

Le package `tnssets` : théorie générale des ensembles et des applications

Code source disponible sur <https://github.com/typensee-latex/tnssets.git>.

Version 0.4.0-beta développée et testée sur Mac OS X.

Christophe BAL

2021-03-02

Table des matières

I.	Introduction	3
II.	Beta-dépendance	3
III.	Packages utilisés	3
IV.	Ensembles	3
	1. Ensembles versus accolades	3
	2. Ensembles pour la géométrie	3
	3. Ensembles probabilistes	4
	4. Ensembles pour l'algèbre générale	4
	5. Ensembles classiques en mathématiques et en informatique théorique	4
	i. La liste complète	4
	ii. Ensembles classiques suffixés	5
	iii. Des suffixes à la carte	5
V.	Intervalles	5
	1. Intervalles réels – Notation française (?)	5
	2. Intervalles réels – Notation américaine	6
	3. Intervalles discrets d'entiers	6
VI.	Unions et intersections en mode ligne	6
VII.	Des applications très spéciales	7
	1. Fonction identité	7
	2. Fonction caractéristique	8
VIII.	Applications	8
	1. Cardinal, image et compagnie	8
	2. Application totale, partielle, injective, surjective et/ou bijective	8
	3. Définition explicite d'une fonction	9
	4. Composition	10
IX.	Historique	12

X.	Toutes les fiches techniques	13
1.	Ensembles	13
i.	Ensembles versus accolades	13
ii.	Ensembles pour la géométrie	13
iii.	Ensembles probabilistes	13
iv.	Ensembles pour l'algèbre générale	13
v.	Ensembles classiques en mathématiques et en informatique théorique	14
vi.	Ensembles classiques en mathématiques et en informatique théorique	14
2.	Intervalles	14
i.	Intervalles réels – Notation française (?)	14
ii.	Intervalles réels – Notation américaine	15
iii.	Intervalles discrets d'entiers	15
3.	Unions et intersections en mode ligne	16
4.	Des applications très spéciales	16
i.	Fonction identité	16
ii.	Fonction caractéristique	16
5.	Applications	16
i.	Cardinal, image et compagnie	16
ii.	Application totale, partielle, injective, surjective et/ou bijective	17
iii.	Définition explicite d'une fonction	17
iv.	Composition	17

I. Introduction

Le package `tnssets` propose des macros utiles pour la rédaction de texte sur la théorie basique et générale des ensembles et des applications via un codage sémantique simple.

II. Beta-dépendance

`tnscom` qui est disponible sur <https://github.com/typensee-latex/tnscom.git> est un package utilisé en coulisse.

III. Packages utilisés

La roue ayant déjà été inventée, le package `tnssets` utilise les packages suivants sans aucun scrupule.

- `amssymb`
- `forloop`
- `mathtools`
- `stackengine`
- `dsfont`
- `mathrsfs`
- `relsize`
- `xstring`

IV. Ensembles

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>	$\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$

2. Ensembles pour la géométrie

Exemple 1

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

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

Exemple 2 – Avec des indices

$\text{\texttt{\$}\setgeo*{C}{1}\text{\texttt{\$}}}$ ou $\text{\texttt{\$}\setgeo*{C}{2}\text{\texttt{\$}}}$	\mathcal{C}_1 ou \mathcal{C}_2
---	------------------------------------

3. Ensembles probabilistes

Exemple 1

$\text{\texttt{\$}\setproba{E}\text{\texttt{\$}}}$ ou $\text{\texttt{\$}\setproba{G}\text{\texttt{\$}}}$	\mathcal{E} ou \mathcal{G}
---	--------------------------------

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

Exemple 2 – Avec des indices

$\text{\texttt{\$}\setproba*{E}{1}\text{\texttt{\$}}}$ ou $\text{\texttt{\$}\setproba*{E}{2}\text{\texttt{\$}}}$	\mathcal{E}_1 ou \mathcal{E}_2
---	------------------------------------

4. Ensembles pour l'algèbre générale

Exemple 1

$\text{\texttt{\$}\setalge{A}\text{\texttt{\$}}}$, $\text{\texttt{\$}\setalge{K}\text{\texttt{\$}}}$, $\text{\texttt{\$}\setalge{h}\text{\texttt{\$}}}$ ou $\text{\texttt{\$}\setalge{k}\text{\texttt{\$}}}$	\mathbb{A} , \mathbb{K} , \mathbb{h} ou \mathbb{k}
---	--

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

Exemple 2 – Avec des indices

$\text{\texttt{\$}\setalge*{k}{1}\text{\texttt{\$}}}$ ou $\text{\texttt{\$}\setalge*{k}{2}\text{\texttt{\$}}}$	\mathbb{k}_1 ou \mathbb{k}_2
--	----------------------------------

5. Ensembles classiques en mathématiques et en informatique théorique

i. 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*).

