pgline}[OC]\{A\}\{B\}$	$(AB]$, $(AB]$ ou $(AB]$
--	---------------------------

Remarque. Les segments utilisent en fait l'option `C` et les demi-droites standard l'option `CO`. La valeur par défaut est `O`.

8.1.3 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.

$\backslash\mathrm{pgline}\{A\}\{B\} \backslash\mathrm{parallel} \backslash\mathrm{pgline}\{C\}\{D\}$ au lieu de $\backslash\mathrm{pgline}\{A\}\{B\}$ $\backslash\mathrm{stdparallel} \backslash\mathrm{pgline}\{C\}\{D\}$	$(AB) \parallel (CD)$ au lieu de $(AB) \parallel (CD)$ $(EF) \nparallel (GH)$ au lieu de $(EF) \nparallel (GH)$
$\backslash\mathrm{pgline}\{E\}\{F\} \backslash\mathrm{nparallel} \backslash\mathrm{pgline}\{G\}\{H\}$ au lieu de $\backslash\mathrm{pgline}\{E\}\{F\}$ $\backslash\mathrm{stdnparallel} \backslash\mathrm{pgline}\{G\}\{H\}$	

8.2 Vecteurs

8.2.1 Les écrire

Exemple 1

$\backslash\mathrm{vect}\{ABCDEF\}$, $\backslash\mathrm{vect}\{e\}_{rot}$ ou $\backslash\mathrm{vect}\{e_{rot}\}$	\overrightarrow{ABCDEF} , \vec{e}_{rot} ou $\overrightarrow{e_{rot}}$
--	---

Exemple 2

$\backslash\mathrm{vect}\{i\}$ ou $\backslash\mathrm{vect}\{j\}_2$	\vec{i} ou \vec{j}_2
---	--------------------------

8.2.2 Norme

Ci-dessous l'argument optionnel de `\vnorm` vaut `b` par défaut pour `b`-ig soit « *gros* » en anglais mais l'on peut aussi utiliser `s` pour `s`-mall soit « *petit* ». Par contre `\vnorm` n'a pas d'option.

<code>\norm{\vect{i}} = \vnorm{i}</code>	$\ \vec{i}\ = \ \vec{i}\ $
<code>\norm{\dfrac{2}{7} \vect*{e}{k}} = \norm[s]{\dfrac{2}{7} \vect*{e}{k}}</code>	$\left\ \frac{2}{7} \vec{e}_k \right\ = \left\ \frac{2}{7} \vec{e}_k \right\ $

Remarque. Le code L^AT_EX pour des doubles barres extensibles ou non vient directement de ce message : <https://tex.stackexchange.com/a/43009/6880>.

8.2.3 Produit scalaire

Exemple 1 – Version longue

<code>\dotprod{\dfrac{1}{2} \vect{u}}{\vect{v}}</code>	$\frac{1}{2} \vec{u} \cdot \vec{v}$
--	-------------------------------------

Exemple 2 – Écriture « à la physicienne »

Dans l'exemple suivant, l'option `r` est pour `r`-after soit « *chevron* » en anglais, et dans `sr` le `s` est pour `s`-mall soit « *petit* » en anglais. Les physiciens aiment bien cette notation.

<code>\dotprod[r]{\dfrac{1}{2} \vect{u}}{\vect{v}}</code>	$\left\langle \frac{1}{2} \vec{u} \mid \vec{v} \right\rangle$
<code>\dotprod[sr]{\dfrac{1}{2} \vect{u}}{\vect{v}}</code>	$\langle \frac{1}{2} \vec{u} \mid \vec{v} \rangle$

Exemple 3 – Version courtes mais restrictive

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

<code>\vdotprod{u}{v}</code>	$\vec{u} \cdot \vec{v}$
<code>\vdotprod[r]{u}{v}</code> comme <code>\vdotprod[sr]{u}{v}</code>	$\langle \vec{u} \mid \vec{v} \rangle$ comme $\langle \vec{u} \mid \vec{v} \rangle$

8.2.4 Produit vectoriel

Exemple 1 - Écriture symbolique - Version longue

<code>\crossprod{\dfrac{1}{2} \vect{i}}{\vect{j}}</code>	$\frac{1}{2} \vec{i} \wedge \vec{j}$
--	--------------------------------------

Exemple 2 - Écriture symbolique - Version courte mais restrictive

<code>\vcrossprod{i}{j}</code>	$\vec{i} \wedge \vec{j}$
--------------------------------	--------------------------

Exemple 3 - Explication des calculs

Dans l'exemple suivant, le préfixe `calc` est pour `calc-uler` et `v` pour `v-ecteur`.

<pre style="color: #c00000;">\$\calccrossprod {\vect{u}}{x }{y }{z }% {\vect{v}}{x'}{y'}{z'} = \vcalscrossprod{u }{x }{y }{z }% {v }{x'}{y'}{z'}\$ \$\vcalscrossprod{AB}{x_B - x_A}% {y_B - y_A}% {z_B - z_A}% {CD}{x_D - x_C}% {y_D - y_C}% {z_D - z_C}\$</pre>	<div style="text-align: center;"> $\begin{array}{c} \vec{u} \quad \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \\ z & z' \\ x & x' \end{array} \right \\ \overline{AB} \end{array} = \begin{array}{c} \vec{u} \quad \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \\ z & z' \\ x & x' \end{array} \right \\ \overline{CD} \end{array}$ </div>
---	---

Avec un public averti on peut juste proposer les coordonnées sans les décorations comme ci-après via la version étoilée de `\vcalscrossprod` mais ceci fonctionne aussi avec `\calccrossprod`.

<pre style="color: #c00000;">\$\vcalscrossprod*{u}{x }{y }{z }% {v}{x'}{y'}{z'}\$</pre>	<div style="text-align: center;"> $\begin{array}{c} \vec{u} \quad \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \\ z & z' \\ x & x' \end{array} \right \end{array}$ </div>
---	---

Enfin si les vecteurs vous gênent il suffira d'utiliser l'option `novec` pour `no vec-tor` soit « *pas de vecteur* » en anglais comme ci-après. Ceci fonctionne aussi pour la macro `\calccrossprod`. Il peut sembler un peu lourd d'avoir des arguments pour des vecteurs non affichés mais ce choix permet à l'usage de faire des copier-coller redoutables d'efficacité!

<pre style="color: #c00000;">\$\vcalscrossprod[novec]{u}{x }{y }{z }% {v}{x'}{y'}{z'} = \vcalscrossprod*[novec]{u}{x }{y }{z }% {v}{x'}{y'}{z'}\$</pre>	<div style="text-align: center;"> $\left \begin{array}{cc} x & x' \\ y & y' \\ z & z' \\ x & x' \end{array} \right = \left \begin{array}{cc} x & x' \\ y & y' \\ z & z' \\ x & x' \end{array} \right$ </div>
---	---

Exemple 4 - Les coordonnées « détaillées »

Pour avoir le détail directement dans des coordonnées vous pouvez faire appel à la macro `\coordcrossprod` où le préfixe `coord` fait référence à `coord-onnée`⁹. On peut utiliser des options pour choisir certains paramètres de mise en forme.

9. En coulisse on utilise la macro `\coord` présentée dans la section 8.3 page 36.

<code>\$\coordcrossprod{\dfrac{1}{2}x}{y}{z}%</code> <code>{x'}</code> <code>{y'}{z'}\$</code>	$\left(y z' - z y' ; z x' - \frac{1}{2} x z' ; \frac{1}{2} x y' - y x' \right)$
<code>\$\coordcrossprod[vb]%</code> <code>{\dfrac{1}{2}x}{y}{z}%</code> <code>{x'}</code> <code>{y'}{z'}\$</code>	$\begin{bmatrix} y z' - z y' \\ z x' - \frac{1}{2} x z' \\ \frac{1}{2} x y' - y x' \end{bmatrix}$
<code>\$\coordcrossprod[sp,c]%</code> <code>{\dfrac{1}{2}x}{y}{z}%</code> <code>{x'}</code> <code>{y'}{z'}\$</code>	$(y \cdot z' - z \cdot y' ; z \cdot x' - \frac{1}{2} x \cdot z' ; \frac{1}{2} x \cdot y' - y \cdot x')$

Voici les options disponibles. Nous expliquons ensuite comment les utiliser.

1. **p** vient de **p**-arenthèses. Ceci donnera une écriture horizontale.
2. **b** vient de **b**-rackets soit « *crochets* » en anglais. Ceci donnera une écriture horizontale.
3. **sp** et **sb** produisent des délimiteurs non extensibles en mode horizontal. Ici **s** vient de **s**-mall soit « *petit* » en anglais.
4. **vp** et **vb** produisent des écritures verticales. Ici **v** vient de **v**-ertical.
5. **s** tout seul demande d'utiliser un espace pour séparer les facteurs de chaque produit.
6. **t** tout seul demande d'utiliser `\times` comme opérateur de multiplication.
7. **c** tout seul demande d'utiliser `\cdot` comme opérateur de multiplication.

On peut indiquer des options vis à vis du mode vertical ou horizontal avec des délimiteurs extensibles ou non éventuellement, ou bien sur le symbole pour les produits. On peut aussi combiner deux de ces types de choix en les séparant par une virgule ce qui fait un total de $6 \times 3 = 18$ combinaisons possibles. La valeur par défaut est **p,s**.

Attention ! Les produits sont produits stupidement. Autrement dit ce sera à vous d'ajouter des parenthèses là où il y en aura besoin sinon vous obtiendrez des horreurs comme celle ci-dessous.

<code>\$\coordcrossprod[vb]{x_B - x_A}%</code> <code>{y_B - y_A}%</code> <code>{z_B - z_A}%</code> <code>{x_D - x_C}%</code> <code>{y_D - y_C}%</code> <code>{z_D - z_C}\$</code>	$\begin{bmatrix} y_B - y_A z_D - z_C - z_B - z_A y_D - y_C \\ z_B - z_A x_D - x_C - x_B - x_A z_D - z_C \\ x_B - x_A y_D - y_C - y_B - y_A x_D - x_C \end{bmatrix}$
--	---

Ici nous n'avons pas d'autre choix que de corriger le tir nous-même. Ceci étant indiqué, ce genre de situation est très rare dans la vraie vie mathématique où l'on évite d'avoir à calculer un produit vectoriel avec des expressions compliquées.

<code>\$\coordcrossprod[vb]{(x_B - x_A)}%</code> <code>{(y_B - y_A)}%</code> <code>{(z_B - z_A)}%</code> <code>{(x_D - x_C)}%</code> <code>{(y_D - y_C)}%</code> <code>{(z_D - z_C)}\$</code>	$\begin{bmatrix} (y_B - y_A)(z_D - z_C) - (z_B - z_A)(y_D - y_C) \\ (z_B - z_A)(x_D - x_C) - (x_B - x_A)(z_D - z_C) \\ (x_B - x_A)(y_D - y_C) - (y_B - y_A)(x_D - x_C) \end{bmatrix}$
--	---

8.2.5 Plan – Déterminant de deux vecteurs

Exemple 1 – Version décorée

Dans l'exemple suivant, le préfixe **calc** est pour **calc**-uler.

$\begin{aligned} &\$ \backslash \text{calcdetplane} \{ \backslash \text{vect} \{ u \} \} \{ x \} \{ y \} \% \\ &\quad \{ \backslash \text{vect} \{ v \} \} \{ x' \} \{ y' \} \$ \\ \text{ou} \\ &\$ \backslash \text{calcdetplane} \{ \backslash \text{vect} \{ AB \} \} \% \\ &\quad \{ x_B - x_A \} \{ y_B - y_A \} \% \\ &\quad \{ \backslash \text{vect} \{ CD \} \} \% \\ &\quad \{ x_D - x_C \} \{ y_D - y_C \} \$ \end{aligned}$	$\begin{array}{c} \vec{u} \quad \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right \end{array} \quad \text{ou} \quad \begin{array}{c} \overrightarrow{AB} \quad \overrightarrow{CD} \\ \left \begin{array}{cc} x_B - x_A & x_D - x_C \\ y_B - y_A & y_D - y_C \end{array} \right \end{array}$
---	---

Exemple 2 – Version non décorée

$\begin{aligned} &\$ \backslash \text{calcdetplane} * \{ \backslash \text{vect} \{ u \} \} \{ x \} \{ y \} \% \\ &\quad \{ \backslash \text{vect} \{ v \} \} \{ x' \} \{ y' \} \$ \end{aligned}$	$\begin{array}{c} \vec{u} \quad \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right \end{array}$
--	---

Exemple 3 – Rédaction raccourcie pour les vecteurs

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

$\begin{aligned} &\$ \backslash \text{vcalcdetplane} \{ u \} \{ x \} \{ y \} \% \\ &\quad \{ v \} \{ x' \} \{ y' \} \\ = \\ &\backslash \text{vcalcdetplane} * \{ u \} \{ x \} \{ y \} \% \\ &\quad \{ v \} \{ x' \} \{ y' \} \$ \end{aligned}$	$\begin{array}{c} \vec{u} \quad \vec{v} \quad \vec{u} \quad \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right = \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right \end{array}$
---	---

Exemple 4 – Versions sans les vecteurs

Dans l'exemple suivant, on utilise la valeur `novec` pour l'argument optionnel de `\vcalcdetplane` qui par défaut est `vec` pour pour `vec`-teur. À l'usage ceci permet des copier-coller très efficaces !

