La complejidad de la computación

Ciro Iván García López

Enero 2021

Introducción

La noción de computación es innata al ser humano y está ligada al avance e historia del mismo. Son varios los ejemplos de procesos (construcciones con regla y compás), máquinas (ábacos y máquinas de Babbage) e ideas (máquinas de Turing) que abrieron la discusión y sentaron bases la noción de *computación*, no obstante el estudio formal de ésta es reciente, relativamente lento y dificultoso.

En 1910 David Hilbert pregunta a la comunidad científica por una formalización del concepto de *algoritmo*, en respuesta surgen independientemente tres formalizaciones: las máquinas de Turing (Turing), las funciones parciales recursivas (Gödel) y el cálculo λ (Church); éstos son conocidos como *modelos clásicos de computación*. Posteriormente, Kleene [Kle36] demuestra que dichos modelos son equivalentes y de ahí la llamada Tesis de Turing.

Tesis 1. *Todo lo computable se puede calcular usando una máquina de Turing.*

La tesis ha generado un debate intenso y se han expuesto argumentos en contra de su validez [Coo03, EGW04]. Por otro lado, el surgimiento de las computadoras digitales permitió el diseño y construcción de maquinas interactivas que sobrepasan los límites de los modelos clásicos.

El poco conocimiento que se tiene de los fenómenos computacionales motiva su estudio y ha permitido que sea un proceso continuo, en donde se rebaten o repiensan algunas de las ideas desarrolladas y es gracias a este dialogo continuo en donde vuelven a emerger inquietudes como:

- ¿Por qué es relevante la computación para el hombre?
- ¿Qué es computación y ser computable?
- ¿Cuál es la relación de la computación y las matemáticas?

El objetivo de este documento es abrir la discusión sobre algunos de estas preguntas, naturalmente no se puede garantizar una respuesta absoluta atendiendo a que la computación esta sometida a un proceso de construcción continua.

Estas reflexiones han sido el resultado del curso sobre Complejidad Computacional impartido por el Doctor Gilberto Calvillo en la Universidad Nacional Autónoma de México durante el segundo semestre de 2020.

1. Lo computable; una necesidad

A lo largo de la historia de las civilizaciones se encuentran referencias que documentan la necesidad de automatizar procesos y construir máquinas que permitan al ser humano *calcular*; si por un momento se reflexiona el Teorema de Pitágoras este puede ser entendido como un método que automatiza la tarea de encontrar la menor distancia entre dos puntos. A su vez en algunos otros casos, como el ábaco, es mucho más evidente la emergencia de procesos computables.

Tal vez uno de los elementos que más une al ser humano con la noción de computable se da mediante el cerebro, órgano capaz de realizar cómputos a una velocidad y con una precisión alta. Gracias a éste hemos sido capaces de abstraer el mundo, crear representaciones para los objetos que nos rodean y computar funciones en el sentido de transformar cierto tipo de datos en otros, lo que nos lleva a concluir que biológicamente somos seres computacionales.

Por lo que es razonable considerar un modelo matemático que capture esta esencia del ser humano y preguntarse *¿qué representa el cerebro, es una máquina?* Este tipo de preguntas han sido estudiadas con anterioridad, un artículo notable escrito por David King [Kin96] expone argumentos para afirmar que el cerebro traspasa computacionalmente a los modelos clásicos, uno de dichos argumentos es la incapacidad de las máquinas de trabajar con la noción de infinito.

Un punto de inflexión y que cambia el curso de la historia es el surgimiento de la Internet y esto no habría sido posible de no ser por el surgimiento de las computadoras digitales. Las redes han permitido que la computación sea apropiada por los seres humanos al punto de hoy en día contar con máquinas que apoyan nuestro día a día y se presentan en diferentes formatos, computadoras, celulares, dispositivos inteligentes como cafeteras, hornos, etc, y están presentes en casi cualquier lugar del mundo.

A su vez el uso continuo de dispositivos motiva la búsqueda de respuestas de manera inmediata, es decir optimizar los tiempos de ejecución. La búsqueda estas respuestas ha permitido refinar la comprensión de lo computable, en particular ha permitido avanzar en la compresión de los algoritmos, los cuales a su vez han mejorado gracias a herramientas computacionales. En el corazón de estas búsquedas se encuentra la necesidad *optimizar* el tiempo finito del hombre en la tierra, por lo que entonces se puede afirmar que el hombre tiene una autentica necesidad de computar.

