# Qué es una demo?

- De qué carreras son? Quién es de compu? Quién es de datos? Quién es de otra carrera, y de qué carrera son?
- Quiénes cursaron álgebra? Quiénes cursaron algo3?
- Quiénes se sienten seguros de poder escribir una demo suficientemente formal para publicar en un paper?

Primero tenemos que definir lo que es una demostración.

- Tírenme ideas de lo que podría ser una demo:
  1. "Una forma de convencer a alguien de una verdad matemática"
  2. "Una secuencia de aplicación de teoremas y axiomas"
  3. etc?

A mí me gusta más la segunda, porque puedo convencer a alguien de algo que no sea verdad. Pero pará, si una demo es una secuencia de llamados a teoremas y axiomas, cómo puede ser que las escribamos en español? Bueno, lo que escribimos es un *boceto* de la demo, no la demo en sí. La demo propiamente dicha es un algoritmo. No lo digo en el sentido figurativo; es literalmente un algoritmo; lo podés programar en el lenguaje de unas cosas que se llaman "interactive theorem provers" y correrlos en tu compu para ver si la demo vale.

Veamos un ejemplo bien pavo. Queremos probar $(a + 0) + c = a + (c + 0)$. En general vamos a empezar pensando las ideas de la demo en un lenguaje que a nuestro cerebro le guste. Quizá lo primero que se nos ocurre es:

**Demostración 1.** asdasdasdasdasd asdasd asd asd

Cuando escribís una demo en lenguaje natural, estás escribiendo un pseudocódigo, que se debería poder convertir en

Mi definición de demo incluye los métodos de inducción de la lógica de primer orden, tipo modus ponens y cosas. También debería incluir todas las demos escritas en otros lenguajes para los que exista una función de conversión entre lenguajes. Todo esto para decir: . Ese código debería venir un matemático y poder convertirlo a código real, que sería por ejemplo convertir todas las afirmaciones en fórmulas lógicas y que una sea el resultado de aplicar un cambio sintáctico a las anteriores. Por ej, para probar podríamos hacer

1. (a+0)+c = a + (c + 0)
1. <=> 2. (a)+c = a + (c + 0) -> por axioma de la suma de 0
2. <=> 3. a + c = a + (c + 0) -> por regla de paréntesis extra
3. <=> 4. a + c = a + (c) -> por axioma de la suma de 0
4. <=> 5. a + c = a + c -> por regla de paréntesis extra
5. <=> 6. True -> por identidad de términos

o podríamos hacer

Teorema: La suma de un número a y cero, sumado a un número c, es equivalente a la suma de a con la suma de c y cero. Demo: Nótese que `a` sumado a `c` es equivalente a sí mismo. Luego, por axioma de suma de cero, podemos sumar un cero a la a de la izquierda, y sumar un cero a la c de la izquierda, y obtener el enunciado del teorema. QED

La idea es que la demo en lenguaje natural, para ser una demo, se debería poder traducir al otro lenguaje con formulitas, y en general a cualquier otro lenguaje formal. Nótese que en la demo en lenguaje natural no puse absolutamente todas las reglas y axiomas que usé. Parte de la función de

transformación entre lenguajes puede hacer la inferencia de cómo rellenar algunas partes de la demo para que sea “rigurosa”. Otra demo posible sería:

Demo: Vale por propiedades básicas de la aritmética.

Y de nuevo, un matemático podría traducir esto a la demo de arriba con formulitas, pero básicamente la tuvo que escribir él mismo, porque la última demo sólo dio una ayudita. Pueden ver que hay una diferencia en la cantidad de pasos que mencioné que hay que hacer en los dos ejemplos de demos en español. Podríamos decir que la primera es más “rigurosa” que la otra, y quizá hasta podríamos definirlo como la complejidad temporal del algoritmo que convierta de un formato al otro. Cuanto más rigurosa la demo, menos baches tiene que rellenar el algoritmo, y entonces va a ir más rápido.

