Tips para escribir papers de matemática en LaTeX

Eric Brandwein

6 de abril de 2025

Esta es una guía de las cosas que estuve aprendiendo mientras escribía papers de matemática en LaTeX. Obviamente va a seguir creciendo infinitamente, y no necesariamente va a estar bien organizada.

LaTeX no es lindo, ni bueno, ni muy comunicativo. Muchas veces te va a aparecer un error, y la única forma de solucionarlo va a ser o googleando, o no haciendo lo que querías hacer. Y podés estar horas al principio para cada pelotudez.[1] Solo sabé que esto es normal y no te desesperes.

Hay cosas que directamente no podés hacer con LaTeX, como por ejemplo agregar emojis. La alternativa es LuaTeX, pero los journals probablemente no te permitan usarlo. Así que por ahora no hay emojis en los papers :(

Esta guía no te va a explicar la sintaxis básica de LaTeX. Lo mejor para aprender eso son los tutoriales de Overleaf, por ejemplo `https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes`, o directamente ver el código fuente de esta guía en .

Poner link a github

[1] Typst trata de solucionar estas cosas, pero no es LaTeX. Ojalá todos nos pasásemos a Typst.

Índice

1. Herramientas

1.1. Distribución de LaTeX

Para compilar LaTeX necesitás una distribución. En Linux, la más común es TeX Live.

Si estás en ArchLinux o una distribución basada en Arch, podés usar `pacman` para instalarlo:

```
$ sudo pacman -S texlive
```

Aunque es mucho más probable que estés usando Ubuntu o una distribución basada en Ubuntu. En ese caso, podés usar `apt`:

```
$ sudo apt install texlive-full
```

Si no estás usando ninguna de estas distros, investigalo vos. Y si estás usando Mac o Windows, suerte, no sé nada yo.

1.2. Editor de texto

Yo recomiendo usar VSCode con la extensión LaTeX Workshop. Tiene muchas cosas copadas:

- Compila automáticamente el documento cada vez que guardás.
- Podés ver el PDF generado al lado del código, y podés saltar entre el PDF y el código con SyncTeX. Para saltar del código al PDF, es Ctrl + Alt + J, y al revés es Ctrl + clic izquierdo (no sé cómo es en Mac, esto es en Linux).
- Tiene syntax highlight, así sabés si estás en math mode o no.
- Te ayuda un poco con los errores; en general podés ir a Problems y te muestra dónde está el problema. Ver Fig. 1.
- etc. etc. etc.