$\begin{aligned} &\$ \backslash \text{vcalcdetplane} [\text{novec}] \{ u \} \{ x \} \{ y \} \% \\ &\quad \{ v \} \{ x' \} \{ y' \} \\ = &\backslash \text{vcalcdetplane} * [\text{novec}] \{ u \} \{ x \} \{ y \} \% \\ &\quad \{ v \} \{ x' \} \{ y' \} \$ \end{aligned}$	$\left \begin{array}{cc} x & x' \\ y & y' \end{array} \right = \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right $
--	---

Remarque. Ce qui précède marche aussi avec les macros `\calcdetplane` et `\calcdetplane*`.

Exemple 5 – Calcul développé

Grâce à l'argument optionnel de `\calcdetplane` ou `\vcalcdetplane`, il est aussi possible d'obtenir le résultat développé du calcul comme ci-après où `exp` est pour `exp`-and soit « *développer* » en anglais, `c` pour `\cdot` et enfin `t` pour `\times`. Même si les vecteurs ne sont pas utilisés pour la mise en forme, on obtient ici une méthode très pratique à l'usage car permettant de faire des copier-coller.

<code>\vcalcdetplane[exp]{u}{x }{y }%</code> <code>{v}{x'}{y'}\$</code>	$x y' - y x'$
<code>\vcalcdetplane[cexp]{u}{x }{y }%</code> <code>{v}{x'}{y'}\$</code>	$x \cdot y' - y \cdot x'$
<code>\vcalcdetplane[texp]{u}{x }{y }%</code> <code>{v}{x'}{y'}\$</code>	$x \times y' - y \times x'$

Remarque. Ce qui précède marche aussi avec les versions étoilées.

Attention ! Le développement effectué est stupide. Autrement dit ce sera à vous d'ajouter des parenthèses là où il y en aura besoin sinon vous obtiendrez des horreurs comme celle qui suit.

<code>\vcalcdetplane[exp]{AB}%</code> <code>{x_B - x_A}%</code> <code>{y_B - y_A}%</code> <code>{CD}%</code> <code>{x_D - x_C}%</code> <code>{y_D - y_C}\$</code>	$x_B - x_A y_D - y_C - y_B - y_A x_D - x_C$
--	---

Ici nous n'avons pas d'autre choix que de régler le problème à la amin. Ce genre de situation n'est pas rare dans la vraie vie mathématique.

<code>\vcalcdetplane[exp]{AB}%</code> <code>{(x_B - x_A)}%</code> <code>{(y_B - y_A)}%</code> <code>{CD}%</code> <code>{(x_D - x_C)}%</code> <code>{(y_D - y_C)}\$</code>	$(x_B - x_A)(y_D - y_C) - (y_B - y_A)(x_D - x_C)$
--	---

8.3 Coordonnées

Exemple 1 – Des coordonnées seules

lymath propose, via un argument optionnel, six façons différentes de rédiger des coordonnées seules (*nous verrons après des macros pour les coordonnées d'un point et celles d'un vecteur afin de produire un code $L^A T_E X$ plus sémantique*). Commençons par les écritures horizontales où vous noterez l'utilisation de | pour séparer les coordonnées dont le nombre peut être quelconque.

<code>\coord {\dfrac{1}{3} -4 0}\$</code> ou <code>\coord[sp]{\dfrac{1}{3} -4 0}\$</code>	$\left(\frac{1}{3}; -4; 0\right)$ ou $(\frac{1}{3}; -4; 0)$
<code>\coord[b] {\dfrac{1}{3} -4 0}\$</code> ou <code>\coord[sb]{\dfrac{1}{3} -4 0}\$</code>	$\left[\frac{1}{3}; -4; 0\right]$ ou $[\frac{1}{3}; -4; 0]$

Il existe en plus deux versions verticales.

<code>\coord[vp]{3 -4}\$</code> ou <code>\coord[vb]{3 -4}\$</code>	$\begin{pmatrix} 3 \\ -4 \end{pmatrix}$ ou $\begin{bmatrix} 3 \\ -4 \end{bmatrix}$
---	--

Voici d'où viennent les noms des options.

1. **p**, qui est aussi la valeur par défaut, vient de **p**-arenthèses.
2. **b** vient de **b**-rackets soit « *crochets* » en anglais.
3. **s** pour **s**-mall soit « *petit* » en anglais permet d’avoir des délimiteurs non extensibles en mode horizontal car par défaut ils le sont.
4. **v** pour **v**-ertical demande de produire une écriture verticale.

Exemple 2 – Coordonnées d’un point

La macro `\pcoord` avec **p** pour **p**-oint prend un argument supplémentaire avant les coordonnées qui est le nom d’un point qui sera mis en forme par la macro `\pt`. Si vous ne souhaitez pas que `\pt` soit appliquée, il suffit de passer via la version étoilée `\pcoord*`.

<code>\$\pcoord{A}{3 -4 0 -1}\$ ou <code>\$\pcoord*{\Sigma}{7 9 8}\$</code> </code>	$A(3; -4; 0; -1)$ ou $\Sigma(7; 9; 8)$
--	--

Toutes les options disponibles avec `\coord` le sont aussi avec `\pcoord`.

<code>\$\pcoord[b]{A}{3 -4 0 -1}\$ ou <code>\$\pcoord*[b]{\Sigma}{7 9 8}\$</code> </code>	$A[3; -4; 0; -1]$ ou $\Sigma[7; 9; 8]$
--	--

Exemple 3 – Coordonnées d’un vecteur

Le fonctionnement de `\vcoord` est similaire à celui de `\pcoord` si ce n’est que c’est la macro `\vect` qui sera appliquée si besoin.

<code>\$\vcoord{u}{3 -4}\$ ou <code>\$\vcoord*{\dfrac{1}{2} \vect{u}}{3 -4}\$</code> </code>	$\vec{u}(3; -4)$ ou $\frac{1}{2}\vec{u}(3; -4)$
<code>\$\vcoord[vp]{u}{3 -4}\$ ou <code>\$\vcoord*[vp]{\dfrac{1}{2} \vect{u}}{3 -4}\$</code> </code>	$\vec{u}\left(\begin{smallmatrix} 3 \\ -4 \end{smallmatrix}\right)$ ou $\frac{1}{2}\vec{u}\left(\begin{smallmatrix} 3 \\ -4 \end{smallmatrix}\right)$

8.4 Nommer un repère

Exemple 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*).

<code>\$\axes{\pt{O} % \pt{I} \pt{J}}\$</code>	$(O; I, J)$
---	-------------

Exemple 2 – La méthode basique en version étoilée

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

```

 $\backslash axes{\backslash pt{0} \%$ 
 $\backslash \dfrac{7}{3} \backslash vect{i} \%$ 
 $\backslash \backslash vect{j}}{\backslash pt{0} \%}$ 
ou
 $\backslash axes*{\backslash pt{0} \%$ 
 $\backslash \dfrac{7}{3} \backslash vect{i} \%$ 
 $\backslash \backslash vect{j}}{\backslash pt{0} \%}$ 

```

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

Exemple 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 $n > 0$.

```

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

```

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

Exemple 4 – Repère affine

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

```

 $\backslash paxes{0 | I | J | K} \$$ 
au lieu de
 $\backslash axes{\backslash pt{0} \%$ 
 $\backslash \backslash pt{I} | \backslash \backslash pt{J} | \backslash \backslash pt{K}}{\backslash pt{0} \%}$ 

```

$(O; I, J, K)$ au lieu de $(O; I, J, K)$

Exemple 5 – Repère vectoriel (méthode 1)

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

```

 $\backslash vaxes{\backslash pt{0} | i | j} \$$ 
au lieu de
 $\backslash axes{\backslash pt{0} | \backslash vect{i} | \backslash vect{j}}{\backslash pt{0} \%}$ 

```

$(O; \vec{i}, \vec{j})$ au lieu de $(O; \vec{i}, \vec{j})$

Exemple 6 – Repère vectoriel (méthode 2)

Dans l'exemple suivant, le préfixe pv permet de combiner ensemble les fonctionnalités proposées par les préfixes p et v.

```

 $\backslash pvaxes{0 | i | j} \$$ 
au lieu de
 $\backslash axes{\backslash pt{0} | \backslash vect{i} | \backslash vect{j}}{\backslash pt{0} \%}$ 

```

$(O; \vec{i}, \vec{j})$ au lieu de $(O; \vec{i}, \vec{j})$

8.5 Arcs circulaires

Exemple 1

\circ{ABCDEF} , $\circ*{A}{rot}$ ou $\circ{A_{rot}}$	\widehat{ABCDEF} , \hat{A}_{rot} ou $\widehat{A_{rot}}$
--	---

Exemple 2

\circ{i} ou $\circ*{j}{2}$	\hat{i} ou \hat{j}_2
---------------------------------	--------------------------

8.6 Angles

8.6.1 Angles géométriques « intérieurs »

Exemple 1

\anglein{ABCDEF} $\anglein*{A}{rot}$ $\anglein{A_{rot}}$	\widehat{ABCDEF} \hat{A}_{rot} A_{rot}
--	--

Exemple 2 – Cacher les points du i et du j

\anglein{i} et $\anglein*{j}{2}$	\hat{i} et \hat{j}_2
---------------------------------------	--------------------------

8.6.2 Angles orientés de vecteurs

Sans chapeau - Version longue

L'option par défaut est p pour p-arenthèse. Dans sp le s est pour s-mall soit « *petit* » en anglais.

$\angleorient{\dfrac{1}{2} \vect{i}}{\vect{j}}$ $\angleorient[sp]{\dfrac{1}{2} \vect{i}}{\vect{j}}$	$\left(\frac{1}{2} \vec{i} ; \vec{j} \right)$ $\left(\frac{1}{2} \vec{i} ; \vec{j} \right)$
--	--

Sans chapeau - Version courte mais restrictive

Dans l'exemple suivant, le préfixe v est pour v-ecteur qui permet de simplifier la saisie quand l'on a juste des vecteurs nommés avec des lettres (*notez que l'option sp n'apporte rien de nouveau*).

$\vangleorient{i}{j}$ comme $\vangleorient[sp]{i}{j}$	$(\vec{i} ; \vec{j})$ comme $(\vec{i} ; \vec{j})$
--	---

Avec un chapeau

Dans l'exemple suivant, **h** est pour **h**-at soit « *chapeau* » en anglais. Notez au passage que **sh** produit juste des parenthèses petites mais ce choix de nom simplifie l'utilisation de la macro (*c'est mieux que hsp par exemple*).

<code>\angleorient[h] {\dfrac{1}{2} \vect{i}}% {\vect{j}}\$</code>	$\widehat{\left(\frac{1}{2}\vec{i}; \vec{j}\right)}$
<code>\angleorient[sh]{\dfrac{1}{2} \vect{i}}% {\vect{j}}\$</code>	$\overline{\left(\frac{1}{2}\vec{i}; \vec{j}\right)}$
<code>\vangleorient[h] {i}{j}\$ comme \vangleorient[sh]{i}{j}\$</code>	$\overline{(\vec{i}; \vec{j})} \text{ comme } \widehat{(\vec{i}; \vec{j})}$

9 Analyse

9.1 Constantes et paramètres

9.1.1 Constantes classiques

La liste complète

<code>\ggamma\$, \ppi\$, \tttau\$, \ee\$, \ii\$, \jj\$ et \kk\$ où \tttau = 2 \ppi\$</code>	$\gamma, \pi, \tau, e, i, j \text{ et } k \text{ où } \tau = 2\pi$
--	--

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.

9.1.2 Constantes latines personnelles

Exemple

La macro `\param` est surtout là pour une utilisation pédagogique.

<code>\param{a} x^2 + \param{b} x + \param{c}\$ ou \$a x^2 + b x + c\$</code>	$ax^2 + bx + c \text{ ou } ax^2 + bx + c$
---	---

9.2 La fonction valeur absolue

Exemple

<code>\abs{2}\$, \abs {\dfrac{3}{5}}\$ ou \abs*{\dfrac{3}{5}}\$</code>	$ 2 , \left \frac{3}{5}\right \text{ ou } \left \frac{3}{5}\right $
---	--

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

9.3 Fonctions nommées spéciales

9.3.1 Sans paramètre

Exemple

Quelques fonctions nommées supplémentaires où `fch` est pour `f-renc` soit « *français* » en anglais (*ce choix a été fait pour éviter des incompatibilités avec quelques autres packages*). La liste complète des fonctions nommées est donnée un peu plus bas dans une section dédiée.