Ahora que definimos qué podría decirse como “rigurosa”, viene la pregunta: cuán rigurosa tiene que ser una demo escrita por nosotros? No les puedo dar la respuesta general. Hay áreas de la matemática que piden mayor rigurosidad que otras. Algo 2 pide más rigurosidad que Algo 3. Pero cualquier demo de correctitud de un algoritmo en Algo 3 se tiene que poder convertir en una demo de Algo 2, porque sino no sería una demo. Y cualquier demo de grafos se tiene que poder convertir en aplicación de teoremas y axiomas sobre grafos, que en realidad terminan siendo todos teoremas y axiomas de conjuntos, porque los grafos no son nada más que un par de conjuntos con algunas restricciones.

Lo que sí les puedo dar es recomendaciones. Mi recomendación es: ante la duda, formalicen. Ya saben los axiomas de conjuntos de Algebra I, y si no los saben tienen la intuición. Lo mismo con funciones. Además ya saben los teoremas y definiciones. Me pueden decir la definición de máximo de un conjunto? La definición de secuencia?

Otra recomendación: hay algunas cosas que tienen formas de decirse bastante estandarizada. Usen esas formas. “Sea X un conjunto” “Si pasa tal cosa, entonces por tal y tal pasa tal otra”. Pueden dar más ejemplos? (inducción, contraejemplo, etc.)

El mundo de la intuición de un matemático también podemos considerarlo como otro “lenguaje”. Lo que hace un matemático en realidad para creerse una demo muchas veces es pasar del español que escribiste vos a su mundo de ideas, con una función de conversión, un compilador ponele. Queremos que al compilador que tenga él en la cabeza le cueste poco hacer la conversión, y le va mejor cuando usamos términos con los que ya asoció una idea antes, como los que dijeron recién. A veces, no usar esos términos hace que su compilador no sepa qué hacer y tire error, explayado completamente con tres signos de pregunta y una M en tu parcial.

Otra: desambigüen. Usan un nuevo término que no tiene una definición estándar, como “llegar de acá a acá” o “Carle gana plata cuando hace tal cosa”? Definanlo en algo que sí tiene definiciñón estándar. “LLegar de acá a acá” podría ser una secuencia de pasos que cumpla tal y tal condiciñón. Habría que definir “pasos” también. Otro ejemplo?

Una vez que lo definieron, ahora pueden usar con mucha más confianza los teoremas e intuiciones de conjuntos que tenemos todos en la cabeza, porque al compilador le ahorraste el problema de encontrar una definición que enganche bien con tus palabras. Es tipo cuando hacés inferencia de tipos en algún lenguaje tipo Haskell o Kotlin o rust o cualquiera moderno tipado.

- transformar una demo hablada en una secuencia de aplicación de teoremas y axiomas
  - separar en el pizarron? entre cosas que escribimos y cosas que asumimos obvias
  - los errores en papers es porque esto no se hace nunca
  - no es que tienen que hacer esto siempre, pero eso porque en un momento confían tanto en su intuición que no necesitan bajar más que eso. Por ej intuición de cosas conexas en grafos -> ejemplo de poder separar max k componentes conexas de un vertice usando k vertices

- ej para la clase: agarrar una frase de alguna demo de cubawiki y hacer lo mismo que hicimos recien. Pueden venir a consultarme axiomas que no sepan

- mostrar un programa (quiza de la demo que muestre) escrita en un theorem prover
  - mencionar que al que le interese esta la optativa de pablo barenbaum
  - mencionar el natural numbers game de lean

- 3 tipos de errores de demos:
  1. Deducir algo falso
  2. Saltearse un paso "obvio" (o no tan obvio)
  3. Que no se entienda nada, i.e. errores de escritura (ambiguedad, gramática, caligrafía, palabras raras o frases largas, pocas palabras para algo complicado)

- crear una pagina en cubawiki para el taller? Y que suban sus correcciones y demos bien escritas?

Bibliografia util: https://users.metu.edu.tr/serge/courses/111-2011/textbook-math111.pdf

https://longformmath.com/proofs-book/