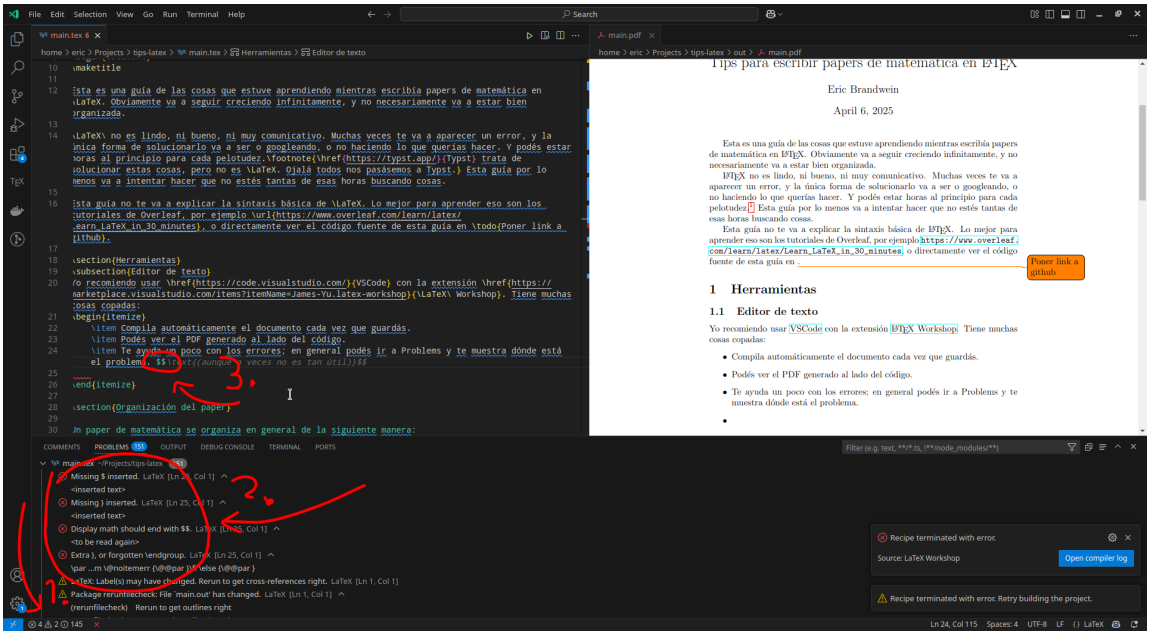

Figura 1: Errores en LaTeX Workshop. 1. Dónde cliquear para ver los problemas 2. Los mensajes de error 3. El problema en cuestión: empecé un bloque de matemática con dos $ y no lo cerré.

También agregarle la extensión LTeX que te corrige la gramática. Por default te corrige inglés; podés cambiarlo con la configuración `ltex.language` a `es-AR`.

Al VSCode le podés también agregar GitHub Copilot si te logueás a tu cuenta de GitHub. La verdad que ayuda más en código que escribiendo un paper, pero tiene un chat al que podés preguntarle cosas de LaTeX y a veces te responde bien.

También podés usar Overleaf, que es un editor online. Tiene la ventaja de que no tenés que instalar nada, y tiene bastante mejores mensajes de error. Lo malo es que tenés que pagar si querés colaborar con más de una persona creo, y no tenés todo lo que tenés con VSCode. Por ejemplo, si tu proyecto es muy grande, directamente no te deja compilar. Desgraciadamente, la mayoría de los matemáticos usan Overleaf, así que a veces vas a tener que soportarlo. Lo que hago yo es elegir la opción de descargar el código fuente comprimido, descomprimirlo, editarlo en VSCode, y después subir todos los archivos de nuevo al Overleaf.

1.3. Compilando el PDF

La extensión de VSCode y Overleaf te compilan solos el PDF. Si querés compilarlo en consola podés usar `pdflatex`, pero a veces tenés que correrlo muchas veces para que te ponga bien las referencias y otras cosas. Para no tener que hacer eso, lo mejor es usar `latexmk`:

```
$ latexmk -pdf main.tex
```

1.4. Git

Seguro vas a tener que colaborar con gente si querés avanzar en tu carrera paperística. Y sino, igual vas a escribir cosas, las vas a guardar, te vas a ir a dormir, te vas a despertar, te vas a dar cuenta que no te gustaron, y las vas a querer deshacer, y el Ctrl + Z no te va a servir. Por eso tenés que usar Git, y subir tus cosas a GitHub. Acá no voy a explicar cómo usar Git, pero si no lo usás y perdés cosas que hiciste y puteás, yo te avisé. El VSCode tiene Git integrado, así que supongo que esa es la forma más fácil de usarlo.

1.5. Zotero

Zotero está bueno para irse guardando los papers interesantes que te encuentres.

2. Cómo escribir el contenido

Una muy buena guía para escribir se encuentra en el blog de Terence Tao. Recomiendo fuertemente leer todos sus posts sobre cómo escribir papers. También recomiendo leer todas las cosas que cita al final, aunque yo no las leí todas.

Para saber lo que se estila a la hora de escribir un paper, lo mejor es leer otros papers. Sin embargo, la mayoría de los papers están escritos para el orto; los papers no están hechos para ser leídos, están hechos para que los resultados sean citados, así que tampoco te copies directamente de lo que leas. Pero está bueno tener una idea para elegir exactamente qué cosas no copiar.

El objetivo de tu paper es básicamente que te lo publiquen y te lo citen. Una forma de lograr eso es tener resultados importantes, pero si no los tenés, la otra forma es que al que te revise el paper le resulte fácil de leer (para que recomiende su publicación) y que al investigador que se encuentre con tu paper le resulte fácil de leer (para poder aportar en el mismo tema, y por ende citarte). Parece que tenés que escribir un paper que sea fácil de leer. Si tu objetivo no es que te lo publiquen y te lo citen, sino avanzar el conocimiento científico, con más razón querés escribir un paper que sea fácil de leer; si nadie lo puede leer nadie va a ganar mucho conocimiento.