$\$ \backslash fch x \backslash neq ch x$,$ $\$ \backslash ppcm(x;y)$$ ou $\$ \backslash lg x$$	$ch x \neq chx$, $ppcm(x;y)$ ou $lg x$
--	---

9.3.2 Avec un paramètre

Exemple

$\$ \backslash logb\{2\} x = \lg x$$ ou $\$ \backslash expb\{6\} y = 6^y$$	$\log_2 x = \lg x$ ou $\exp_6 y = 6^y$
---	--

9.3.3 Toutes les fonctions nommées en plus

<code>pgcd</code> : <code>pgcd...</code>	<code>atanh</code> : <code>atanh...</code>
<code>ppcm</code> : <code>ppcm...</code>	<code>fch</code> : <code>ch...</code>
<code>acos</code> : <code>acos...</code>	<code>fsh</code> : <code>sh...</code>
<code>asin</code> : <code>asin...</code>	<code>fth</code> : <code>th...</code>
<code>atan</code> : <code>atan...</code>	<code>afch</code> : <code>ach...</code>
<code>arccosh</code> : <code>arccosh...</code>	<code>afsh</code> : <code>ash...</code>
<code>arcsinh</code> : <code>arcsinh...</code>	<code>afth</code> : <code>ath...</code>
<code>arctanh</code> : <code>arctanh...</code>	<code>expb{p}</code> : <code>exp_p...</code>
<code>acosh</code> : <code>acosh...</code>	<code>logb{p}</code> : <code>log_p...</code>
<code>asinh</code> : <code>asinh...</code>	

9.4 Des notations complémentaires pour des suites spéciales

Exemple

$\$ \backslash seqplus\{F\}\{1\}\{2\}$$ $\$ \backslash seqhypergeo\{F\}\{1\}\{2\}$$ $\$ \backslash seqsuprageo\{F\}\{1\}\{2\}\{3\}\{4\}$$ <code>pour les fous\dots :-)</code>	F_1^2 ${}_1F_2$ ${}_1F_2^3$ pour les fous... :-)
--	--

9.5 Calcul différentiel

9.5.1 Les opérateurs ∂ et d

Exemple 1

$\$\\dd{t} = \\dd{1}{t}$ ou \$\\dd{n}{x}$$	$dt = d^1t$ ou d^nx
$\$\\pp{t} = \\pp{1}{t}$ ou \$\\pp{n}{x}$$	$\partial t = \partial^1t$ ou ∂^nx

9.5.2 Dérivations totales d'une fonction – Version longue mais polymorphe

Exemple 1 – Différentes écritures possibles

La macro `\der` est stricte du point de vue sémantique car on doit lui fournir la fonction, l'ordre de dérivation et la variable de dérivation (*voir la section 9.5.3 qui présente la macro `\sder` permettant une rédaction efficace pour obtenir $f^{(1)}$ ou f'*). Voici plusieurs mises en forme faciles à taper via l'option de `\der`. Attention bien entendu à n'utiliser l'option par défaut `u` qu'avec un ordre de dérivation de valeur naturelle connue !

$\$\\der{f}{3}{x}$ $= \\der{e}{f}{3}{x}$	$f''' = f^{(3)}$
$\$\\der{i}{u}{k}{x}$ $= \\der{f}{u}{k}{x}$ $= \\der{sf}{u}{k}{x}$	$d_x^k u = \frac{d^k u}{dx^k} = \frac{d^k u}{dx^k}$

On peut aussi ajouter autour de la fonction des parenthèses extensibles ou non. Ci-dessous on montre aussi une écriture du type « *opérateur fonctionnel* ».

$\$\\der{osf,sp}{\frac{1}{2} uv}{k}{x}$ $= \\der{of,p}{\dfrac{1}{2} uv}{k}{x}$	$\frac{d^k}{dx^k}(\frac{1}{2} uv) = \frac{d^k}{dx^k} \left(\frac{1}{2} uv \right)$
---	---

Remarque. Expliquons les valeurs des options.

1. `u`, la valeur par défaut, est pour `u`-suel soit l'écriture avec les primes. Cette option ne marchera pas avec un nombre symbolique de dérivations.
2. `e` est pour `e`-xposant.
3. `i` est pour `i`-ndice.
4. `f` est pour `f`-raction avec aussi `sf` pour une écriture réduite où `s` est pour `s`-mall soit « *petit* » en anglais.
5. `of` et `osf` utilisent le préfixe `o` pour `o`-pérateur.
6. `p` est pour `p`-arenthèse : dans ce cas les parenthèses seront extensibles.
7. `sp` est pour des parenthèses non extensibles.

Exemple 2 – Pas de uns inutiles

$\$\\der{i}{u}{1}{x}$ $= \\der{f}{u}{1}{x}$ $= \\der{sf}{u}{1}{x}$ $= \\der{of}{u}{1}{x}$	$d_x u = \frac{du}{dx} = \frac{du}{dx} = \frac{d}{dx} u$
--	--

Remarque. Voici comment forcer les exposants 1 si besoin.

```
\der[i]{u}{\,\!1}{x}
= \der[f]{u}{\,\!1}{x}
= \der[sf]{u}{\,\!1}{x}
= \der[of]{u}{\,\!1}{x}
```

$$d_x^1 u = \frac{d^1 u}{dx^1} = \frac{d^1 u}{dx^1} = \frac{d^1}{dx^1} u$$

9.5.3 Dérivations totales d'une fonction – Version courte pour les écritures standard

Dans l'exemple suivant le code manque de sémantique car on n'indique pas la variable de dérivation. Ceci étant dit à l'usage la macro `\sder` rend de grands services. Ici le préfixe **s** est pour **s**-imple voire **s**-impliste... Voici des exemples où de nouveau l'option par défaut **u** ne sera fonctionnelle qu'avec un ordre de dérivation de valeur naturelle connue !

```
\sder{f}{1} = \der{f}{1}{x}$

\sder{f}{1}
= \sder[e]{f}{1}$

\sder[sp]{\dfrac{1}{2} uv}{2}
= \sder[e,p]{\dfrac{1}{2} uv}{2}$
```

$$\begin{aligned} f' &= f' \\ f' &= f^{(1)} \\ \left(\frac{1}{2}uv\right)'' &= \left(\frac{1}{2}uv\right)^{(2)} \end{aligned}$$

Remarque. Ici les seules options disponibles sont **u**, **e**, **p** et **sp**.

9.5.4 L'opérateur de dérivation totale

Ce qui suit peut rendre service au niveau universitaire. Les options possibles sont **f**, valeur par défaut, **sf** et **i** avec les mêmes significations que pour la macro `\der`.

```
\derope {k}{x}
= \derope[sf]{k}{x}
= \derope[i]{k}{x}$

\sderope {1}{x}
= \derope[sf]{1}{x}
= \derope[i]{1}{x}$
```

$$\begin{aligned} \frac{d^k}{dx^k} &= \frac{d^k}{dx^k} = d_x^k \\ \frac{d}{dx} &= \frac{d}{dx} = d_x \end{aligned}$$

9.5.5 Dérivations partielles

Exemple 1 – Différentes écritures possibles

La macro `\pder`¹⁰ avec **p** pour **p**-artielle permet de rédiger des dérivées partielles en utilisant facilement plusieurs mises en forme via une option qui vaut **f** par défaut. Cette macro attend une fonction, les dérivées partielles effectuées et l'ordre total de dérivation. Voici les deux types de mise en forme où vous noterez comment `x | y^2` est interprété.

```
\pder {f}{x | y^2}{3}
= \pder[sf]{f}{x | y^2}{3}$
ou
\sder[i]{f}{x | y^2}{3}$
```

$$\frac{\partial^3 f}{\partial x \partial y^2} = \frac{\partial^3 f}{\partial x \partial y^2} \text{ ou } \partial_{xy(2)}^3 f$$

10. `\partial` existe déjà pour obtenir ∂ .

On peut aussi ajouter autour de la fonction des parenthèses extensibles ou non. Ci-dessous on montre aussi une écriture du type « *opérateur fonctionnel* ».

<pre> \pder[of] {f}{x y^2}{ } = \pder[osf]{f}{x y^2}{ }\$ ou \pder[i,sp]{u + v}{x y^2}{ }\$ </pre>	$\frac{\partial}{\partial x \partial y^2} f = \frac{\partial}{\partial x \partial y^2} f \text{ ou } \partial_{x y(2)}(u + v)$
--	--

Remarque. Les options disponibles sont **f**, **sf**, **of**, **osf**, **i**, **p** et **sp** avec des significations similaires à celles pour la macro `\der`.

Exemple 2 – Pas de uns inutiles

<pre> \pder {u}{x}{1} = \pder[sf]{u}{x}{1} = \pder[i] {u}{x}{1}\$ </pre>	$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial x} = \partial_x u$
--	--

9.5.6 L’opérateur de dérivation partielle

Ce qui suit peut rendre service au niveau universitaire. Les options possibles sont **f**, valeur par défaut, **sf** et **i** avec les mêmes significations que pour la macro `\der`.

<pre> \pderope {x y^2}{3} = \pderope[sf]{x y^2}{3} = \pderope[i] {x y^2}{3}\$ </pre>	$\frac{\partial^3}{\partial x \partial y^2} = \frac{\partial^3}{\partial x \partial y^2} = \partial_{x y(2)}^3$
--	---

9.6 Calcul intégral

9.6.1 Intégrales multiples

Commençons par un point important : le package réduit les espacements entres des symboles \int successifs. Voici un exemple.

<pre> \displaystyle \int \int \int F(x;y;z) \dd{x} \dd{y} \dd{z}\$ \displaystyle \int_{a}^{b} \int_{c}^{d} \int_{e}^{f} F(x;y;z) \dd{x} \dd{y} \dd{z}\$ </pre>	$\int \int \int F(x; y; z) \, dx \, dy \, dz$ $\int_a^b \int_c^d \int_e^f F(x; y; z) \, dx \, dy \, dz$
--	---

Remarque. Par défaut, L^AT_EX affiche $\int \int \int F(x; y; z) \, dx \, dy \, dz$ et $\int_a^b \int_c^d \int_e^f F(x; y; z) \, dx \, dy \, dz$. Nous avons obtenu ce résultat en utilisant `\stdint` qui est l’opérateur proposé de façon standard par L^AT_EX.

9.6.2 Un opérateur d'intégration clés en main

Exemple 1 – À quoi bon ?

Le 1^{er} exemple qui suit semblera être une hérésie pour les habitués de L^AT_EX mais rappelons que le but de `lymath` est de rendre les documents facilement modifiables globalement ou localement comme le montre le 2^e exemple.

<pre> <math displaystyle<="" math=""> \integrate{a}{b}{f(x)}{x} = \int_{x=a}^{x=b} f(x) \, \mathrm{d}x\$ <math displaystyle<="" math=""> \integrate*{a}{b}{f(x)}{x} \eqdef \integrate{a}{b}{f(x)}{x}\$ </math></math></pre>	$\int_{x=a}^{x=b} f(x) \, \mathrm{d}x = \int_{x=a}^{x=b} f(x) \, \mathrm{d}x$ $\int_a^b f(x) \, \mathrm{d}x \stackrel{\text{déf}}{=} \int_{x=a}^{x=b} f(x) \, \mathrm{d}x$
--	--

Exemple 2 – Le mode `displaystyle`

La macro `\dintegrate*` présentée ci-dessous possède aussi une version non étoilée `\dintegrate`.

<pre> $\dintegrate*{a}{b}{f(x)}{x}$ = \integrate*{a}{b}{f(x)}{x}\$ </pre>	$\int_a^b f(x) \, \mathrm{d}x = \int_a^b f(x) \, \mathrm{d}x$
--	---

9.6.3 L'opérateur crochet

Exemple 1

<pre> $\hook{a}{b}{F(x)}{x}$ \eqdef F(b) - F(a)\$ $\dintegrate*{a}{b}{f(x)}{x}$ = \hook*{a}{b}{F(x)}{x}\$ </pre>	$\left[F(x) \right]_{x=a}^{x=b} \stackrel{\text{déf}}{=} F(b) - F(a)$ $\int_a^b f(x) \, \mathrm{d}x = \left[F(x) \right]_a^b$
---	---

Remarque. Il faut savoir que `\hook` signifie « *crochet* » en anglais mais la bonne traduction du terme mathématique est en fait « *square bracket* ». Ceci étant dit l'auteur de `lymath` trouve plus efficace d'utiliser `\hook` comme nom de macro.

Exemple 2 – Des crochets non extensibles

Dans l'exemple suivant, on utilise l'option `sb` pour **s**-mall **b**-rackets soit « *petits crochets* » en anglais. Les options sont disponibles à la fois pour `\hook` et `\hook*`.