$\text{\texttt{\$}\nullset\text{\texttt{\$}}}$	
$\text{\texttt{\$}\NN\text{\texttt{\$}}}$, $\text{\texttt{\$}\ZZ\text{\texttt{\$}}}$, $\text{\texttt{\$}\PP\text{\texttt{\$}}}$	\emptyset \mathbb{N} , \mathbb{Z} , \mathbb{P}
$\text{\texttt{\$}\DD\text{\texttt{\$}}}$, $\text{\texttt{\$}\QQ\text{\texttt{\$}}}$, $\text{\texttt{\$}\RR\text{\texttt{\$}}}$, $\text{\texttt{\$}\CC\text{\texttt{\$}}}$	\mathbb{D} , \mathbb{Q} , \mathbb{R} , \mathbb{C}
$\text{\texttt{\$}\HH\text{\texttt{\$}}}$, $\text{\texttt{\$}\OO\text{\texttt{\$}}}$	\mathbb{H} , \mathbb{O}
$\text{\texttt{\$}\FF\text{\texttt{\$}}}$	\mathbb{F}

ii. Ensembles classiques suffixés

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

$\$\\RRn\$$, $\$\\RRp\$$, $\$\\RRs\$$	\mathbb{R}_- , \mathbb{R}_+ , \mathbb{R}^*
$\$\\RRsn\$$, $\$\\RRsp\$$	\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 $\$\\CCn\$$ pour \mathbb{C}_- car l'ensemble \mathbb{C} ne possède pas de structure ordonnée standard. Jetez un oeil à la section suivante pour apprendre à taper \mathbb{C}_- si vous en avez besoin. L'interdiction est ici purement sémantique !

Remarque. La table 1 de la présente page montre les associations autorisées entre ensembles classiques et suffixes.

TABLE 1 – Suffixes

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

iii. Des suffixes à la carte

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

$\$\\setspecial\\{\\CC\\}{n}\$$, $\$\\setspecial\\{\\HH\\}{sp}\$$ ou $\$\\setspecial*\\{\\setproba{P}\\}{n}\$$	\mathbb{C}_- , \mathbb{H}_+^* ou $\mathcal{P}_{\leq 0}$
---	---

V. Intervalles

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 = .

$$\text{\$I =]a ; b] = \text{\textbackslashintervalOC\{a\}\{b\}\$}$$

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

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} }$
```

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

2. Intervalles réels – Notation américaine

Dans l'exemple suivant la syntaxe fait référence à P-arenthèse. Cette notation est utilisée aux États Unis.

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

$$(a ; b] =]a ; b] \text{ et } (a ; b) =]a ; b[.$$

3. Intervalles discrets d'entiers

Dans l'exemple suivant la syntaxe fait référence à \mathbb{Z} l'ensemble des entiers relatifs (les crochets doubles \llbracket et \rrbracket sont accessibles via `\Zlbracket` et `\Zrbracket` directement¹).