2. La computación: ciencia o apéndice

Hablar de la computación como ciencia nos remite a la historia; al revisar libros que tratan la historia de las matemáticas [Bal60] se pueden encontrar referencias de fenómenos o dispositivos que hoy en día se podrían llamar computacionales: construcciones regla y compás (griegos), construcción de pirámides (egipcios), ábaco (chinos), la máquina de engranajes de Charles Babage o la máquina de sumas de Freiderich Leibniz, dado lo cual, no es posible o claro el punto en el tiempo donde podamos situar un origen para el inicio del estudio de la computación.

A pesar de la originalidad y brillantez de las ideas o máquinas concebidas, en ese momento de la humanidad no logró florecer la idea de computación [Rob15] y no es sino hasta el siglo XX en donde el concepto de computación logra un gran impuso y pasa a ser un campo de estudio. El impulso inicial se debe a David Hilbert y su conferencia de 1910, en dicha conferencia Hilbert propone el problema de decidir si dado un conjunto de axiomas y reglas, Ω, existe un proceso automático que para cualquier proposición φ permitiera decidir si era derivable en Ω, no obstante en ese momento la matemática no contaba con herramientas para responder, pues no había una noción formal de *proceso automático*.

Formalizar el concepto de proceso automático o *algoritmo* no ha sido trivial y requiere contemplar otras nociones que surgen como máquina computacional o cómputo. En un primer intento de formalización surgen, independientemente, tres teorías que buscaban aclarar lo que significa un algoritmo[1]:

- Alan Turing y las máquinas de Turing [Tur37] : se diseñan maquinas abstractas que constan de 3 partes, una cinta infinita, un cabezal y una función de transición. La máquina conoce su posición sobre la cinta gracias al cabezal y es capaz de leer o escribir sobre la cinta, su comportamiento está determinado por la función de transición. Un algoritmo viene representado por una máquina de Turing, un computo es una aplicación de la función de transición y se dice que algo es Turing computable si existe una máquina de Turing que lo genera como salida.

- Kurt Gödel y las funciones parciales recursivas [Gö31] : se parte de considerar que las funciones cero, sucesor y proyecciones son algoritmos, esta suposición es razonable atendiendo la sencillez para evaluar dichas funciones. A partir de ellas se construyen algunas otras funciones por medio de composición, recursión primitiva y μ-minimización. En esta formalización, los algoritmos son representados por funciones, los cómputos son representados por la evaluación de una función y lo recursivamente computable es aquello que pueda ser producido por una función parcial recursiva.

- Alonso Church y el cálculo Lambda [Chu36] : la idea en este modelo es construir cadenas de símbolos que representan las funciones, para ello se definen los términos λ (λ-T) que están producidos por la siguiente gramática:

[1] La exposición de los modelos clásicos de computación omite algunos detalles, para una mejor presentación de los temas se puede recurrir a textos como [Sip12, HS08].

```cpp
int power(int a, unsigned int n){
    int res = 1;
    while (n > 0){
        if (n & 1){
            res = (res*a);
        }
        n = n>>1;
        a = (a*a);
    }
    return res;
}
```

```cpp
int power(int a, unsigned int n){
    int res = 1;
    while (n > 0){
        ret = ret*a;
        n = n-1;
    }
    return res;
}
```

Cuadro 1: Dos codificaciones en el mismo lenguaje, C++.

$$\langle \lambda\text{-T} \rangle \to \langle variable \rangle \;\; \| \;\; \lambda x.\langle \lambda\text{-T} \rangle \;\; \| \;\; (\langle \lambda\text{-T} \rangle)(\langle \lambda\text{-T} \rangle)$$

En este caso, los algoritmos son términos λ, los cómputos están dados por simplificaciones a los términos o β-reducciones y lo λ computable es aquello para lo cual podemos encontrar un término λ que se reduzca a lo deseado.

A pesar de la naturaleza distinta de cada una de los modelos clásicos, Kleene [Kle36] logra demostrar que son equivalentes, en otras palabras son capaces de computar lo mismo. Esta equivalencia permitió introducir la siguiente definición.

Definición 1. Una función es computable si y solo si es Turing computable si y solo si es recursivamente computable si y solo si es λ computable.

Observe que esta definición es esencialmente la Tesis de Turing 1, tesis que ha sido uno de los paradigmas más fuertes y ampliamente aceptado en computación [EGW04]. Otra de las consecuencias que tuvo la equivalencia de los modelos clásicos demostrada por Kleene, es la aceptación generalizada de la computación como una parte de las matemáticas y no como una ciencia propia con sus procesos y dinámicas.