<pre> $\hook*{a}{b}\%$ {\dfrac{x - 1}{5 + x^2}}{x} = \hook*[sb]\% {a}{b}\% {\dfrac{x - 1}{5 + x^2}}{x}\$ </pre>	$\left[\frac{x - 1}{5 + x^2} \right]_a^b = \left[\frac{x - 1}{5 + x^2} \right]_a^b$
--	---

Exemple 3 – Un trait vertical épuré

Via les options `r` et `sr` pour `s-mall` et `r-ull` soit « *petit* » et « *trait* » en anglais, on obtient ce qui suit.

```
\hook[r] {a}{b}%
      {\dfrac{x - 1}{5 + x^2}}{x}
= \hook[sr]{a}{b}%
      {\dfrac{x - 1}{5 + x^2}}{x}$
```

$$\frac{x - 1}{5 + x^2} \Big|_{x=a}^{x=b} = \frac{x - 1}{5 + x^2} \Big|_{x=a}^{x=b}$$

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 via le réglage `\tkzTabSetup[arrowstyle = triangle 60]`.

Nous donnons quelques exemples classiques d’utilisation (*les codes ont été mis en forme pour faciliter la compréhension de la syntaxe à suivre*). Si besoin reportez vous à la documentation de `tkz-tab` pour obtenir des compléments d’information.

Exemple 1 – Avec des 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	−

Exemple 2 – Avec des variations

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

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

Exemple 3 – Variations via 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

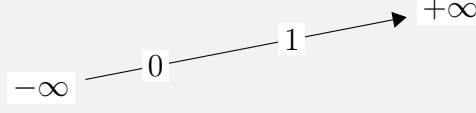
Exemple 4 – Une image intermédiaire avec une seule flèche

```
\begin{tikzpicture}
  \tkzTabInit{
    $x$ / 1 ,
    $3 x^2$ / 1 ,
    $x^3$ / 1.5
  }{
    $-\infty$ , $0$ , $+\infty$
  }
  \tkzTabLine{ , + , 0 , + , }
  \tkzTabVar {-/ $-\infty$ , R , +/ $+\infty$}
  %
  \tkzTabIma{1}{3}{2} % Position entre 1ière et 3ième valeur puis rang entier relatif.
                     % Valeur de l'image.
\end{tikzpicture}
```

x	$-\infty$	0	$+\infty$
$3x^2$	$+$	0	$+$
x^3	$-\infty$	0	$+\infty$

Exemple 5 – Valeurs intermédiaires ou interdites

```
\begin{tikzpicture}
  \tkzTabInit[espc1 = 6]{ % Largeur entre les valeurs du tableau.
    $x$ /1 ,
    $\dfrac{1}{x}$ /1.25 ,
    $\ln$ /1.75
  }{
    $0$ , $+\infty$
  }
  \tkzTabLine{d , + , }
  \tkzTabVar {D- / $-\infty$ , + / $+\infty$}
  %
  \tkzTabVal{1}{2}{0.35} % Position entre 1ière et 2ième valeur puis en proportion.
    {1}{0} % x_1 et f(x_1)
  \tkzTabVal{1}{2}{0.65} % Position entre 1ière et 2ième valeur puis en proportion.
    {$\infty$}{1} % x_2 et f(x_2)
\end{tikzpicture}
```

x	0	1	e	$+\infty$
$\frac{1}{x}$		+		
ln				

Voici un autre exemple pour comprendre comment utiliser `\tkzTabVal` avec en plus l'option `draw` qui peut rendre service.

```

\begin{tikzpicture}
  \tkzTabInit[espc1 = 4]{
    $x$ / 1 ,
    $f'(x)$ / 1 ,
    $f(x)$ / 1.5
  }{
    $0$ , $e$ , $+\infty$
  }
  \tkzTabLine{d , + , 0 , - , }
  \tkzTabVar {D- / $-\infty$ , + / $\ee$ , - / $0$ }
  %
  \tkzTabVal[draw]{1}{2}{0.5} % Position entre 1ière et 2ième valeur au milieu.
    {$1$}{$\dfrac{1}{\ee}$}
  \tkzTabVal[draw]{2}{3}{0.5} % Position entre 2ième et 3ième valeur au milieu.
    {$\ee^2$}{$1$}
\end{tikzpicture}

```

x	0	1	e	e^2	$+\infty$
$f'(x)$		+	0	-	
$f(x)$	$-\infty$	$\frac{1}{e}$	e	1	0

9.8 Comparaison asymptotique de suites et de fonctions

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

Exemple 1

Les notations suivantes sont dues à Landau.

\mathcal{O} ou \mathcal{o}

\mathcal{O} ou \mathcal{o}

Exemple 2

$\mathcal{O}(x) \neq \mathcal{o}(x)$ ou
 $e^{t + \mathcal{O}(t)} = e^{\mathcal{O}(t)}$

$\mathcal{O}(x) \neq \mathcal{o}(x)$ ou $e^{t+\mathcal{O}(t)} = e^{\mathcal{O}(t)}$

9.8.2 La notation Ω

Exemple 1

La notation suivante est due à Hardy et Littlewood.

Ω

Ω

Exemple 2

Dans l'exemple suivant, $f(n) = \Omega(g(n))$ signifie : $\exists(m, n_0)$ tel que $n \geq n_0$ implique $f(n) \geq mg(n)$.

$f(n) = \Omega(g(n))$
 $f(n) = \Omega(g(n))$

9.8.3 La notation Θ

Exemple 1

 Θ
 Θ

Exemple 2

Dans l'exemple suivant, $f(n) = \Theta(g(n))$ signifie : $\exists(m, M, n_0)$ tel que $mg(n) \leq f(n) \leq Mg(n)$ dès que $n \geq n_0$.

 $f(n) = \Theta(g(n))$
 $f(n) = \Theta(g(n))$

10 Probabilité

10.1 Probabilité « simple »

Exemple 1

 $p(A)$
 $p(A)$

Exemple 2 – Choisir le nom de la probabilité

 $P(A)$
 $P(A)$

10.2 Probabilité conditionnelle

Exemple 1 – Les deux écritures classiques

La 1^{re} notation, qui est devenue standard, permet de comprendre l'ordre des arguments.

 $p_B(A) = p(A | B)$
 $p_B(A) = p(A | B)$

Exemple 2 – Obtenir la formule de définition

Le suffixe **exp** est pour **exp-and** soit « *développer* » en anglais¹¹.

 $\frac{p(A \cap B)}{p(B)} = \frac{p(A|B)}{p(B)}$
 $\frac{p(A \cap B)}{p(B)} = \frac{p(A|B)}{p(B)}$

11. Pour ne pas alourdir l'utilisation de `\probacond`, il a été choisi d'utiliser un suffixe au lieu d'un système de multi-options.

Exemple 3 – Choisir le nom de la probabilité

```
\probacond [P]{B}{A}
= \probacond* [P]{B}{A}
= \probacondexp*[P]{B}{A}
= \probacondexp [P]{B}{A}$
```

$$P_B(A) = P(A | B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

10.3 Espérance

Exemple

expval vient de *exp*-ected *val*-ue soit « *espérance* » en anglais.

```
$\expval{X}$
```

$E(X)$

Choisir le nom de l'espérance

```
$\expval[E_1]{X}$
```

$E_1(X)$

10.4 Arbres pondérés

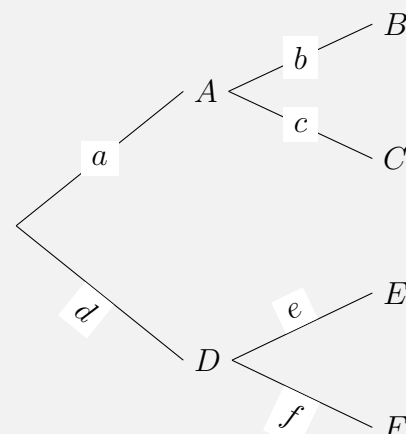
Que se passe-t-il en coulisse ?

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

Exemple 1 – Le cas 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. À cela s'ajoutent les styles spéciaux **pweight**, **apweight** et **bpweight** qui facilitent l'écriture des pondérations sur les branches ¹².

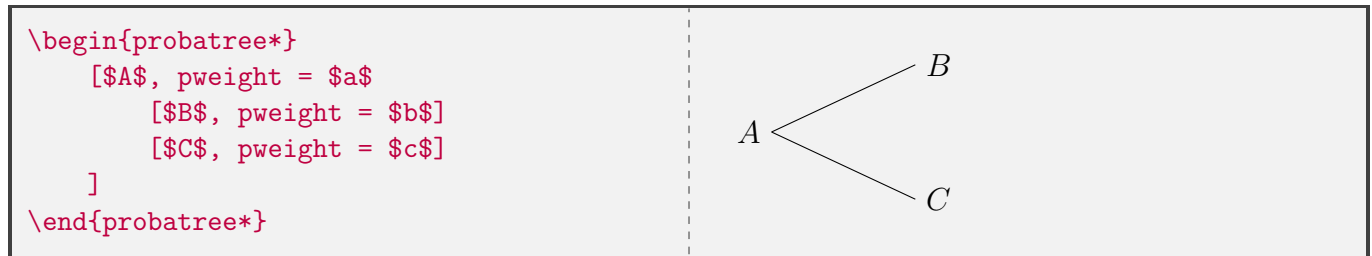
```
\begin{probatree}
[
  [$A$, pweight = $a$
    [$B$, pweight = $b$]
    [$C$, pweight = $c$]
  ]
  [$D$, bpweight = $d$
    [$E$, apweight = $e$]
    [$F$, bpweight = $f$]
  ]
]
\end{probatree}
```



12. **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.

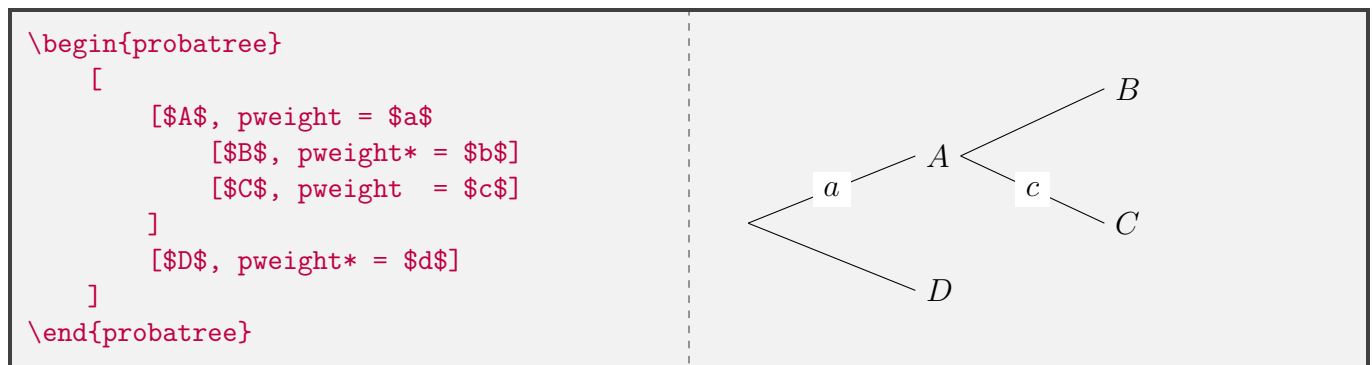
Exemple 2 – Des poids cachés partout

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



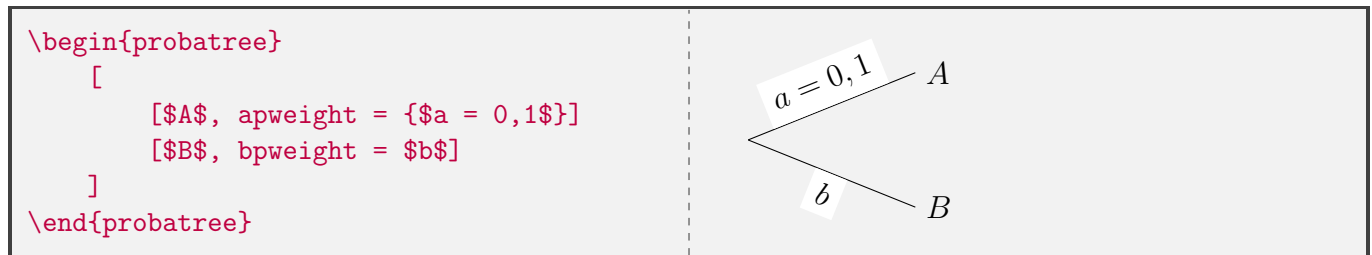
Exemple 3 – Des poids cachés localement

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



Exemple 4 – Un signe = et/ou une virgule dans les étiquettes

Vous ne pouvez pas utiliser directement un signe = ou une virgule dans les étiquettes des branches. L'astuce pour contourner cette limitation consiste juste à mettre le contenu de l'étiquette dans des accolades.