```
\ZintervalC{-1}{4} =
\{ -1 ; 0 ; 1 ; 2 ; 3 ; 4 \}$
\ZintervalC{-1}{4} =
\ZintervalO{-2}{5}$.
```

$$\begin{aligned} \llbracket -1 ; 4 \rrbracket &= \{-1 ; 0 ; 1 ; 2 ; 3 ; 4\} \\ \llbracket -1 ; 4 \rrbracket &= \llbracket -2 ; 5 \rrbracket. \end{aligned}$$

VI. Unions et intersections en mode ligne

L^AT_EX permet d'afficher sans souci $\cup_{k=1}^n$ mais ne propose pas $\bigcup_{k=1}^n$. Les macros `\dcap`, `\dcup` et `\dsqcup` donnent accès à ce type de fonctionnalité pour \cap , \cup et \sqcup respectivement. Voici des exemples d'utilisation.

1. Ces deux symboles « proviennent » du code source du package `stmaryrd`, lequel package n'est pas chargé.

Exemple 1 – Les symboles « seuls »

<code>\$A \dcap B = C \cap D\$</code>	$A \cap B = C \cap D$
<code>\$A \dcup B = C \cup D\$</code>	$A \cup B = C \cup D$
<code>\$A \dsqcup B = C \sqcup D\$</code>	$A \sqcup B = C \sqcup D$

Exemple 2 – Des intersections indicées

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

<code>\$\cap_{k=1}^n A_k\$, <code>\$\dcap_{k=1}^n B_k\$, <code>\$\bigcap_{k=1}^n C_k\$, <code>\$\displaystyle%</code> <code>\bigcap_{k=1}^n D_k\$</code></code></code></code>	$\cap_{k=1}^n A_k , \cap_{k=1}^n B_k , \cap_{k=1}^n C_k , \bigcap_{k=1}^n D_k$
--	--

Exemple 3 – Des unions indicées

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

<code>\$\cup_{k=1}^n A_k\$, <code>\$\dcup_{k=1}^n B_k\$, <code>\$\bigcup_{k=1}^n C_k\$, <code>\$\displaystyle%</code> <code>\bigcup_{k=1}^n D_k\$</code></code></code></code>	$\cup_{k=1}^n A_k , \cup_{k=1}^n B_k , \cup_{k=1}^n C_k , \bigcup_{k=1}^n D_k$
--	--

Exemple 4 – Des unions disjointes indicées

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

<code>\$\sqcup_{k=1}^n A_k\$, <code>\$\dsqcup_{k=1}^n B_k\$, <code>\$\bigsqcup_{k=1}^n C_k\$, <code>\$\displaystyle%</code> <code>\bigsqcup_{k=1}^n D_k\$</code></code></code></code>	$\sqcup_{k=1}^n A_k , \sqcup_{k=1}^n B_k , \sqcup_{k=1}^n C_k , \bigsqcup_{k=1}^n D_k$
--	--

VII. Des applications très spéciales

1. Fonction identité

<code>\$\id_A\$ vérifie par définition <code>\$\forall x \in A\$, \$\id_A(x) = x\$.</code></code>	Id_A vérifie par définition $\forall x \in A, \text{Id}_A(x) = x$.
--	--

2. Fonction caractéristique

Exemple 1 – Version très utilisée

$\text{\texttt{\$}\charact_A\$}$ vérifie par définition $\text{\texttt{\$}\forall x \in A\$, \text{\texttt{\$}\charact_A(x) = 1\$}}$ et $\text{\texttt{\$}\forall x \notin A\$, \text{\texttt{\$}\charact_A(x) = 0\$}}$.	χ_A vérifie par définition $\forall x \in A, \chi_A(x) = 1$ et $\forall x \notin A, \chi_A(x) = 0$.
--	--

Exemple 2 – Version alternative

$\text{\texttt{\$}\caractone_A = \charact_A\$}$	$\mathbb{1}_A = \chi_A$
---	-------------------------

VIII. Applications

1. Cardinal, image et compagnie

Exemple 1 – Cardinal

$\text{\texttt{\$}\card* E = \card E\$}$	$\#E = \text{card } E$
--	------------------------

Exemple 2 – Image et compagnie

Ci-dessous se trouve la macro `\ker` proposée par `amsmath` qui est importé par `tnssets`.

$\text{\texttt{\$}\ker f\$}$, $\text{\texttt{\$}\dom f\$}$, $\text{\texttt{\$}\im f\$}$ ou $\text{\texttt{\$}\codom f\$}$	$\ker f$, $\text{dom } f$, $\text{im } f$ ou $\text{codom } f$
--	--

2. 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 partielles² (*on parle aussi d'applications, sans qualificatif, et de fonctions*).

Exemple 1 – Applications totales