Aceptar la Tesis de Turing impacta el estudio de la computación, ya que limita lo que entendemos por *computable*, Eberbach [EGW04] discute en su artículo un ejemplo físico sencillo que pone en vilo la tesis de Turing, propone considerar el problema de calcular la ruta de la escuela a la casa para un vehículo automático, según lo expuesto en el artículo esta tarea resulta imposible de calcular bajo la mirada de la computación clásica, ya que sería necesario conocer y procesar una cantidad exponencial (potencialmente infinita) de datos que cambian constantemente.

Ahora bien, abundan las aplicaciones que podemos descargar a nuestros celulares que calculan la ruta, es decir no es un problema sin solución. *¿Qué paso aquí entonces?* Eberbach responde que esto es consecuencia de la mirada estática que tiene la computación clásica sobre el mundo, mirada poco realista ya que vivimos en un mundo dinámico y caótico, en constante cambio.

Por otro lado, la materialización de la ENIAC permite retomar la idea de construir máquinas capaces de computar, sentando bases para las computadoras digitales las cuales van a ampliar el concepto de computación pues es necesario diseñar nuevos entornos, por ejemplo los lenguajes de programación, que permitan la interacción hombre-máquina, lo cual va a diversificar el horizonte de lo computable.

Para ilustrar este punto analice la siguiente situación; a dos personas A y B se les enseña el mismo lenguaje de programación con exactamente el mismo repertorio de instrucciones y se les asigna que diseñen un programa para calcular a^n. La codificación de la solución obtenida por cada persona puede ser diferente obsérvese, el Cuadro 1, la pregunta es *¿por qué esta diferencia?* En principio las personas conocían el mismo conjunto de instrucciones, pero ven el mundo de maneras diferente y cada uno plasma su visión particular en su codificación, lo cual resulta importante para lo que podemos llamar computación hoy en día.

Lo expuesto anteriormente permite que gane fuerza la idea de concebir la computación como ciencia, con sus ideas, retos y dificultades propias. En esta dirección se han realizado esfuerzos de diseñar y formalizar nuevos modelos computacionales importando ideas de otras áreas del conocimiento que permitan ampliar el horizonte [Syr10]. También han surgido modelos que apoyados en la matemática permiten estudiar fenómenos computacionales más generales [LN15], llevando a concebir nuevamente la computación como parte de las matemáticas y es por esto que es prudente introducir las siguientes discusiones.

2.1. Métodos numéricos

Los métodos numéricos más conocidos, como el método de la bisección, la factorización LU o los que se encuentran en libros sobre el tema, son naturalmente funciones computables, de hecho es posible encontrar miles de codificaciones para ellos en linea.

Al estudiar dichos métodos a fondo se observa un comportamiento particular, muchos son procesos repetitivos y realmente su salida es el cómputo de un punto fijo. Para detallar este aspecto piense en el método de Newton, en éste $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ y se obtiene una solución en el momento en que $|x_{n+1} - x_n| < \epsilon$ para un ϵ dado. Observe que para operar el método realmente no es tan relevante quien sea f, la operación puede ser ejecutada sin importar la naturaleza de esta función.

Esta observación nos permitiría pensar en el método de Newton como una operación que recibe una función y nos devuelve *algo* sobre dicha función, en otras palabras es un método recursivo que trabaja sobre funciones. Y en efecto muchos paquetes matemáticos codifican el método de Newton bajo esta filosofía, reciben una función y devuelven una solución.

No obstante esta operación que recibe funciones como entrada, no es sencilla de formalizar usando máquinas de Turing y en muchos de los casos no es posible lograr una implementación desacoplada de la función f particular. En este escenario surgen como respuesta los modelos computacionales de orden superior, en donde las funciones son *objetos de primera categoría* o en palabras sencillas se vuelven objetos que pueden ser parámetros de otras funciones.

2.2. Lógica

La computación clásica se han cimentado sobre la base de la axiomatización de la teoría de conjuntos y la lógica clásica, ya que las funciones parciales recursivas son representables en esta teoría. A su vez, uno de los principios ampliamente aceptados en teoría de conjuntos es la regla del tercio excluido, la cual afirma que siempre se cumple una proposición p o su negación. Esta regla es sumamente importante en matemática e incluso en computación clásica se encuentra presente. No obstante hay aplicaciones de esta regla en donde se hace un *salto de fé*, en el sentido que no es claro exactamente cuál de las dos posibilidades se da, este mismo fenómeno se repite al considerar el cuantificador existencial.