No sé si alguna vez leíste un paper de matemática. Si no lo hiciste lo primero que tenés que hacer es leer uno. Si lo hiciste, seguro te diste cuenta que son *dificilísimos* de leer. Se hace un poco más fácil con el tiempo, pero nunca se vuelve fácil fácil. Siempre vas a leer muchas veces una definición nueva para entenderla, y te vas a olvidar qué significaba después de leer 2 otras.

Pero no tiene que ser tan así. Compará un paper con una charla, o una clase que te hayan dado, o un video, o un artículo de divulgación. ¿Cuáles son las diferencias?

- *El vocabulario.* Cuando te explican algo se usan palabras simples, y se repiten muchas veces los mismos conceptos para recordarlos. Copiate de esto.

- *El formalismo.* En general en charlas no te dan las cosas tan formales, matemáticamente hablando. Podés copiarte de esto dando antes de tus resultados una explicación informal de cómo lo vas a demostrar antes de dar la demostración formal.

- *La cantidad de imágenes.* En las buenas clases que me dieron, había una imagen por diapositiva más o menos. Es más, había *animaciones*. En PDF no se puede hacer eso, pero sí se pueden poner dibujitos. En mi opinión, las demostraciones deberían tener un dibujo por cada paso, *especialmente* las de grafos. Uno no piensa en grafos con palabras.

  Esto atenta contra la máxima cantidad de páginas que te dejan poner en algunas publicaciones. Mi recomendación es hacer los dibujos y sacarlos si no te entran. Siempre se pueden poner en la versión de arXiv.

- *La subestimación del lector.* Las mejores clases que tuve son en las que el profesor asumía que era un ignorante total. Cuando estés en duda, tirate para abajo en la dificultad. Explicá las cosas obvias, porque son obvias para vos, pero el que va a leer tu paper podría tranquilamente ser uno que está tratando de hacer su tesis de licenciatura y no sabe nada. Es más, seguro que leés de nuevo lo que escribiste en 6 meses y no entendés nada.

- *La longitud.* Las charlas son cortas, y tu paper va a ser largo. Por eso tenés que repartir tu paper en secciones y subsecciones.

- *La interactividad.* Lamentablemente el formato PDF no soporta responderte preguntas sobre el paper[2],

[2] Aunque NotebookLM sí.

o dejarte hacer ejercicios sobre el tema y decirte si están bien hechos.[3]

- *El sonido de la voz de alguien explicándote.* Esto no se puede agregar en un paper lamentablemente, y no sé cuánto ayuda.

Entonces, ¿cómo hacer un paper que sea fácil de leer?

- *No dejar ningún error gramatical.* Usá un grammar checker.
- *Usar palabras básicas.* Muy lindo lo de querer parecer inteligente, pero mientras tengas un mínimo de formalismo matemático, no hace falta usar palabras rebuscadas. Podés repetir palabras tranquilamente.
- *Recortar las frases.* Frases largas son más difíciles.
- *Cagarlo a imágenes.*
- *Recordar las definiciones que usaste o definiste hace mucho.* La gente se olvida.
- *Escribir pensando que lo va a leer el vos de hace 6 meses.*
- *Sin sorpresas.* El que esté leyendo un Lema o una definición tiene que saber por qué lo está leyendo. Tipo, ¿para qué resultado importante voy a necesitar esto que leo?

2.1. Organización del paper

Un paper de matemática se organiza en secciones, en general de la siguiente manera:

1. **Título, autores, afiliaciones, y funding.**
2. **Abstract o Resumen.**
3. **Introduction.**
4. **Preliminaries.**
5. **Results.** (No necesariamente con este título, y pueden ser muchas secciones)
6. **Conclusions.** (Opcional)
7. **Open problems.** (Opcional aunque no tanto)
8. **Acknowledgements.**

Paso a explicar qué poner en cada lugar.