2. $a : E \rightarrow F$ 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 .

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

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

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

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

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

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

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

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

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 .

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

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

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

3. Définition explicite d'une fonction

Exemple 1 – Écriture par défaut

`$\funcdef{f}{x}{x^2}%`
`{I}{J}$`

$f : \left| \begin{array}{l} I \rightarrow J \\ x \mapsto x^2 \end{array} \right.$

Remarque. Même si cela est peu utile, vous pouvez utiliser la mise en forme dans du texte pour obtenir $f : \left| \begin{array}{l} I \rightarrow J \\ x \mapsto x^2 \end{array} \right.$ mais c'est un peu affreux.

Exemple 2 – Écriture alternative

On peut cacher le trait vertical via l'option `s` pour `s`-hort soit « *court* » en anglais.

`$\funcdef[s]{f}{x}{x^2}%`
`{I}{J}$`

$f : I \rightarrowtail J$
 $x \mapsto x^2$

Exemple 3 – Écriture en ligne

Pour avoir tout sur une ligne, ce qui est l'idéal pour une insertion dans du texte, il suffit d'utiliser l'option `h` pour `h`-horizontal.

Soit <code>\funcdef[h]{f}{x}{x^2}% {I}{J}\$...</code>	Soit $f : x \in I \mapsto x^2 \in J \dots$
--	--

Exemple 4 – Écriture en ligne incomplète

En mode horizontal, les ensembles peuvent être de valeur vide pour ne pas les indiquer.

<code>\funcdef[h]{f}{x}{x^2}{}{J}\$</code>	$f : x \mapsto x^2 \in J$
<code>\funcdef[h]{f}{x}{x^2}{I}{}\$</code>	$f : x \in I \mapsto x^2$
<code>\funcdef[h]{f}{x}{x^2}{}{}\$</code>	$f : x \mapsto x^2$

Exemple 5 – Écriture textuelle

On peut enfin obtenir une version « *textuelle* » via la macro `\txtfuncdef` où `txt` est pour `texte`. Cette macro ne s'utilise qu'en mode texte³ et elle accepte l'omission de l'un ou des deux ensembles.

<code>\txtfuncdef{f}{x}{x^2}{I}{J}</code>	$f(x) = x^2$ pour $x \in I$ (on sait que $f(x) \in J$ est toujours vérifié)
<code>\txtfuncdef{f}{x}{x^2}{}{J}</code>	$f(x) = x^2$ (on sait que $f(x) \in J$ est toujours vérifié)
<code>\txtfuncdef{f}{x}{x^2}{I}{}\$</code>	$f(x) = x^2$ pour $x \in I$
<code>\txtfuncdef{f}{x}{x^2}{}{}\$</code>	$f(x) = x^2$

4. Composition

Exemple 1 – Opérateur

La macro `\compo` est juste une version un peu plus petite de `\circ`.

<code>\$f \compo g\$</code> et non <code>\$f \circ g\$</code>	$f \circ g$ et non $f \circ g$
--	--------------------------------

Exemple 2 – Compositions successives

La macro `\multicompo` sert à indiquer la composition d'une application plusieurs fois de suite par elle-même⁴. Voici toutes les mises en forme disponibles où l'option `exp` nécessite que le nombre d'applications composées soit un naturel non nul connu. Vous noterez que l'écriture par défaut, qui n'est pas standard, n'est pas $f^{(p)}$ car cette notation est traditionnellement utilisée pour indiquer la dérivée p^e d'une application.

3. Logic! Isn't it?

4. Une telle fonction f doit vérifier $\text{im } f \subseteq \text{dom } f$.