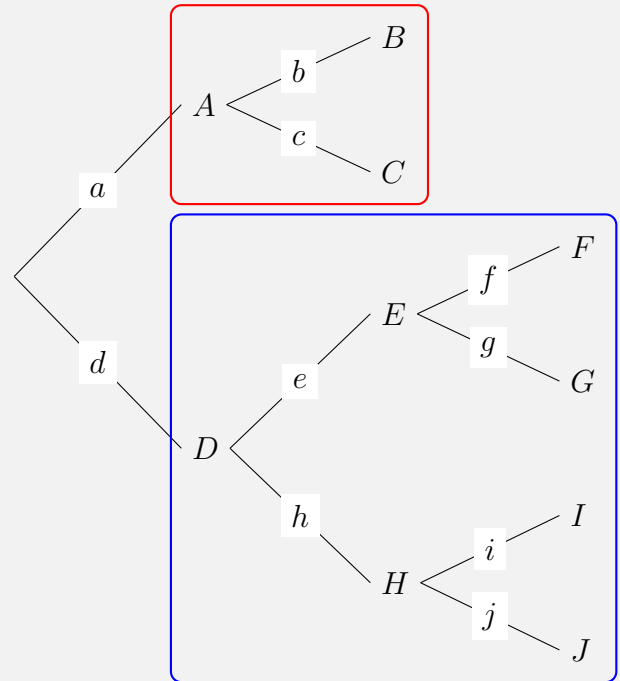
Exemple 5 – Des cadres facilement

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.

```

\begin{probatree}
[{}], s sep = 1.3cm
% Astuce pour espacer les cadres.
[$A$, 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}

```



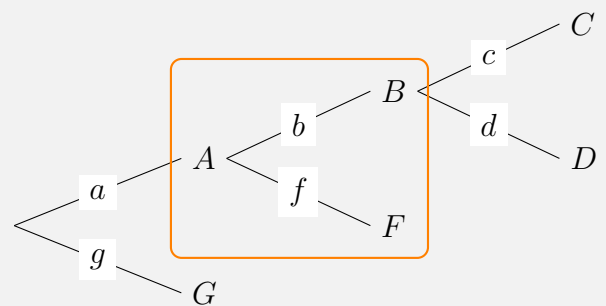
Exemple 6 – Des cadres faits à la main

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 est tout simplement infaisable avec la clé `frame`.

```

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

```



11 Arithmétique

11.1 Opérateurs de base

Pour des raisons d'expressivité des codes L^AT_EX, les opérateurs binaires `\divides`, `\ndivides` et `\modulo` ont été ajoutés comme alias respectifs de `\mid`, `\nmid` et `\bmod` qui sont proposés par le package `amssymb`. Un opérateur `\nequiv` a été aussi ajouté.

`$10 \divides 150$` au lieu de
`$10 \mid 150$`

`$10 \ndivides 154$` au lieu de
`$10 \not\mid 154$`

`$a \nequiv b \modulo p`
`\iff`
`p \ndivides (a - b)$`.

$10 \mid 150$ au lieu de $10 \mid 150$
 $10 \nmid 154$ au lieu de $10 \not\mid 154$
 $a \not\equiv b \bmod p \iff p \nmid (a - b)$.

11.2 Fractions continuées

11.2.1 Fractions continuées standard

Exemple

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

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

11.2.2 Fractions continuées généralisées

Exemple

Voici comment écrire une fraction continuée généralisée.

```


$$\frac{a + \frac{b}{c + \frac{d}{e + \frac{f}{\dots + \frac{y}{z}}}}}{1}$$


```

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

11.2.3 Comme une fraction continuée isolée

Exemple

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

```


$$\frac{a}{b}$$


```

$\frac{a}{b}$ pour les fous... :-)

11.2.4 L'opérateur \mathcal{K}

Exemple 1

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

```


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


```

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

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

Exemple 2

```

$$u_0 + \operatorname{contfrac}_{k=1}^n (1:u_k)$$
  


$$= \operatorname{contfrac}\{u_0 \mid u_1 \mid u_2 \mid \dots \mid u_n\}$$

```

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

12 Algèbre

12.1 Polynômes, séries formelles et compagnie

12.1.1 Polynômes et fractions polynômiales

Exemple 1 – Polynômes

```
 $\operatorname{setpoly}\{\mathbb{R}\}\{X\}$  ou  

 $\operatorname{setpoly}\{\mathbb{R}\}\{X \mid Y \mid Z\}$ 
```

$\mathbb{R}[X]$ ou $\mathbb{R}[X; Y; Z]$

Exemple 2 – Fractions polynômiales

```
 $\operatorname{setpolyfrac}\{\mathbb{Q}\}\{T\}$  ou  

 $\operatorname{setpolyfrac}\{\mathbb{Q}\}\%$   

 $\{S_1 \mid S_2 \mid \dots \mid S_k\}$ 
```

$\mathbb{Q}(T)$ ou $\mathbb{Q}(S_1; S_2; \dots; S_k)$

12.1.2 Séries formelles et leurs corps de fractions

Exemple 1 – Séries formelles

```
 $\operatorname{setserie}\{\mathbb{C}\}\{X\}$  ou  

 $\operatorname{setserie}\{\mathbb{C}\}\{T \mid O \mid P\}$ 
```

$\mathbb{C}[[X]]$ ou $\mathbb{C}[[T; O; P]]$

Exemple 2 – Corps des fractions de séries formelles

```
 $\operatorname{setseriefrac}\{\mathbb{Z}\}\{X\}$  ou  

 $\operatorname{setseriefrac}\{\mathbb{Z}\}\{Z \mid T \mid O \mid P\}$ 
```

$\mathbb{Z}((X))$ ou $\mathbb{Z}((Z; T; O; P))$

12.1.3 Polynômes de Laurent et séries formelles de Laurent

Exemple 1 – Polynômes de Laurent

Ci-dessous, la notation $\mathbb{R}\{X_1; X_2\}$ n'est pas standard.

```
\setpolylaurent{\RR}{X} \eqdef
\setpoly{\RR}{X | X^{-1}}$
```

$$\mathbb{R}\{X\} \stackrel{\text{déf}}{=} \mathbb{R}[X; X^{-1}]$$

```
\setpolylaurent{\RR}{X_1 | X_2} \eqdef
\setpoly{\RR}{X_1 | X_1^{-1} %
| X_2 | X_2^{-1}}$
```

$$\mathbb{R}\{X_1; X_2\} \stackrel{\text{déf}}{=} \mathbb{R}[X_1; X_1^{-1}; X_2; X_2^{-1}]$$

Exemple 2 – Séries formelles de Laurent

Ci-dessous, la notation $\mathbb{Q}\{\{X_1; X_2\}\}$ n'est pas standard.

```
\setserielaurent{\QQ}{X} \eqdef
\setserie{\QQ}{X | X^{-1}}$
```

$$\mathbb{Q}\{\{X\}\} \stackrel{\text{déf}}{=} \mathbb{Q}[[X; X^{-1}]]$$

```
\setserielaurent{\QQ}{X_1 | X_2} \eqdef
\setserie{\QQ}{X_1 | X_1^{-1} %
| X_2 | X_2^{-1}}$
```

$$\mathbb{Q}\{\{X_1; X_2\}\} \stackrel{\text{déf}}{=} \mathbb{Q}[[X_1; X_1^{-1}; X_2; X_2^{-1}]]$$

12.2 Matrices

Tout le boulot ou presque est fait par l'excellent package `nicematrix`¹³ avec l'ajout d'une macro « *maison* » à but pédagogique. Veuillez vous reporter à la documentation de `nicematrix` pour savoir comment s'y prendre en général.

12.2.1 Calculs expliqués des déterminants 2×2

Exemple

Dans l'exemple suivant, le préfixe `c` est pour `c`-alculer et `two` signifie « *deux* » en anglais¹⁴.

```
\calcdettwo* {a}{c}%
{b}{d}
= \calcdettwo {a}{c}%
{b}{d}
= \calcdettwo[exp]{a}{c}%
{b}{d}$
```

$$\begin{vmatrix} a & c \\ b & d \end{vmatrix} = \begin{vmatrix} a & \overset{c}{\curvearrowright} \\ b & \ominus d \end{vmatrix} = a d - b c$$

Remarque. Il existe deux autres types de développement.

1. $a \cdot d - b \cdot c$ s'obtient via l'option `cexp`.
2. $a \times d - b \times c$ s'obtient via l'option `texp`

`exp` est pour `exp-and` soit « *développer* » en anglais, `c` pour `\cdot` et enfin `t` pour `\times`.

¹³. On impose l'option `transparent`.

¹⁴. En coulisse on utilise les outils pour le critère de colinéarité présentés dans la section 8.2.5.

12.2.2 Quelques exemples pour bien démarrer

Exemple 1 – Vu dans 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

```
\begin{vmatrix}
  1 & & \cdots & \cdots & 1 & \\
  0 & & \ddots & & & \vdots \\
  \vdots & & \ddots & \ddots & & \vdots \\
  0 & & \cdots & 0 & & 1 \\
\end{vmatrix}
```

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

Exemple 3

```
\begin{bmatrix}
  1 & & \cdots & \cdots & 1 & \\
  0 & & \ddots & & & \vdots \\
  \vdots & & \ddots & \ddots & & \vdots \\
  0 & & \cdots & 0 & & 1 \\
\end{bmatrix}
```

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

Exemple 4 – Vu dans la documentation de nicematrix

```
\begin{pNiceMatrix}[name = mymatrix]
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{pNiceMatrix}

\tikz[remember picture,
  overlay]
\draw[red]
  (mymatrix-2-2) circle (2.5mm);
```

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

Exemple 5 – Vu dans la documentation de nicematrix

```

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

```

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

Exemple 6 – Proposition de l’auteur de nicematrix suite à une discussion par mail

```

% Besoin du package ‘‘ifthen‘‘.
\newcommand\aij{%
  a_{\arabic{iRow}\arabic{jCol}}%
}

 $\begin{bNiceArray}{*{5}{>{
\ifthenelse{\value{iRow}>0}{\aij}{}}{C}}[
  first-col,
  first-row,
  code-for-first-row
    = \mathbf{\arabic{jCol}},
  code-for-first-col
    = \mathbf{\arabic{iRow}}
]$ 

```

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 1 & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 2 & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \end{array}$$

Exemple 7 – Avec des calculs automatiques

```

\newcounter{cntaij}
\newcommand\aij{%
  \setcounter{cntaij}{\value{iRow}}%
  \addtocounter{cntaij}{\value{jCol}}%
  \addtocounter{cntaij}{-1}%
  \arabic{cntaij}%
}

Si $a_{ij} = i + j - 1$ alors

$(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5}$
=
\begin{bNiceArray}{*{5}{>{\aij}C}}
& & & & \backslash \\
& & & & \backslash \\
& & & & \\
\end{bNiceArray}$

```

Si $a_{ij} = i + j - 1$ alors

$$(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

13 Historique

Nous ne donnons ici qu'un très bref historique récent ¹⁵ de `lymath` à destination de l'utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier `change-log` : voir le code source de `lymath` sur `github`.

2020-06-27 Nouvelle version mineure `1.1.0-beta`.

- **LOGIQUE.**

- Suppression des environnements `aexplain` et `aexplain*`.
Leurs mises en forme restent accessibles respectivement via les options `style = ar` et `style = sar` de l'environnement `explain`.
- L'environnement `explain` propose des options de type clé-valeur.
- Ajout de petits commentaires pour les étapes via `\comthis` et si besoin `\comthis*`.
- Modification de `\explnext*` pour le mode universitaire sans flèche via l'ajout de `\exptxtupdown` qui gère la mise en forme de deux explications non vides. Ceci a pour effet l'alignement du rendu avec l'opérateur.
- Les environnements `demoexplain` et `demoexplain*` utilisent `longtable` en coulisse afin de pouvoir écrire un tableau sur plusieurs pages.

2020-06-21 Nouvelle version majeure `1.0.0-beta`.

- Le changement vers une nouvelle version majeure se justifie par de nouvelles règles strictes pour définir les signatures des macros. À l'avenir ces règles seront appliquées tout le temps sauf dans de très rares cas.

Ceci a créé beaucoup de nouvelles façons de rédiger.

-
- **ALGÈBRE LINÉAIRE** : `\calcdettwo` est un outil pédagogique pour expliquer le calcul d'un déterminant 2×2 .
-

- **ANALYSE.**

- Calcul intégral.
 - `\integrate` et `\dintegrate` servent à rédiger des intégrales simples.
 - `\hook` s'utilise différemment : on doit taper `\hook{a}{b}{F(x)}{x}` avec l'obligation de donner la variable.
 - `\hook` marche avec des options. Du coup `\vhook` and `\vhook*` ont été supprimées mais les mises en forme correspondantes existent toujours via `\hook[r]` et `\hook[sr]`.
- Dérivées totales.
 - Il ne reste plus que trois macros : `\sder`, `\der` et `\derope`.
 - `\sder` et `\sder[e]` remplacent `\derpow*` et `\derpow` avec en plus la possibilité d'ajout automatique de parenthèses pour faire comme avec `\derpar*`, `\derpar`, `\sderpar*` et `\sderpar` avant.