En contraste, una de las características presentes en la computación es su naturaleza constructiva, en general no se busca únicamente la solución a un problema sino también una manera de construir dicha solución. De lo anterior, se hace necesario estudiar la lógica subyacente a la computación pues parece no corresponder con la versión clásica.

En matemáticas han emergido lógicas no clásicas las cuales se han encargado de estudiar otro tipo fenómenos, en particular surge la lógica intuicionista fundamentada en la idea que toda afirmación matemática se acepta por probada si es posible proveer una construcción de la misma. En esta lógica se rechaza la ley del tercio excluido y los cuantificadores existenciales sin testigos, pues al realizar una disyunción se debe especificar explícitamente el camino a seguir o el objeto que satisface el existencial.

Posterior al desarrollo y formalización del intuicionismo se logra probar el conocido isomorfismo de Curry-Howard, un resultado importante en computación pues establece una correspondencia entre el cálculo Lambda con tipos y pruebas en el sistema intuicionista [SU06], una aplicación de este hecho es la extracción automática de pruebas para una proposición a partir de la codificación un programa de computador.

Del isomorfismo de Curry-Howard surge la idea de explorar resultados similares para otros tipo de sistemas computacionales y lógicos. Así, por ejemplo hoy en día se está trabajando en la relación existente entre la lógica lineal y el cálculo π. Lo cual puede llevar a pensar que existe una relación y retroalimentación mucho más activa entre estas ramas. También surge la pregunta entorno a la existencia de lógicas

para todo tipo de fenómenos computacionales, por ejemplo cuál es la lógica detrás de los protocolos de red.

2.3. Decidibilidad y Conjuntos

En el desarrollo del curso surge una pregunta realizada por Tonatiuh Widerhoold *¿es la decidibilidad en la teoría de conjuntos equivalente a la decidibilidad de las máquinas de Turing?* Pregunta que no puede ser respondida con prontitud y permite abrir una discusión llena de riqueza.

Para empezar se deben fijar las nociones básicas de lo que significa ser decidible; en teoría de conjuntos se dice que una proposición es decidible si es posible deducir la proposición o su negación a partir de un conjunto de reglas y axiomas. Por otro lado un problema, del tipo Si/No, se dice que es decidible si dada una instancia del problema existe una máquina de Turing que computa la solución y siempre termina sin importar la entrada. Poonen [Poo14] estudia la conexión entre las nociones de decidibilidad entre sistemas axiomáticos y máquinas de Turing, así como una amplia colección de problemas matemáticos indecidibles en el sentido de Turing.

Hartmanis [HH76] presenta un ejemplo de máquina de Turing para la cual no es posible demostrar su tiempo de ejecución usando la teoría de conjuntos, por otra parte Hajek [Háj79] en su artículo sobre jerarquías aritméticas y complejidad, va un más allá y muestra la existencia de máquinas de Turing con Oráculos para las cuales la pregunta $P = NP$ resulta independiente de la teoría de conjuntos.

Los anteriores artículos abren camino a la siguiente pregunta, *¿será posible que $P = NP$ sea independiente de la teoría de conjuntos?* Aaronson [Aar03] discute esta idea e indica que es un tema de interés para la comunidad, ya que una posible independencia derivaría en nuevas ramas para la computación en donde $P = NP$ y otras en donde $P \neq NP$, como ha ocurrido con otros problemas tales como la hipótesis del continuo o las geometrías no euclidianas.

En particular Aaronson cuestiona si será posible que una pregunta con tanta profundidad y consecuencia tanto para la comunidad académica como para la industria pueda quedar sin respuesta. Una visión optimista del asunto evocaría el comentario de Hilbert sobre *No-ignorabimus* [Rei19] para motivar el trabajo en torno a este problema.

2.4. Computación Paralela y distribuida

Por último y para concluir esta sección es importante mencionar la existencia de ideas surgidas en la computación para las cuales no existe una formalización matemática satisfactoria; hoy en día es común hablar de computación paralela y distribuida. La idea en este marco de trabajo es contar con más de una unidad de computo para resolver un problema, se pide que el problema sea divisible en partes y de esta manera resolver cada parte en las distintas unidades, obteniendo soluciones parciales que construyen la solución total.

