Paquete `mycats`

Luis Turcio

15 de enero de 2024

1 Introducción

El paquete `mycats` es el resultado de la necesidad de tener un conjunto de definiciones que se usan frecuentemente en mis notas. Como tal es sólo un conjunto de definiciones de comandos que intentan mantener consistencia en la escritura de conceptos de categorías y teoría de topos, además de tener una buena relación con el significado de los comandos.

2 Instalación

Para usar el paquete `mycats` se debe copiar el archivo `mycats.sty` en el directorio de trabajo. Esto obliga a tener una copia del archivo en cada proyecto. Para evitar esto, se puede copiar en el directorio `texmf`. La ubicación de este directorio se puede encontrar escribiendo en la terminal

```
kpsewhich -var-value=TEXMFHOME
```

En mi caso, la ubicación es `/home/luis/texmf`. Soi no existe la carpeta `texmf`, se debe crear. Dentro de esta se debe crear una carpeta llamada `tex` y dentro de esta una carpeta llamada `latex`. Finalmente, dentro de la carpeta `latex` se debe crear una carpeta llamada `mycats` y dentro de esta se debe copiar el archivo `mycats.sty`.

Con el segundo método se puede usar el paquete en cualquier proyecto sin tener que copiar el archivo `mycats.sty` en cada proyecto. En cualquiera de los dos métodos el paquete se carga con el comando

```
\usepackage{mycats}
```

3 Comandos

3.1. Objetos de un símbolo

El paquete `mycats` define una notación para escribir categorías y topos. Hay una definición por cada letra mayúscula. Por ejemplo,

```
\begin{gather*}
  \catA, \catB, \catX, \dots\\
  \toposE, \toposF, \toposG, \dots
\end{gather*}
```

$$A, B, X, \dots$$
$$\mathcal{E}, \mathcal{F}, \mathcal{G}, \dots$$

Es posible definir objetos de un sólo símbolo de manera simple. Para esto se debe usar

\defset{⟨\comando⟩}{⟨*nombre*⟩}{⟨*lista*⟩}

Crea un conjunto de comandos, uno por cada elemento de la ⟨*lista*⟩. El ⟨\comando⟩ es un comando de LaTeX que se toma un argumento obligatorio. Por ejemplo, \vec. El ⟨*nombre*⟩ es el nombre del comando que se va a crear y debe terminar con #1. Por ejemplo, vec#1. La ⟨*lista*⟩ es una lista de elementos separados por comas

Para hacer las definiciones de categorías, por ejemplo, se usó el comando

```
\defset{\symsfit}{cat#1}%
{A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}
```

De la misma manera se pueden hacer conjuntos de definiciones de comandos para objetos de un sólo símbolo, por ejemplo

```
\defset{\vec}{vec#1}{u,v,w,x,y}
```

define cinco vectores. En este caso obtenemos

```
\(\vecu,\vecv,\vecw,\vecx,\vecy\)
```

$$\vec{u}, \vec{v}, \vec{w}, \vec{x}, \vec{y}$$

En realidad el comando \defset es un comando que puede definir un conjunto de comandos sin importar el número de caracteres que tenga el comando. Por ejemplo, si hacemos la siguiente definición de comandos

```
\defset{\widehat}{wh#1}{abcd,xyz}
```

entonces obtenemos dos comandos para escribir sombreros:

```
\(\whabcd \quad \whxyz\)
```

$$\widehat{abcd} \quad \widehat{xyz}$$

La desventaja de usar este comando para definir comandos de más de un símbolo es que la sintaxis es cada vez más grande y difícil de escribir.

3.2. Algunas categorías

Hay algunas categorías especiales que se usan frecuentemente. Para estas introducimos los siguientes comandos

```
\begin{equation*}
  \con,\est,\gru,\esv,\mo,\alg
\end{equation*}
```

$$Con, Top, Gru, Vec, Mod, Alg$$

Se tomó la decisión de definir estas categorías mediante comandos separados en lugar de usar el comando `\defset` ya que de esta manera la sintaxis del comando es más cercana a su salida en el pdf. Comparar, por ejemplo, `\catCon` con `\con`.

En el caso de espacios vectoriales, módulos y álgebras es más conciso escribir, por ejemplo, `k-\alg` para obtener $k - Alg$ ya que la sintaxis del código es más parecida a la salida esperada.[1]

Para obtener la opuesta de una categoría C hay que usar el comando `\op`. De esta forma tenemos

```
\(\catC\op\)
```

$$C^{op}$$

Otra categoría especial es la categoría de pregavillas. Para esta se ha preferido la notación de Grothendieck con un sombrero, ya que la notación como una exponencial es muy alta y suele echar a perder el interlineado del texto. Sin embargo, las dos notaciones están disponibles. Además, en ambos casos se puede "trucar".

```
\begin{alignat*}{3}
  &\pg{\catC} &\quad& \pg*{\catC} &\quad& \pg{\Delta}[1]\\
  &\pge{\catC} && \pge*{\catC} && \pge{\Delta}[1]
\end{alignat*}
```

$$\begin{alignat*}{3}
&\hat{C} && \widehat{C^{op}} && \hat{\Delta}_1\\
&Con^{C^{op}} && Con^{C} && Con^{\Delta_1^{op}}
\end{alignat*}$$