15. On ne va pas au-delà de un an depuis la dernière version.

- `\der` s'utilise en indiquant la variable de dérivation. Cette macro propose différentes options pour différentes mises en forme (*on peut toujours obtenir la même chose que ce que proposaient `\derfrac`, `\derfrac*` et `\dersub`*).
- `\derope` sert à écrire un opérateur fonctionnel.
- Dérivées partielles.
 - Il ne reste plus que deux macros : `\pder` et `\pderope`.
 - `\pder` possède des options permettant d'obtenir le même résultat qu'avec les anciennes macros `\partialfrac` et `\partialsub`.
 - La mise en forme proposée par `\partialprime` n'a pas été gardée.
 - `\pderope` sert à écrire un opérateur fonctionnel.

• ARITHMÉTIQUE.

- `\notdivides` a été renommée `\ndivides`.
 - Ajout de `\nequiv`.
-

• GÉOMÉTRIE.

- `\calcdetplane`, `\calcdetplane*`, `\vcalcdetplane` et `\vcalcdetplane*` permettent de détailler le calcul du déterminant de deux vecteurs en dimension 2 (*utile pour le critère de colinéarité*).
- `\calccrossprod`, `\calccrossprod*`, `\vcalccrossprod` et `\vcalccrossprod*` permettent de détailler le calcul du produit vectoriel de deux vecteurs en dimension 3.
- `\coordcrossprod` permet d'obtenir formellement les coordonnées d'un produit vectoriel avec des formats du type $(yz' - zy', zx' - xz', xy' - yx')$.
- Angles orientés.
 - `\angleorient` devient l'unique macro pour rédiger des angles orientés de différentes façons via des options.
 - `\angleorient*` a été remplacée par `\angleorient[sp]`.
 - `\hangleorient` a été remplacée par `\dotprod[h]`.
 - `\hangleorient*` a été remplacée par `\dotprod[sh]`.
- Produit scalaire.
 - `\dotprod` devient l'unique macro pour rédiger des produits scalaires de différentes façons grâce à des options.
 - `\adotprod` a été remplacée par `\dotprod[a]`.
 - `\adotprod*` a été remplacée par `\dotprod[sa]`.
- Coordonnées.
 - Il faut passer via l'une des macros : `\coord`, `\pcoord`, `\pcoord*`, `\vcoord` et `\vcoord*`. Toutes ces macros proposent des options pour choisir la mise en forme : des parenthèse en mode horizontal, des crochets en mode vertical...

- `\coord` est pour des coordonnées seules.
 - `\pcoord` et `\pcoord*` sont pour un point avec ses coordonnées.
 - `\vcoord` et `\vcoord*` sont pour un vecteur avec ses coordonnées.
 - La version étoilée `\coord*` a été supprimée.
 - Norme.
 - `\norm` fonctionne maintenant avec des options. Du coup `\norm*` a été supprimée mais la mise en forme correspondante existe toujours via `\norm[s]`.
 - `\vnorm` évite d'avoir à utiliser `\vect` pour des vecteurs juste nommés.
-

• LOGIQUE.

- La macro `\explain` a été supprimée pour être remplacée par l'environnement `explain` qui est redoutable d'efficacité pour détailler un calcul ou un raisonnement simple.
 - `explain` est complété par les deux environnements `aexplain` et `aexplain*` qui utilisent des flèches pour les indications.
 - Les environnements `demoexplain` et `demoexplain*` permettent de rédiger de « vraies » démonstrations via des tableaux efficaces.
 - Toutes les macros négatives avec pour préfixe `not` auparavant utilisent maintenant juste le préfixe `n`.
 - Tous les opérateurs de comparaison ont une version négative.
-

• PROBABILITÉS.

- `\probacond**` et `\dprobacond**` sont devenues `\probacondexp*` et `\probacondexp` respectivement.
- `\expval` est une nouvelle macro pour l'écriture symbolique de l'espérance d'une variable aléatoire.

2020-06-08 Nouvelle version mineure 0.7.0-beta.

• ANALYSE.

- Pour éviter des conflits avec d'autres packages les renommages suivants ont dû être faits.
 - `\ch` et `\ach` sont devenus `\fch` et `\afch` où `f` est pour `f-rench`.
 - `\sh` et `\ash` sont devenus `\fsh` et `\afsh`.
 - `\th` et `\ath` sont devenus `\fth` et `\afth`.
- Les macros `\acosh`, `\asinh` et `\atanh` ont été ajoutées.
- `\derpar` et `\derpar*` servent à rédiger des dérivées avec des parenthèses extensibles. En coulisse, `\derpow` et `\derpow*` sont appelées.
Pour utiliser des parenthèses non extensibles, on passera par `\sderpar` et `\sderpar*` où `s` est pour `s-mall`.
- Ajout de `\stdint` pour rendre public l'opérateur intégral proposé par défaut par L^AT_EX.

- **GÉOMÉTRIE.**

- La macro `\pts` a été supprimée car sans signification sémantique puisqu'un point peut être nommé avec deux lettres.
- Les macros `\gline` et `\pgline` servent à indiquer des droites définies par deux points.
- La macro `\hgline`, avec `h` pour `h-alf`, est pour les demi-droites définies par deux points.
- La macro `\segment` est utile pour les segments définis par deux points.

- **LOGIQUE.**

- Pour les inégalités, on peut maintenant utiliser le décorateur `plot`.
- Ajout des versions négatives des opérateurs logiques verticaux.

- **PROBABILITÉS.**

- Ajout de `\proba` pour écrire des probabilités.
- Le comportement de `\probacond` a été modifié pour le rendre plus logique.

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

- **ANALYSE.**

- `\hypergeo` est devenu `\seqhypergeo`.
- `\suprageo` est devenu `\seqsuprageo`.

- **ENSEMBLES :** `\CSinterval` a été déplacée dans le package `lyalgo` disponible à l'adresse <https://github.com/bc-latex/ly-algo>.

- **GÉOMÉTRIE :** `\notparallel` est devenu `\nparallel`.

- **LOGIQUE.**

- `\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.

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

- **ALGÈBRE.**

- `\polyset` est devenu `\setpoly`.
- `\polyfracset` 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.

- **ENSEMBLES.**
 - `\algeset` est devenu `\setalge`.
 - `\geoset` est devenu `\setgeo`.
 - `\geneset` est devenu `\setgene`.
 - `\proba` est devenu `\setproba`.
 - `\specialset` est devenu `\setspecial`.
- **LOGIQUE :** la macro `\explain` possède maintenant un argument optionnel pour indiquer l'espacement avant le symbole. Ceci s'accompagne de la suppression des macros obsolètes `\explain*` et `\textexplainspacebefore`.
- **PROBABILITÉ.**
 - 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.

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

- **ENSEMBLES :** pour l'informatique théorique la macro `\CSinterval` permet d'obtenir quelque chose comme `a..b`.
- **GÉOMÉTRIE :** la macro `\notparallel` a été rajoutée.
- **LOGIQUE :** il y a deux nouvelles macros sémantiques `\neqid` et `\eqchoice`.
- **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.

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

- **ARITHMÉTIQUE :** ajout des opérateurs `\divides`, `\notdivides` et `\modulo`.
- **DIVERS :** ajout des macros `\dsum` et `\dprod` qui sont vis à vis de `\sum` et `\prod` des équivalents de `\dfrac` pour `\frac`.
- **GÉOMÉTRIE.**
 - `\pts` permet d'indiquer plusieurs points.
 - `\parallel` utilise des obliques pour symboliser le parallélisme au lieu de barres verticales.
- **LOGIQUE.**
 - 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.

- **ALGÈBRE LINÉAIRE** : intégration du package `nicematrix` pour écrire des matrices.
- **ANALYSE** : intégration du package `tkz-tab` pour rédiger des tableaux de variations et de signes.
- **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.

14 Toutes les fiches techniques

14.1 Quelques modifications générales

14.1.1 Deux séparateurs d'arguments par défaut

`\lymathsep <macro>` (Sans argument)
`\lymathsubsep <macro>` (Sans argument)

14.1.2 Espace et fractions

`\frac <macro>` (2 Arguments)
`\dfrac <macro>` (2 Arguments)
`\stdfrac <macro>` (2 Arguments)
`\stddfraction <macro>` (2 Arguments)

- Argument 1: le numérateur.
- Argument 2: le dénominateur.

14.1.3 Espace et racines n-ièmes d'un réel

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

- Option: l'indice à indiquer pour une racine n-ième.
- Argument: le radicande, c'est à dire ce qui sera écrit sous le radical.

14.1.4 Espace après la négation logique

`\neg <macro>` (Sans argument)
`\stdneg <macro>` (Sans argument)

14.1.5 Sommes et produits en mode ligne

Les macros suivantes sans argument ont un comportement spécifique vis à vis des mises en index et en exposant.

`\dprod <macro>` (Sans argument)
`\dsum <macro>` (Sans argument)

14.2 Logique et fondements

14.2.1 Les textes pour les opérateurs de « comparaison algébrique » et de logique

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

`\textoptest <macro> (Sans argument)`

14.2.2 Les opérateurs de « comparaison algébrique »

`\eqdef <macro> (Sans argument)`
`\eqdef* <macro> (Sans argument)`
`\eqid <macro> (Sans argument)`
`\eqid* <macro> (Sans argument)`
`\eqplot <macro> (Sans argument)`
`\eqappli <macro> (Sans argument)`
`\eqchoice <macro> (Sans argument)`
`\eqcond <macro> (Sans argument)`
`\eqcons <macro> (Sans argument)`
`\eqhyp <macro> (Sans argument)`
`\eqtest <macro> (Sans argument)`

`\neqid <macro> (Sans argument)`
`\neqplot <macro> (Sans argument)`
`\neqappli <macro> (Sans argument)`
`\neqchoice <macro> (Sans argument)`
`\neqcond <macro> (Sans argument)`
`\neqcons <macro> (Sans argument)`
`\neqhyp <macro> (Sans argument)`
`\neqtest <macro> (Sans argument)`

`\lessplot <macro> (Sans argument)`
`\lessappli <macro> (Sans argument)`
`\lesschoice <macro> (Sans argument)`
`\lesscond <macro> (Sans argument)`
`\lesscons <macro> (Sans argument)`
`\lesshyp <macro> (Sans argument)`
`\lesstest <macro> (Sans argument)`

`\nlessplot <macro> (Sans argument)`
`\nlessappli <macro> (Sans argument)`
`\nlesschoice <macro> (Sans argument)`
`\nlesscond <macro> (Sans argument)`
`\nlesscons <macro> (Sans argument)`
`\nlesshyp <macro> (Sans argument)`
`\nlesstest <macro> (Sans argument)`

`\leqplot <macro> (Sans argument)`
`\leqappli <macro> (Sans argument)`
`\leqchoice <macro> (Sans argument)`
`\leqcond <macro> (Sans argument)`
`\leqcons <macro> (Sans argument)`
`\leqhyp <macro> (Sans argument)`

`\leqtest <macro> (Sans argument)`

`\nleqplot <macro> (Sans argument)`
`\nleqappli <macro> (Sans argument)`
`\nleqchoice <macro> (Sans argument)`
`\nleqcond <macro> (Sans argument)`
`\nleqcons <macro> (Sans argument)`
`\nleqhyp <macro> (Sans argument)`
`\nleqtest <macro> (Sans argument)`

`\gtrplot <macro> (Sans argument)`
`\gtrappli <macro> (Sans argument)`
`\gtrchoice <macro> (Sans argument)`
`\gtrcond <macro> (Sans argument)`
`\gtrcons <macro> (Sans argument)`
`\gtrhyp <macro> (Sans argument)`
`\gtrtest <macro> (Sans argument)`

`\ngtrplot <macro> (Sans argument)`
`\ngtrappli <macro> (Sans argument)`
`\ngtrchoice <macro> (Sans argument)`
`\ngtrcond <macro> (Sans argument)`
`\ngtrcons <macro> (Sans argument)`
`\ngtrhyp <macro> (Sans argument)`
`\ngtrtest <macro> (Sans argument)`

`\geqplot <macro> (Sans argument)`
`\geqappli <macro> (Sans argument)`
`\geqchoice <macro> (Sans argument)`
`\geqcond <macro> (Sans argument)`
`\geqcons <macro> (Sans argument)`
`\geqhyp <macro> (Sans argument)`
`\geqtest <macro> (Sans argument)`

14.2.3 Les opérateurs de logique

`\iff <macro> (Sans argument)`
`\iffappli <macro> (Sans argument)`
`\iffchoice <macro> (Sans argument)`
`\iffcond <macro> (Sans argument)`
`\iffcons <macro> (Sans argument)`
`\iffhyp <macro> (Sans argument)`
`\ifftest <macro> (Sans argument)`

`\niff <macro> (Sans argument)`
`\niffappli <macro> (Sans argument)`
`\niffchoice <macro> (Sans argument)`

`\niffcond <macro> (Sans argument)`
`\niffcons <macro> (Sans argument)`
`\niffhyp <macro> (Sans argument)`
`\nifftest <macro> (Sans argument)`

`\implies <macro> (Sans argument)`
`\impliesappli <macro> (Sans argument)`
`\implieschoice <macro> (Sans argument)`
`\impliescond <macro> (Sans argument)`
`\impliescons <macro> (Sans argument)`
`\implieshyp <macro> (Sans argument)`
`\impliestest <macro> (Sans argument)`

`\nimplies <macro> (Sans argument)`
`\nimpliesappli <macro> (Sans argument)`
`\nimplieschoice <macro> (Sans argument)`
`\nimpliescond <macro> (Sans argument)`
`\nimpliescons <macro> (Sans argument)`
`\nimplieshyp <macro> (Sans argument)`
`\nimpliestest <macro> (Sans argument)`

`\liesimp <macro> (Sans argument)`
`\liesimpappli <macro> (Sans argument)`
`\liesimpchoice <macro> (Sans argument)`
`\liesimpcond <macro> (Sans argument)`
`\liesimpcons <macro> (Sans argument)`
`\liesimphyp <macro> (Sans argument)`
`\liesimptest <macro> (Sans argument)`

`\nliesimp <macro> (Sans argument)`
`\nliesimpappli <macro> (Sans argument)`
`\nliesimpchoice <macro> (Sans argument)`
`\nliesimpcond <macro> (Sans argument)`
`\nliesimpcons <macro> (Sans argument)`
`\nliesimphyp <macro> (Sans argument)`
`\nliesimptest <macro> (Sans argument)`

14.2.4 Les opérateurs de logique « verticaux »

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

14.2.5 Des versions alternatives du quantificateur \exists

`\existmulti <macro> (1 Argument)`
`\nexistmulti <macro> (1 Argument)`

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

`\existstone <macro> (Sans argument)`
`\nexiststone <macro> (Sans argument)`

14.2.6 Détailler un raisonnement simple

`explain <env> [1 Option]`

— **Option**: la valeur utilise une syntaxe de type clé-valeur. Voici les différentes clés disponibles.