```

 $\backslashmulticompo$        $\{g\}\{5\}$ 
=  $\backslashmulticompo[\text{exp}]\{g\}\{5\}$ 

 $\backslashmulticompo$        $\{f\}\{p\}$ 
=  $\backslashmulticompo[\text{dot}]\{f\}\{p\}$ 

```

$$g^{\langle 5 \rangle} = g \circ g \circ g \circ g \circ g$$

$$f^{\langle p \rangle} = f \circ \dots \circ f$$

Remarque. La convention retenue est analogue à ce que l'on fait avec les puissances de nombres réels. En particulier, $f^{\langle 1 \rangle} = f$ et $f^{\langle 0 \rangle} = \text{Id}_{\text{dom } f}$.

IX. Historique

Nous ne donnons ici qu'un très bref historique récent⁵ de `tnssets` à destination de l'utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier `change-log` : voir le code source de `tnssets` sur `github`.

2021-03-02 Nouvelle version mineure 0.4.0-beta.

- **CROCHETS DOUBLES** : ajout des symboles `\Zlbracket` et `\Zrbracket`⁶.
-

2020-08-05 Nouvelle version mineure 0.3.0-beta.

- **DÉFINITION EXPLICITE D'UNE FONCTION** : intégration de deux macros proposées avant par `tnsana` disponible sur <https://github.com/typensee-latex/tnsana.git>.
 - `\funcdef` peut produire trois versions symboliques.
 - `\txtfuncdef` produit une version textuelle courte.
 - **FONCTIONS SPÉCIALES**.
 - Ajout de `\id` pour la fonction identité.
 - Ajout de `\caract` et `\caractone` pour deux versions de la fonction caractéristique d'un ensemble.
-

2020-07-30 Nouvelle version mineure 0.2.0-beta.

- **COMPOSITION D'APPLICATIONS**.
 - `\compo` est un opérateur de composition de deux applications.
 - `\multicomp` permet d'indiquer des compositions successives d'une application par elle-même.
-

2020-07-10 Première version 0.0.0-beta.

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

6. En coulisse, on commence à tenter de gérer plus proprement les symboles particuliers utilisés.

X. Toutes les fiches techniques

1. Ensembles

i. Ensembles versus accolades

`\setgene[#opt]{#1}`

— **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.

ii. Ensembles pour la géométrie

`\setgeo{#1}`

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

`\setgeo*{#1..#2}`

— **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.

iii. Ensembles probabilistes

`\setproba{#1}`

— **Argument**: un seul caractère ASCII majuscule indiquant un ensemble probabiliste.

`\setproba*{#1..#2}`

— **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.

iv. Ensembles pour l'algèbre générale

`\setalge{#1}`

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

`\setalge*{#1..#2}`

— **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.

v. Ensembles classiques en mathématiques et en informatique théorique

Ensembles classiques suffixés

`\NN` `\NNs`

`\PP`

`\ZZ` `\ZZn` `\ZZp` `\ZZs` `\ZZsn` `\ZZsp`

`\DD` `\DDn` `\DDp` `\DDs` `\DDsn` `\DDsp`

`\QQ` `\QQn` `\QQp` `\QQs` `\QQsn` `\QQsp`

`\RR` `\RRn` `\RRp` `\RRs` `\RRsn` `\RRsp`

`\CC` `\CCs`

`\HH` `\HHs`

`\OO` `\OOs`

vi. Ensembles classiques en mathématiques et en informatique théorique

Des suffixes à la carte

`\setspecial {#1..#2}`

`\setspecial*{#1..#2}`

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

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

2. Intervalles

i. 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.

`\intervalC0 {#1..#2}`

`\intervalC0*{#1..#2}`

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

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

`\intervalC {#1..#2}`
`\intervalC*{#1..#2}`

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

`\intervalO {#1..#2}`
`\intervalO*{#1..#2}`

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

`\intervalOC {#1..#2}`
`\intervalOC*{#1..#2}`

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

ii. 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 {#1..#2}`
`\intervalCP*{#1..#2}`

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

`\intervalP {#1..#2}`
`\intervalP*{#1..#2}`

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

`\intervalPC {#1..#2}`
`\intervalPC*{#1..#2}`

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

iii. 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.

`\ZintervalCO {#1..#2}`

`\ZintervalC0*{#1..#2}`

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

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

`\ZintervalC {#1..#2}`

`\ZintervalC*{#1..#2}`

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

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

`\ZintervalI0 {#1..#2}`

`\ZintervalI0*{#1..#2}`

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

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

`\ZintervalI0C {#1..#2}`

`\ZintervalI0C*{#1..#2}`

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

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

3. Unions et intersections en mode ligne

`\dcap`

`\dcup`

`\dsqcup`

4. Des applications très spéciales

i. Fonction identité

`\id`

ii. Fonction caractéristique

`\caract`

`\caractone`

5. Applications

i. Cardinal, image et compagnie

`\card`

`\card*`

`\dom`

`\codom`

`\im`

ii. Application totale, partielle, injective, surjective et/ou bijective

<code>\to</code>	<code>\pto</code>
<code>\onetoone</code>	<code>\ponetoone</code>
<code>\onto</code>	<code>\ponto</code>
<code>\bijet</code>	<code>\pbijet</code>

iii. Définition explicite d'une fonction

`\funcdef` [`#opt`] {`#1`..`#5`}

— `Option`: la valeur par défaut `u`.

1. `u` : écriture empilée avec un trait vertical.
2. `s` : écriture empilée sans trait vertical.
3. `h` : écriture horizontale en ligne.

— `Argument 1`: la fonction.

— `Argument 2`: la variable.

— `Argument 3`: la formule explicite de définition.

— `Argument 4`: l'ensemble de départ. Cet argument peut être vide si le mode `h` est activé

— `Argument 5`: l'ensemble d'arrivée.

`\txtfuncdef` {`#1`..`#5`}

— `Arguments 1..5`: voir les explications données ci-dessus pour la macro `\funcdef`.

iv. Composition

<code>\compo</code>	<code>compo = compo-sition</code>
---------------------	-----------------------------------

`\multicompo` [`#opt`] {`#1`..`#2`}

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

- | | |
|--|-----------------------------|
| 1. <code>r</code> : écriture utilisant des chevrons. | <code>r = r-after.</code> |
| 2. <code>exp</code> : écriture développée. | <code>exp = exp-and.</code> |
| 3. <code>dot</code> : écriture faussement développée utilisant des points de suspension. | |

— `Argument 1`: l'application.

— `Argument 2`: le nombre d'applications composées.