[3] Si se implementasen las ideas de `https://willcrichton.net/notes/portable-epubs/` sería un mundo mejor.

Título, autores, afiliaciones, y funding En general en matemática van los autores en orden alfabético. Las afiliaciones sería la universidad o empresa a la que pertenece cada uno. El funding es la beca o el proyecto de investigación que pagó por que vos labures en el paper.

Abstract Acá ponés uno o dos párrafos explicando los resultados del paper, o sea, por qué alguien querría leer lo que escribiste. Esto decide si el que lo lea va a meterse a leer el resto del paper, así que es importante que sea claro y conciso.

- **Claro:** Tenés que definir las cosas que necesites para decir el resultado al que llegaste. Nada de hablar de una "solución consistente apropiada de nivel 2" sin definir todo eso. Preferentemente, no usar palabras cuya definición que no se pueda leer de otro paper. ¿Por qué es interesante una solución de ese tipo, si no se usa en ningún otro lado?
- **Conciso:** Creo que se entiende.

En el abstract a veces se pone también un poco de motivación. Tipo, "el resultado que presentamos avanza en la resolución de un problema abierto en el paper de Fulano" por ejemplo.

Introduction Acá ponés un poco de motivación, y decís qué es lo que vas a hacer en el paper. En general, se pone un par de párrafos hablando de los resultados previos en otros papers, y después decís qué es lo que vas a hacer vos. Tratá de llegar lo más rápido posible a tus resultados, y no los demuestres acá, eso va después.

Preliminaries Acá van las definiciones que vas a usar en todo el paper desde la primera sección. Por ejemplo, decir que los grafos que usamos son no dirigidos y no tienen múltiples aristas, etc. Parafrasear otros preliminaries está bueno, porque casi siempre son muy parecidos. *Ojo: copiarse exacto puede hacer que salten alarmas de plagio cuando lo subas a arXiv o a journals, así que no copies, parafraseá.*

Si hay resultados que se usan en una sola sección, ponerlos justo cuando se empiecen a usar, y no antes. La gente no va a leer los preliminaries, van a ir a tus teoremas y cuando no entiendan algo van a ir a buscar la definición, así que mejor que esté cerquita.

Results Esta parte del paper puede tener muchas secciones y subsecciones. Esto es lo más libre del paper. Acá vas a tener tus resultados en forma de:

- *Theorems* para los resultados principales de tu paper, los que ponés en el abstract.
- *Lemmas* para resultados que necesitás para tus teoremas.
- *Corollaries* para resultados que se deducen de los teoremas o lemas.
- *Definitions* para definiciones.

- *Claims* lo uso para demostraciones adentro de demostraciones.
- *Observations* para resultados que son tan fáciles que no necesitan demostración.
- *Propositions* que no sé para qué se usan, yo no uso.

Conclusions No es tan usado en matemática, pero si tenés experimentos de algún tipo podés sacar alguna conclusión. Podés también resumir tus resultados en una tabla o algo así.

Open problems Siempre está bueno poner problemas abiertos que surgen de tu trabajo, porque así ayudás a que tu paper tenga más citas, y eso es todo lo que queremos.

2.2. Estrategia general de escritura

La recomendación general es: *¡No te trabes!* Si no te sale escribir algo, escribilo así nomás. Igual vas a tener que releer y reescribir un montón. Por la misma razón, no te podés encariñar con algo que escribiste. Dejalo ir. Casi siempre cuando lo reescribas va a salir mejor.

En particular, me gusta seguir más o menos este orden:

1. Escribir los statements de los resultados.
2. Escribir una guía en Español de cada demostración.
3. Identificar qué lemas voy a necesitar para la demo, y escribir los statements.
4. Repetir Items 2 and 3 hasta que estén todos los lemas que necesito.
5. Escribir las definiciones en Inglés.
6. Escribir las demostraciones en Inglés.
7. Escribir el texto informal que rodea a los teoremas y ayuda a entenderlos. Incluir ejemplos de aplicación de definiciones y teoremas.
8. Agregar muchas imágenes que ayuden a entender todo.
9. Escribir otras secciones.
10. Escribir la Introducción.
11. Escribir el Abstract.