1. **ope** sert à définir l'opérateur utilisé dans tout l'environnement qui sera rédigé en mode mathématique. La valeur par défaut est `{=}` (*et non juste =*).
2. **style** sert à définir le style de mise en forme. Voici les différentes valeurs possibles.
 - (a) **u**, la valeur par défaut, est pour **u**-niversity.
 - (b) **ar** est pour **ar**-row.
 - (c) **sar** est pour **s**-hort **ar**-row.
3. **com** permet de demander l'alignement ou non des commentaires non étoilés entre eux.
 - (a) **nal**, la valeur par défaut, est pour **n**-ot **al**-igned.
 - (b) **al** est pour **al**-igned.

ATTENTION ! La macro `\explnext` est à utiliser sans argument au tout début de l'environnement *aexplain**.

`\explnext <macro> [1 Option] (1 Argument)` où `expl = expl-ain`

— **Option**: le symbole à utiliser pour une explication, la valeur par défaut étant celle du symbole de l'environnement `explain` où `\explnext` est utilisé.

— **Argument**: le texte de l'explication qui peut être vide si aucune explication n'est à afficher.

`\explnext* <macro> [1 Option] (2 Arguments)` où `expl = expl-ain`

— **Option**: le symbole à utiliser pour une explication, la valeur par défaut étant celle du symbole de l'environnement `explain` où `\explnext` est utilisé.

— **Argument 1**: le texte de l'explication pour la 1^{re} ligne. Ce texte peut être vide (*voir l'environnement aexplain pour la raison de ceci*).

— **Argument 2**: le texte de l'explication pour la 2^e ligne. Ce texte peut être vide (*voir l'environnement aexplain pour la raison de ceci*).

`\comthis <macro> (1 Argument)` où `com = com-ment`
`\comthis* <macro> (1 Argument)` où `com = com-ment`

— **Argument**: le texte d'un court commentaire.

14.2.7 Détailler un raisonnement simple – Mise en forme du texte

Les macros suivantes sont juste utilisées par l'environnement `explain`.

`\expltxtspacein <macro>` (Sans argument)

`\expltxt <macro>` (1 Argument) où `expl = expl-ain`

— Argument: le texte de l'explication que l'on veut mettre en forme.

`\expltxtdown <macro>` (1 Argument) où `expl = expl-ain`

— Argument: le texte de l'explication du haut vers le bas que l'on veut mettre en forme.

`\expltxtup <macro>` (1 Argument) où `expl = expl-ain`

— Argument: le texte de l'explication du bas vers le haut que l'on veut mettre en forme.

`\expltxtupdown <macro>` (2 Arguments) où `expl = expl-ain`

— Argument 1: le texte de l'explication du haut vers le bas que l'on veut mettre en forme.

— Argument 1: le texte de l'explication du bas vers le haut que l'on veut mettre en forme.

`\explcom <macro>` (1 Argument) où `expl = expl-ain` et `com = com-ment`

— Argument: le texte d'un court commentaire.

14.2.8 Détailler un « vrai » raisonnement via un tableau

`demoexplain <env>` [4 Options]

— Option *"start"*: le début de la numérotation des identifiants des justifications. La valeur par défaut est 1 et la valeur spéciale `last` permet de reprendre la numérotation là où elle s'était arrêtée le dernier environnement `demoexplain` ou `demoexplain*` utilisé.

— Option *"hyps"*: les hypothèses, au format texte, vérifiées au départ. Cet argument peut être vide et ne doit pas rentrer en conflit avec l'option `hyp`.

— Option *"hyp"*: une unique hypothèse, au format texte, vérifiée au départ. Cet argument peut être vide et ne doit pas rentrer en conflit avec l'option `hyps`.

— Option *"ccl"*: la conclusion, au format texte, du raisonnement détaillé. Cet argument peut être vide.

`demoexplain* <env>` [1 Option]

— Option *"start"*: le début de la numérotation des identifiants des justifications. La valeur par défaut est 1 et la valeur spéciale `last` permet de reprendre la numérotation là où elle s'était arrêtée le dernier environnement `demoexplain` ou `demoexplain*` utilisé.

`\demostep <macro> [1 Option] (Sans argument)`

— **Option**: un texte qui sera utilisé comme label global référénçant le numéro d’une justification.

`\explref <macro> (1 Argument) où expl = expl-ain et ref = ref-erence`

— **Argument**: un numéro de 1 ou 2 chiffres qui sera encadré comme le sont les numérotations des indications.

`\explref* <macro> (1 Argument) où expl = expl-ain et ref = ref-erence`

— **Argument**: un texte correspondant à un label global référénçant le numéro d’une justification.

14.2.9 Détailler un « vrai » raisonnement via un tableau - Textes utilisés

`\textexplainmiddleID <macro> (Sans argument)`

`\textexplainmiddlehyp <macro> (Sans argument)`

`\textexplainmiddleprop <macro> (Sans argument)`

`\textexplainmiddlecons <macro> (Sans argument)`

`\textexplainuniversityhyps <macro> (Sans argument)`

`\textexplainuniversityhyp <macro> (Sans argument)`

`\textexplainuniversityccl <macro> (Sans argument)`

14.3 Ensembles et applications

14.3.1 Ensembles versus accolades

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

— **Option**: la valeur par défaut est `b`. Voici les différentes valeurs possibles.

1. `b` : on utilise des accolades extensibles.
2. `sb` : on utilise des accolades non extensibles.

— **Argument**: la définition de l’ensemble.

— **Argument**: la définition de l’ensemble.

14.3.2 Ensembles pour la géométrie

`\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.

14.3.3 Ensembles probabilistes

`\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.

14.3.4 Ensembles pour l'algèbre générale

`\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.

14.3.5 Ensembles classiques

`\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)

`\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)

14.3.6 Des suffixes à la carte

`\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.

14.3.7 Intervalles réels - Notation française (?)

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 $[a; b]$.
-

`\interval0 <macro> (2 Arguments)`
`\interval0* <macro> (2 Arguments)`

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

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

`\interval0C <macro> (2 Arguments)`
`\interval0C* <macro> (2 Arguments)`

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

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

14.3.8 Intervalles réels - Notation américaine

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.

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

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

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

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

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

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

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

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

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

14.3.9 Intervalles discrets d'entiers

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 $\llbracket a; b\llbracket$.

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

`\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 $\llbracket a ; b \rrbracket$.

14.3.10 Unions et intersections

`\dcap` <macro> (Sans argument)

`\dcup` <macro> (Sans argument)

`\dsqcup` <macro> (Sans argument)

14.3.11 Cardinal, image et compagnie

`\card` <macro> (Sans argument)

`\card*` <macro> (Sans argument)

`\dom` <macro> (Sans argument)

`\codom` <macro> (Sans argument)

`\im` <macro> (Sans argument)

`\ker` <macro> (Sans argument)

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

`\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)

14.4 Géométrie

14.4.1 Points

`\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.

14.4.2 Lignes

`\gline <macro> [1 Option] (2 Arguments)` où `g` = g-eometry
`\pgline <macro> [1 Option] (2 Arguments)` où `p` = p-oint et `g` = g-eometry

— Option: pour indiquer les parenthèses ou crochets à utiliser, les valeurs possibles étant 0, valeur par défaut, C, CO et OC.

- Argument 1: le 1^{er} point géométrique.
- Argument 2: le 2^e point géométrique.

`\hgline <macro> (2 Arguments)` où `h` = h-alf et `g` = g-eometry
`\phgline <macro> (2 Arguments)` où `p` = p-oint, `h` = h-alf et `g` = g-eometry
`\segment <macro> (2 Arguments)`
`\psegment <macro> (2 Arguments)` où `p` = p-oint

- Argument 1: le 1^{er} point géométrique.
- Argument 2: le 2^e point géométrique.

14.4.3 Droites parallèles ou non

`\parallel <macro> (Sans argument)`
`\nparallel <macro> (Sans argument)`
`\stdparallel <macro> (Sans argument)`
`\stdnparallel <macro> (Sans argument)`

14.4.4 Vecteurs

`\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.

14.4.5 Norme

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

— Option: la valeur par défaut est b. Deux options disponibles.

1. `b` : des doubles barres extensibles sont utilisées.
2. `s` : des doubles barres non extensibles sont utilisées.

- Argument: le vecteur sur lequel appliquer la norme.

`\vnorm <macro> (1 Argument)`

— **Argument**: le nom du vecteur sur lequel appliquer la norme.

14.4.6 Produit scalaire

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

— **Option**: la valeur par défaut est `u` pour `u`-sual soit « *habituel* » en anglais. Voici les différentes valeurs possibles.

1. `u` : écriture habituelle avec un point.
2. `r` : écriture « à la physicienne » avec des chevrons extensibles.
3. `sr` : écriture « à la physicienne » avec des chevrons non extensibles.

— **Argument 1**: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

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

`\vdotprod <macro> [1 Option] (2 Arguments)` où `v = v-ector`

— **Option**: voir les explications précédentes données pour `\dotprod`.

— **Argument 1**: le nom du 1^{er} vecteur sans utiliser la macro `\vect`.

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

14.4.7 Produit vectoriel

`\crossprod <macro> (2 Arguments)`

— **Argument 1**: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

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

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

— **Argument 1**: le nom du 1^{er} vecteur sans utiliser la macro `\vect`.

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

`\calccrossprod <macro> (8 Arguments)` où `calc = calc-ulate`

`\calccrossprod* <macro> (8 Arguments)` où `calc = calc-ulate`

— **Argument 1**: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

— **Argument 2..4**: les coordonnées du 1^{er} vecteur.

— **Argument 5**: le 2^e vecteur qu'il faut taper via la macro `\vect`.

— **Argument 6..8**: les coordonnées du 2^e vecteur.

`\vcalccrossprod <macro> (8 Arguments)` où `calc = calc-ulate` et `v = v-ector`

`\vcalccrossprod* <macro> (8 Arguments)` où `calc = calc-ulate` et `v = v-ector`

— **Argument 1**: le 1^{er} vecteur sans utiliser la macro `\vect`.

— **Argument 2..4**: les coordonnées du 1^{er} vecteur.

— **Argument 5**: le 2^e vecteur sans utiliser la macro `\vect`.

— Argument 6..8: les coordonnées du 2^e vecteur.

`\coordcrossprod <macro> [1 Option] (6 Arguments)` où `coord = coord-inat`

— Option: la valeur par défaut est `p,s`. Voici les différentes valeurs possibles pour la mise en forme des coordonnées uniquement (*voir la section 14.4.9 page 82*).

1. `p` : écriture horizontale avec des parenthèses extensibles.
2. `sp` : écriture horizontale avec des parenthèses non extensibles.
3. `vp` : écriture verticale avec des parenthèses.
4. `b` : écriture horizontale avec des crochets extensibles.
5. `sb` : écriture horizontale avec des crochets non extensibles.
6. `vb` : écriture verticale avec des crochets.

Pour les produits, voici ce qui est proposé.

1. `s` : un espace pour séparer les facteurs de chaque produit.
2. `t` : `\times` comme opérateur de multiplication.
3. `c` : `\cdot` comme opérateur de multiplication.

On peut combiner deux types de choix en les séparant par une virgule comme dans `p,s` la valeur par défaut.