Además de las categorías de pregavillas, otra construcción son las categorías de gavillas. De nuevo, estas tienen dos notaciones comunes. La primera es cuando se toma una categoría y una topología de Grothendieck sobre esta. La otra es cuando se toma un topos elemental y una topología de Lawvere-Tierney. Como en algunos lugares también se usa la notación de topos elementales para los topos de Grothendieck, se ha preferido la notación de topos elementales.

```
\(\gav{\toposE,j}\) versus \(\gav*{\catC,J}\)
```

$$\mathscr{E}_j \text{ versus } \mathrm{Gav}(C,J)$$

También es común usar categorías rebanada. Para definir las rebanadas primero definimos un cociente

```
\[
  \quotient{A}{R} \quad
  \quotient{\prod\limits_{i\in I}X_i}{R}
\]
```

$$A/R \quad \prod_{i\in I} X_i \Big/ R$$

Luego, con este cociente definimos las rebanadas de la siguiente manera

```
\[
  \slice{\catC,A} \quad \slice{\toposE,B}
\]
```

$$C/A \quad \mathscr{E}/B$$

[1] Hacer, de forma que la sintaxis no sea compleja, que la raya sea irrompible para que no suceda que se rompa una línea en la raya de una k álgebra. Además hacerla un símbolo ordinario, no un operador

Definimos un comando para cocientes que hace crecer la diagonal del cociente de forma automática ya que en algunos casos se requiere hacer rebana de una rebanada. Con esta definición será evidente cuál es la primera rebanada y cual es la segunda.

```
\[
  \slice{\slice{\catC,A},B}
\]
```

$$C/A\Big/B$$

Otra construcción común es la categoría coma, definimos la categoría coma con el siguiente comando

```
\[
  \comma{F,\catC}
\]
```

$$F \downarrow C$$

3.3. Flechas

En algunos casos es útil tener un comando para escribir flechas como las de `tikz` en el texto. Para estas situaciones se ha definido el comando `\fun` que tiene la sintaxis

```
\fun[*][opciones]{dominio}[nombre]{codominio}[separación]
```

donde las `opciones` son opciones de `tikz` que modifican la flecha en cuestión. Algunos ejemplos son

```
\begin{gather*}
  \fun[tail]{A}{B}\\
  \fun[dashed]{A}[f]{B}[1em]\\
  \fun[hook]{A}[f]{B}\\
  \fun*{A}[f]{B}
\end{gather*}
```

$$A \rightarrowtail B$$
$$A \overset{f}{\dashrightarrow} B$$
$$A \overset{f}{\hookrightarrow} B$$
$$f : A \longrightarrow B$$

4 Algunos otros comandos

4.1. Operadores

El paquete `mycats` define algunos operadores que se usan frecuentemente en categorías y otras áreas de las matemáticas. Estos son

```
\(\dom(f),\cod(f),\id_{A},\im(f)\)
```

$\mathrm{dom}(f), \mathrm{cod}(f), \mathrm{id}_A, \mathrm{im}(f)$

4.2. Delimitadores

También hay algunos delimitadores que se usan frecuentemente. Estos son

```
\(\abs{x}\quad\norm{\vecv}\quad\pair{x,y}\quad
\nom{f}\quad\fami{A_i}_{i\in I}\quad\vb{\varphi}\)
```

$|x| \quad \|\vec{v}\| \quad (x,y) \quad \ulcorner f \urcorner \quad \langle A_i \rangle_{i\in I} \quad [\![\varphi]\!]$

A cada uno de los comandos para delimitadores se les puede aplicar un argumento opcional para ajustar el tamaño del delimitador. Por ejemplo,

```
\[\abs[\big]{x} \qquad \fami[\Big]{A_i}\]
```

$$|x| \qquad \left\langle A_i \right\rangle$$

Además, tienen una versión con un asterisco que ajusta el tamaño del delimitador al tamaño del argumento. Por ejemplo,

```
\[\abs*{\frac{1}{2}} \qquad \fami*{\sum_{i}^{n}x_i}\]
```

$$\left|\frac{1}{2}\right| \qquad \left\langle \sum_{i}^{n} x_i \right\rangle$$

Para hacer los delimitadores se usó el paquete `mathtools`. Como se explica en la sección 3.6 de la documentación de `mathtools`, los delimitadores pueden hacerse más complejos que nuestra simple definición. Un ejemplo, tomado del manual, es la definición para hacer conjuntos

```
\[\set{A_i\st i\in I}\qquad \set*{\sum_{i}^{n}i\st n\in\mathbb{N}}\]
```

$$\{A_i \mid i \in I\} \qquad \left\{ \sum_{i}^{n} i \,\middle|\, n \in \mathbb{N} \right\}$$

4.3. Otros

En LaTeX se debe usar `\colon` para obtener los dos puntos que separan al nombre de una función de su dominio. El espacio qe deja este comando es el de un símbolo de puntuación. Un error común es usar `:`. Los dos puntos en modo matemático están definidos como una relación y dejan el espacio que dejan las relaciones. Veamos una comparación del espacio que dejan

```
\begin{gather*}
  f\colon A\to B\\
  f: A\to B
\end{gather*}
```

$$f\colon A \to B$$
$$f : A \to B$$

Como se puede ver, el espacio que deja `\colon` es más pequeño que el que deja `:`.

Si se quiere escribir un *span* o un *cospan* en el texto, entonces se tendrán problemas de espacio:

```
\(a\colon 1\to A\leftarrow 1\colon b\)
```

$a\colon 1\to A\leftarrow 1\colon b$

Para solucionar esto se ha definido un comando que deja el espacio correcto en los dos puntos de la derecha, es decir, un `\colon` al revés

```
\(a\colon 1\to A\leftarrow 1\noloc b\)
```

$a\colon 1\to A\leftarrow 1\noloc b$