En el medio, vas a tener que releer todo y reescribir y borrar muchas cosas. Si usás Git está bueno porque no perdés lo que borraste. No tengas miedo de borrar cosas, seguro estaba escrito para el orto y ahora con las ideas claras lo escribís mucho mejor.

El orden no es estricto para nada. Siempre seguí el consejo de no trabarte. Si no tenés ganas de escribir una parte, seguí con otra, y dejá una notita al costado para acordarte de lo que tenías que hacer.

Es necesario que escribas el paper en muchos días diferentes, y que releas lo que escribiste días después. Así vas a saber si se entiende o no. Mientras releas, preguntate: "¿por qué se cumple esto que dije acá en esta demo?". Si la explicación que te das a vos mismo no está escrita, escribila. Si implica dibujar algo, dibujalo en el paper. Recordá lo tonto que va a ser tu lector.

Tratá de tener muchos lemas, dividiendo las demostraciones de los teoremas en cachitos.

3. Usando LaTeX

LaTeX funciona con paquetes que se agregan con `\usepackage`. Una distribución completa debería tener todos los paquetes de CTAN. Si tenés dudas de cómo usar un paquete, en esa página va a estar la documentación, aunque a veces es más fácil buscar en internet cómo se usa.

El código fuente de esta guía está en . Copiate de los paquetes que uso acá para lo que necesites.

Poner link a github

También podés copiarte de lo que hacen otros papers. En arXiv te podés descargar el código fuente del paper que quieras. Ver Fig. 2.

3.1. Citar otros papers

Hay un montón de paquetes para citar papers en latex. El que más me gusta es `biblatex`. Leer `https://www.overleaf.com/learn/latex/Bibliography_management_in_LaTeX` para una descripción por arriba.

Cosas que descubrí por mi cuenta de `biblatex`:

- En vez de `\cite` podés usar `\textcite`, que te pone los nombres de los autores y la cita.
- Podés citar un teorema en particular de un paper con `\cite[Theorem 1]{paper}`, y te pone "[Paper, Theorem 1]". *Siempre poné el número de teorema o de página de donde citás cosas para que sea fácil de encontrar.*

Para armar la bibliografía, tenés que crear un archivo `.bib` y ponerle el nombre que quieras. Yo siempre lo llamo `references.bib`. Para llenarlo de entradas, podés sacar las citas de Google Scholar, o de DBLP si estás haciendo computación. Ojo que hay una diferencia entre citar un paper de un journal y un preprint de arXiv, esto se explica más en Section 4.

No tengas miedo de ponerles nombres largos a las entradas, nadie lo va a ver, y los editores de texto te autocompletan los nombres en general.