— Argument 1..3: les coordonnées du 1^{er} vecteur.

— Argument 4..6: les coordonnées du 2^e vecteur.

14.4.8 Plan – Déterminant de deux vecteurs

`\calcdetplane <macro> [1 Option] (6 Arguments)` où `calc = calc-ulate`

`\calcdetplane* <macro> [1 Option] (6 Arguments)` où `calc = calc-ulate`

— Option: la valeur par défaut est `vec`. Voici les différentes valeurs possibles.

1. `vec` : les vecteurs sont affichés si besoin.
2. `novec` : les vecteurs ne sont jamais affichés.
3. `exp` : ceci demande d’afficher une formule développée en utilisant un espace pour séparer les facteurs de chaque produit.
4. `cexp` : comme `exp` mais avec le symbole \cdot obtenu via `\cdot`.
5. `texp` : comme `exp` mais avec le symbole \times .

— Argument 1: le 1^{er} vecteur qu’il faut taper via la macro `\vect`.

— Argument 2: la 1^{re} coordonnée du 1^{er} vecteur.

— Argument 3: la 2^e coordonnée du 1^{er} vecteur.

— Argument 4: le 2^e vecteur qu’il faut taper via la macro `\vect`.

— Argument 5: la 1^{re} coordonnée du 2^e vecteur.

— Argument 6: la 2^e coordonnée du 2^e vecteur.

`\vcalcdetplane <macro> [1 Option] (6 Arguments)` où `calc = calc-ulate` et `v = v-ector`

`\vcalcdetplane* <macro> [1 Option] (6 Arguments)` où `calc = calc-ulate` et `v = v-ector`

— Option: voir les indications données pour les macros `\calcdetplane` et `\calcdetplane*`.

— Argument 1..6: voir les indications données pour les macros `\calcdetplane` et `\calcdetplane*`.

14.4.9 Coordonnées

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

— **Option**: la valeur par défaut est `p`. Voici les différentes valeurs possibles.

1. `p` : écriture horizontale avec des parenthèses extensibles.
2. `sp` : écriture horizontale avec des parenthèses non extensibles.
3. `vp` : écriture verticale avec des parenthèses.
4. `b` : écriture horizontale avec des crochets extensibles.
5. `sb` : écriture horizontale avec des crochets non extensibles.
6. `vb` : écriture verticale avec des crochets.

— **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.

`\pcoord <macro> [1 Option] (2 Arguments) où v = v-ertical`

— **Option**: voir les indications données pour la macro `\coord`.

— **Argument 1**: le point auquel sera appliqué automatiquement la macro `\pt`.

— **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

`\pcoord* <macro> [1 Option] (2 Arguments) où v = v-ertical`

— **Option**: voir les indications données pour la macro `\coord`.

— **Argument 1**: le point auquel ne sera pas appliqué automatiquement la macro `\pt`.

— **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

`\vcoord <macro> [1 Option] (2 Arguments) où v = v-ertical`

— **Option**: voir les indications données pour la macro `\coord`.

— **Argument 1**: le vecteur sans utiliser la macro `\vect`.

— **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

`\vcoord* <macro> [1 Option] (2 Arguments) où v = v-ertical`

— **Option**: voir les indications données pour la macro `\coord`.

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

— **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

14.4.10 Nommer un repère

`\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.

14.4.11 Arcs circulaires

`\circarc <macro> (1 Argument)` où `circ` = circ-ular

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

`\circarc* <macro> (2 Arguments)` où `circ` = circ-ular

— **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.

14.4.12 Angles géométriques « intérieurs »

`\anglein <macro> (1 Argument)` où `in` = in-terior

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

`\anglein* <macro> (2 Arguments)` où `in` = in-terior

— **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.

14.4.13 Angles orientés de vecteurs

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

— Option: la valeur par défaut est `p`. Voici les différentes valeurs possibles.

1. `p` : écriture habituelle avec des parenthèses extensibles.
2. `sp` : écriture habituelle avec des parenthèses non extensibles.
3. `h` : écriture avec un chapeau et des parenthèses extensibles.
4. `sh` : écriture avec un chapeau et des parenthèses non extensibles.

— 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> [1 Option] (2 Arguments)` où `v = v-ector`

— Option: voir les explications précédentes données pour `\angleorient`.

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

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

14.5 Analyse

14.5.1 Constantes classiques

`\ggamma <macro> (Sans argument)`

`\pppi <macro> (Sans argument)`

`\tttau <macro> (Sans argument)`

`\ee <macro> (Sans argument)`

`\ii <macro> (Sans argument)`

`\jj <macro> (Sans argument)`

`\kk <macro> (Sans argument)`

14.5.2 Constantes latines personnelles

`\param <macro> (1 Argument)`

— Argument: un texte utilisant l'alphabet latin. `\abs <macro> (1 Argument)`

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

— Argument: l'expression sur laquelle appliquer la fonction valeur absolue.

14.5.3 Sans paramètre

`\pgcd <macro> (Sans argument)`

`\ppcm <macro> (Sans argument)`

`\acos <macro> (Sans argument)`

`\asin <macro> (Sans argument)`

`\atan <macro> (Sans argument)`

`\arccosh <macro> (Sans argument)`

`\arcsinh <macro> (Sans argument)`

`\arctanh <macro> (Sans argument)`

`\acosh <macro>` (Sans argument)
`\asinh <macro>` (Sans argument)
`\atanh <macro>` (Sans argument)

`\fch <macro>` (Sans argument) où $f = f\text{-rench}$
`\fsh <macro>` (Sans argument) où $f = f\text{-rench}$
`\fth <macro>` (Sans argument) où $f = f\text{-rench}$

14.5.4 Avec un paramètre

`\expb <macro>` (1 Argument)

— Argument : la base de l'exponentielle

`\logb <macro>` (1 Argument)

— Argument : la base du logarithme

14.5.5 Des suites spéciales

`\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.

14.5.6 Calcul différentiel

`\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 .

14.5.7 Dérivations totales

`\der <macro> [1 Option] (3 Arguments)`

— Option: la valeur par défaut est `u`.

1. `u` : écriture usuelle avec des primes (*ceci nécessite d'avoir une valeur entière naturelle connue du nombre de dérivations successives*).
2. `e` : écriture via un exposant entre des parenthèses.
3. `i` : écriture via un indice.
4. `f` : écriture via une fraction en mode display.
5. `sf` : écriture via une fraction en mode non display.
6. `of` : écriture via une fraction en mode display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
7. `osf` : écriture via une fraction en mode non display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
8. `p` : ajout de parenthèses extensibles autour de la fonction.
9. `sp` : ajout de parenthèses non extensibles autour de la fonction.

— Argument 1: la fonction à dériver.

— Argument 2: l'ordre de dérivation.

— Argument 3: la variable de dérivation.

`\sder <macro> [1 Option] (2 Arguments)` où `s` = s-imple

— Option: la valeur par défaut est `u`. Les options disponibles sont `u`, `e`, `p` et `sp` : voir la fiche technique de `\sder` ci-dessus.

— Argument 1: la fonction à dériver.

— Argument 2: l'ordre de dérivation.

14.5.8 Opérateur de dérivation totale

`\derope <macro> [1 Option] (2 Arguments)` où `ope` = ope-rator

— Option: la valeur par défaut est `f`. Les options disponibles sont `f`, `sf` et `i` : voir la fiche technique de `\der` donnée un peu plus haut.

— Argument 1: la fonction à dériver.

— Argument 2: l'ordre de dérivation.

14.5.9 Dérivations partielles

`\pder <macro> [1 Option] (2 Arguments)` où `p` = p-artial

— Option: la valeur par défaut est `f`.

1. `f` : écriture via une fraction en mode display.
2. `sf` : écriture via une fraction en mode non display.
3. `of` : écriture via une fraction en mode display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
4. `osf` : écriture via une fraction en mode non display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
5. `i` : écriture via un indice.

- 6. `p` : ajout de parenthèses extensibles autour de la fonction.
- 7. `sp` : ajout de parenthèses non extensibles autour de la fonction.

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

— **Argument 2** : les variables utilisées avec leur ordre de dérivation pour la dérivation partielle en utilisant une syntaxe du type `x | y^2 | ...` qui indique de dériver suivant x une fois, puis suivant y trois fois... etc.

— **Argument 3** : l'ordre total de dérivation.

14.5.10 Opérateur de dérivation partielle

`\pderope <macro> [1 Option] (2 Arguments)` où `p` = `p-artial` et `ope` = `ope-rator`

— **Option** : la valeur par défaut est `f`. Les options disponibles sont `f`, `sf` et `i` : voir la fiche technique de `\pder` juste avant.

— **Argument 1** : les variables utilisées avec leur ordre de dérivation via la syntaxe indiquée ci-dessus.

— **Argument 2** : l'ordre total de dérivation.

14.5.11 Le symbole d'intégration standard

`\stdint <macro> (Sans argument)`

14.5.12 Fonctionnelle d'intégration

`\integrate <macro> (4 Arguments)`

`\integrate* <macro> (4 Arguments)`

`\dintegrate <macro> (4 Arguments)` où `d` = `d-isplaystyle`

`\dintegrate* <macro> (4 Arguments)` où `d` = `d-isplaystyle`

— **Argument 1** : ce qui est en bas du symbole \int_{\bullet} .

— **Argument 2** : ce qui est en haut du symbole \int^{\bullet} .

— **Argument 3** : la fonction intégrée.

— **Argument 4** : suivant quoi on intègre.

14.5.13 L'opérateur crochet

`\hook <macro> [1 Option] (4 Arguments)`

`\hook* <macro> [1 Option] (4 Arguments)`

— **Option** : la valeur par défaut est `b`. Voici les différentes valeurs possibles.

1. `b` : des crochets extensibles sont utilisés.
2. `sb` : des crochets non extensibles sont utilisés.
3. `r` : un unique trait vertical extensible est utilisé à droite.
4. `sr` : un unique trait vertical non extensible est utilisé à droite.

— **Argument 1** : ce qui est en bas du crochet fermant.

— **Argument 2** : ce qui est en haut du crochet fermant.

— **Argument 3** : la fonction sur laquelle effectuer le calcul.

— **Argument 4** : la variable pour les calculs.

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

`\bigO <macro> (1 Argument)`

`\smallO <macro> (1 Argument)`

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

14.5.15 La notation Ω

`\bigomega <macro> (1 Argument)`

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

14.5.16 La notation Θ

`\bigtheta <macro> (1 Argument)`

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

14.6 Probabilité

14.6.1 Probabilité « simple »

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

— **Option**: le nom de la probabilité. La valeur par défaut est `p`.

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

14.6.2 Probabilité conditionnelle

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

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

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

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

— **Option**: le nom de la probabilité. La valeur par défaut est `p`.

— **Argument 1**: l'ensemble qui donne la condition.

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

14.6.3 Espérance

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

— **Option**: le nom de la fonction espérance. La valeur par défaut est `E`.

— **Argument**: la variable aléatoire dont on veut calculer l'espérance.

14.6.4 Arbres pondérés

`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 depuis un noeud vers toutes les feuilles de celui-ci.

14.7 Arithmétique

14.7.1 Opérateurs de base

`\divides <macro>` (Sans argument)
`\ndivides <macro>` (Sans argument)
`\nequiv <macro>` (Sans argument)
`\modulo <macro>` (Sans argument)

14.7.2 Fractions continuées standard

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

- Argument: tous les éléments de la fraction continuée séparés par des |.

14.7.3 Fractions continuées généralisées

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

- Argument: tous les éléments de la fraction continuée généralisée séparés par des |.

14.7.4 Comme une fraction continuée isolée

`\singlecontfrac <macro>` (2 Arguments)

- Argument 1: le pseudo numérateur.
- Argument 2: le pseudo dénominateur.

14.7.5 L'opérateur \mathcal{K}

La macro suivante sans argument a un comportement spécifique vis à vis des mises en index et en exposant.

`\contfracope <macro>` (Sans argument)

14.8 Algèbre

14.8.1 Polynômes, séries formelles et compagnie

`\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.

14.8.2 Calculs expliqués des déterminants 2×2

`\calcdettwo <macro> [1 Option] (4 Arguments)` où `c = c-alculate`
`\calcdettwo* <macro> [1 Option] (4 Arguments)` où `c = c-alculate`

— **Option**: la valeur par défaut est `std` pour `standard`. Voici les différentes valeurs possibles.

1. `std` : on utilise l'écriture matricielle.
 2. `exp` : ceci demande d'afficher une formule développée en utilisant \times pour les produits.
 3. `cexp` : comme `exp` mais avec le symbole \cdot obtenu via `\cdot`.
 4. `sexp` : comme `exp` mais avec un espace pour séparer les facteurs de chaque produit.
- **Argument 1**: l'entrée à la position (1,1)
 - **Argument 2**: l'entrée à la position (1,2)
 - **Argument 3**: l'entrée à la position (2,1)
 - **Argument 4**: l'entrée à la position (2,2)