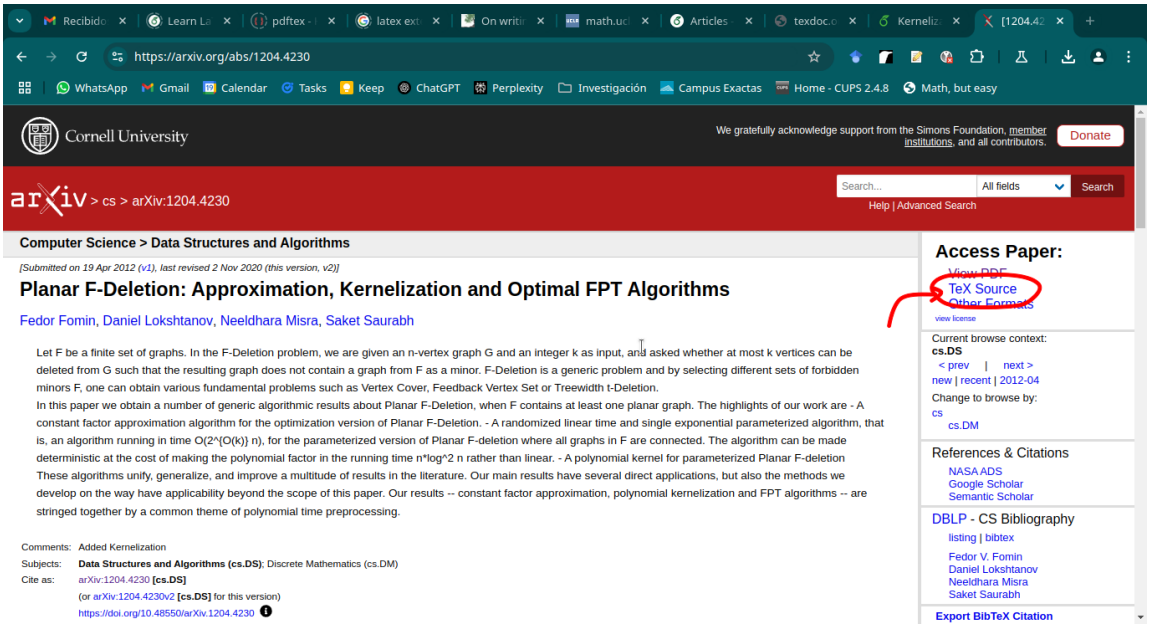

Figura 2: Código fuente de un paper en arXiv.

3.2. Teoremas y definiciones

Leer `https://www.overleaf.com/learn/latex/Theorems_and_proofs`. Los teoremas y los lemas y todo eso deberían todos seguir la misma numeración. Nada de tener una Definition 1 y un Theorem 1 cinco páginas después, eso no ayuda a la búsqueda binaria. Si te descargás el paquete que está en `https://github.com/ericbrandwein/latex-packages/tree/main/theoremdefs`, ya te lo resuelve.

La primera vez que escribas el nombre de una cosa que estás definiendo, ponelo en cursiva. Tipo "Los *números naturales* son los números enteros no negativos". Después de eso, ya no hace falta ponerlo en cursiva.

3.3. Referenciar cosas adentro de tu paper

Cada vez que digas "Esto se cumple por Lemma 5", vas a tener que poner un link al Lemma 5. Esto se hace agregando los siguientes paquetes EN ESTE ORDEN EXACTO (sino se rompen cosas):

```
\usepackage{hyperref}
\usepackage{varioref}
\usepackage{amsthm} % De paso te traés el entorno proof.
\usepackage[capitalise]{cleveref}
```

Después, en donde escribís el lema, ponés

```
\begin{lemma}[Teorema de eriquito]
   \label{fact:teorema-de-eriquito}
   $1+1 = 2$.
\end{lemma}
```

y donde lo citás ponés `\cref{fact:teorema-de-eriquito}` y te va a poner "Lemma 5". Si querés que te ponga "Lemma 5 de la página 23", le ponés `\vref{fact:teorema-de-eriquito}`.

Notar que empiezo el label de los teoremas y lemmas con `fact:`. Algunos diferencian entre teoremas y lemas con `thm:` y `lem:`. Yo prefiero usar `fact:` para toda verdad matemática, porque quizá después pasa a ser un teorema, o un corolario, o lo que sea, y no tengo ganas de cambiar en todos lados el label. Total `\cref` te lo pone bien siempre. Ahora, para definiciones, me gusta usar `def:`, porque ahí sí son cosas diferentes. Para figuras `fig:`, para tablas `table:`, etc.

No importa que los labels sean largos, porque el editor te los autocompleta y el lector no los va a ver.

Ahora, cuando yo leo "Lemma 5", no tengo ni puta idea de lo que dice el Lemma 5. Tengo que cliquear el link para acordarme cuál era el lema y después volver a lo que estaba leyendo. En cambio, estaría bueno que en vez de leer "Lemma 5", lea "Teorema de eriquito (Lemma 5)". Eso es exactamente lo que hace un paquete re pavo que hice que está en `https://github.com/ericbrandwein/latex-packages/tree/main/namedtheoremref`. También vas a necesitar descargar el paquete en `https://github.com/ericbrandwein/latex-packages/tree/main/theoremdefs`. Tenés que descargar los `.sty` y ponerlos en la misma carpeta que tu `.tex`. Después, lo agregás al preámbulo de tu `.tex` con:

```
\usepackage{namedtheoremref}
```

y ponés `\namedtheoremref{fact:teorema-de-eriquito}` y te va a poner "Teorema de eriquito (Lemma 5)" con el link al lema y todo.

Con el mismo paquete podés referenciar definiciones por nombre, tipo "the definition of eriquito (Definition 1)" para referenciar una definition escrita así:

```
\begin{definition}[eriquito]
   \label{def:eriquito}
   \emph{Eriquito} es un doctorando en computación.
\end{definition}
```

3.4. Imágenes

Leer `https://www.overleaf.com/learn/latex/Inserting_Images`. En general vas a insertar imágenes en un entorno `figure`.

Para crear las imágenes algo que se usa mucho es TikZ. Yo no lo recomiendo. Tardás mucho tiempo en aprenderlo, no hace todo lo que querrías, y hacer una imagen nueva tarda horas, incluso para gente re

curtida. Yo primero me pasé a usar Ipe, que es muy fácil de usar, pero tampoco tiene todo lo que quería, así que terminé usando Inkscape con el plugin TexText.

3.5. Fórmulas

Todos conocemos el `$...$` y el `\[...\]`. Pero se puede hacer más que eso. Leer `https://www.overleaf.com/learn/latex/Aligning_equations_with_amsmath`.

Un detalle es que las fórmulas son parte del texto, así que hay que ponerles un punto final si están al final de la oración. Por ejemplo:

$$1 + 1 = 2.$$

Lo mismo pasa con la coma:

$$1 + 2 = 3,$$

y la tenés que poner adentro de la fórmula, porque sino te queda abajo y queda feo.

Muchas veces vas a querer escribir algo en una fórmula y no vas a saber cómo. Si hay un símbolo que no sabés cómo escribir, lo mejor es buscarlo en Detexify.

No empieces una frase con una fórmula, y separá las fórmulas poniéndoles texto en el medio. En vez de "For every graph G, $E(G) \subseteq 2^{V(G)}$.", escribí "For every graph G, we have that $E(G) \subseteq 2^{V(G)}$.", o algo así.

3.6. Comandos

A veces vas a usar una letra para un objeto matemático que no te gusta, y la vas a querer cambiar después de haber escrito 10 páginas de tu paper. Por eso tenés que aprender a usar comandos. Básicamente lo que hace LaTeX es reemplazar el comando por lo que escribiste.

Por ejemplo, si tenés un grafo G y un subgrafo inducido H de G, podés hacer dos comandos:

```
\newcommand{\graph}{G}
\newcommand{\inducedSubgraph}{H}
```

y usarlos así:

```
For every graph $\graph$, there exists an induced
subgraph $\inducedSubgraph$ of $\graph$.
```

Incluso, mientras estés escribiendo el paper, podés hacer que el comando `\inducedSubgraph` aparezca como `miGrafoInducido` o algo más declarativo, así es más fácil de leer. Incluso lo podrías dejar así, aunque no muchos van a estar de acuerdo. Si te gusta más H que `miGrafoInducido`, de vez en cuando está bueno hacer acordar al lector que H es un subgrafo inducido, diciendo por ejemplo "The induced subgraph H has at most k vertices such that...".

No le tengas miedo a ponerles nombres largos a tus objetos y funciones. Si un número cuenta la cantidad de vértices que cumplen tal propiedad, a mi gusto es mejor `niceVertices`(G) que $\alpha(G)$.

3.7. Múltiples archivos

Lo mejor es no tener todo el texto en un solo lugar. Especialmente porque muy posiblemente tengas muchas versiones de un mismo paper: una para el journal, otra para un congreso, otra para arXiv, etc. Vas a querer reusar el mismo texto para todas estas versiones, y no tener que estar copiando y pegando. Para eso, podés usar el paquete `import`. Leer `https://www.overleaf.com/learn/latex/Management_in_a_large_project`.

3.8. Otros paquetes

- `todonotes` – Para poner notas al lado del paper. Tipo así.
- `paireddelimiters` – Delimitadores comunes, tipo `$\set{G \mid \abs{V(G)} > 2}$` se ve tipo $\{G \mid |V(G)| > 2\}$. Para que se vea bien cuando tenés cosas grandes adentro, hay que poner un `*`. Ver diferencias entre estas dos fórmulas:

$$f(\sum_{i=1}^{20} i^2)$$
$$f\left(\sum_{i=1}^{20} i^2\right)$$

- geometry – Para cambiar los márgenes y el tamaño de página. No se usa mucho en journals porque ya te lo definen ellos. Leer `https://www.overleaf.com/learn/latex/Page_size_and_margins`.
- Muchos más, no voy a poner todos.

4. Cómo publicar

¡Felicitaciones! Escribiste tu paper y ahora querés que el mundo lo vea. Para eso tenés que mandarlo a una *revista* (*journal*) para que ellos llamen a otros investigadores que lo revisen (haciendo *peer review*) y decidan si recomiendan publicar tu paper en el journal o no. Esto va a tardar unos meses entre que lo enviás, los reviewers te mandan correcciones, las corregís, y se repite esto hasta que o te lo aceptan o te lo rechazan.

Hay algunos journals que tienen más *impacto* que otros, medido por ejemplo en la cantidad de citas que tiene uno de sus papers en promedio. Cuanto mayor es el impacto, más difícil es publicar ahí, y más prestigio (CV) te da publicarlo ahí. Cuanto más CV, mejores chances de que te admitan para becas y para posiciones académicas en el futuro.

Los journals de mayor impacto dependen de cada área. Probablemente tu director de tesis te pueda ayudar a elegir uno.

Como los journals tardan un montón, en computación lo que se hace mucho es publicar en un congreso antes de publicar en un journal. En estos vos mandás una versión reducida de tu paper (un *extended abstract*), y si te lo aceptan, lo presentás en una charla en el congreso. A veces si no te entran las demostraciones de los resultados podés agregarlas a un apéndice con `\appendix`. Tenés que pagar para ir al congreso aunque vos presentes algo, y alguien te tiene que pagar el costo de los pasajes y la estadía si es en otro país.

También lo que hacen todos es publicar una versión preliminar (un *preprint*) a arXiv. Casi cualquiera puede subir cualquier cosa a arXiv, así que lo mejor es citar de un journal si está disponible, que por lo menos se supone que están un poco más revisados.

Cada journal va a tener un template de LaTeX que vas a tener que descargar y llenar. También van a haber instrucciones para los autores de cómo escribir el paper.

Las políticas de publicación de papers en journals, congresos y arXiv pueden ser complicadas. En general, mandar un paper a un journal te prohíbe mandar a otro journal hasta que el primer journal te rechace el paper. Para saber si podés mandar a un journal al mismo tiempo que mandás a un congreso tenés que leer las condiciones de cada uno a ver si son compatibles. Por otro lado, en general no podés subir a arXiv la versión publicada en un journal. Lo que sí podés hacer en general es subir la versión con las últimas correcciones que hiciste antes de que lo publiquen, que obviamente va a ser casi lo mismo.

Si avanzás un poco más en tu carrera quizá te piden que vos revises un paper de alguien más. Tené en cuenta que no te van a pagar ni te van a dar crédito por hacerlo, porque es anónimo. Además, vas a tardar unos días en leerlo y en entenderlo. Lo que ganás es que tenés acceso más rápido a nuevos resultados, aunque como dije, todos primero publican en arXiv, así que esa ventaja es leve. Si tu puesto es de profesor, probablemente te lo pidan como parte de tu trabajo. Si no, no es necesario que lo hagas, pero una buena forma de aprender a escribir papers es corregir a alguien más.

Y sí, los journals te cobran por descargar los papers, y los que revisan no ganan un peso, y los autores tampoco. Por eso en un momento generaban más ingreso que Google, Apple, y Amazon (y no sé si lo siguen haciendo). Te ofrecen publicar sin que el que lo lea tenga que pagar, pero para eso tenés que pagar vos. Y no es 10 pesos que tenés que pagar; es miles de dólares por paper. Todo por el *prestigio* de la revista. No sé si se nota el odio que le tengo al sistema científico actual, enganchado al pasado cuando los papers se imprimían, no queriendo soltar el chupete. Pero así es la situación actual.