Este tipo de computación ha tenido un crecimiento constante gracias al surgimiento de las GPU, las redes de computo e Internet; este último ha permitido generar redes masivas de computo con un gran capacidad computacional. Sin embargo este modelo de computación no garantiza la solución de cualquier problema y como toda aproximación cuenta con sus propios retos y problemas:

- Lockdown: cuando un problema tiene por partes A, B, si para computar la solución de A se necesita de la solución de B y viceversa, los cómputos se quedan en un bucle infinito esperando mutuamente.
- Particionar el problema: en general no es claro cómo se deberá dividir un problema o los datos que involucra e incluso tampoco es posible saber si al hacer esta partición se esta cambiando el sentido del problema original.
- Convergencia de las soluciones: no existe un método estándar para unificar las soluciones parciales obtenidas, además si es posible omitir soluciones parciales en la construcción de la total.

Estas y algunas otras dificultades son retos abiertos para la comunidad, han surgido algunas teorías como el cálculo π que busca entender los fenómenos que están emergiendo, sin embargo se debe decir que suelen ser tan complejas que imponen una barrera para su estudio y para el público no especializado.

3. Hipercomputación

Las limitaciones que han ido surgiendo con las máquinas de Turing, ponen sobre la mesa la necesidad de diseñar nuevos modelos computacionales. En este sentido se han realizado esfuerzos inmensos, como los recopilados por Syropoulous en [Syr10], de materializar nuevos modelos. Muchos de estos modelos son conocidos como modelos super-Turing o modelos de orden superior, sus motivaciones y esencia son diferentes. A continuación se expondrán dos de estas ideas así como algunos detalles de los mismos.

En los siguientes modelos el concepto abstracto de computación es el mismo, informalmente se refiere a la manipulación de datos con la intención de generar nuevos datos y que consta de dos ingredientes fundamentales *operaciones computacionales* y *datos*.

3.1. Modelos de ordenes superior

Longley y Normann [LN15] han trabajado en la formalización de los modelos de ordenes superiores, la esencia de éstos es permitir que las operaciones sirvan como entrada para otras operaciones. Una primera motivación para el estudio de este tipo de modelos es la dificultad que tienen las máquinas de Turing para materializar esta idea, una segunda motivación es el paradigma de programación funcional, el cual esta cimentado en este tipo de comportamiento y que algunos lenguajes de programación populares como Java han apropiado[2] en su repertorio de instrucciones.

Considere el siguiente ejemplo, sea $\varphi : \mathbb{N} \to \mathbb{N}$ la función que decodifica las máquinas de Turing por medio de una enumeración de Gödel y dado un n tiene sentido la siguiente operación $\Delta(n) = \varphi(n)(n)$, llámela operación diagonal. Ahora bien, el estudio de esta operación es complicado bajo la mirada tradicional ya que depende en gran medida de la función φ que se tome; aquí los modelos de orden superior representan una ventaja ya que la diagonal puede ser abstraída como $\Delta : (\mathbb{N} \to \mathbb{T}) \to \mathbb{N} \to \mathbb{N}$ que representa una operación que toma una función que convierte naturales en máquinas de Turing, un natural y los transforma en un natural.

Los modelos computacionales de orden superior consideran los datos como objetos que constan de un nombre y una implementación particular del mismo, tal como ocurre en los lenguajes de programación, por ejemplo un vector puede ser implementado usando arreglos o listas enlazadas, a su vez las operaciones son funciones parciales. La siguiente es la definición formal:

Definición 2. Un modelo computacional de orden superior C sobre un conjunto de nombres T consiste en:

- una familia indexada $|C| = \{C(\tau) : \tau \in T\}$ de conjuntos llamados los tipos.
- para cada $\sigma, \tau \in T$ un conjunto $C[\sigma, \tau]$ de funciones parciales $f : C(\sigma) \to C(\tau)$.

Tal que,

1. para cada $\tau \in T$ la función identidad $id : C(\tau) \to C(\tau)$ se encuentra en $C[\tau, \tau]$.
2. para cada $f \in C[\sigma, \tau]$ y $g \in C[\tau, \omega]$ se tiene que $g \circ f \in C[\sigma, \omega]$.

Matemáticamente hablando, un modelo computacional de orden superior es una categoría. Esta aproximación ofrece ventajas a la hora de trabajar con las operaciones ya que permite un enfoque más general y uniforme sobre las mismas, además ofrece flexibilidad en el nivel de abstracción a considerar, en otras palabras estudiar las operaciones en diferentes niveles e incluso considerar elementos situacionales de las mismas, tal como sucede con los fragmentos del Cuadro 1.

Se presenta ahora un ejemplo de un modelo computacional de orden superior. Sean $f, g \in \mathbb{N}^{\mathbb{N}}$ dos funciones y $\langle \dots \rangle$ una codificación adecuada, se define:

$$(f \odot g)(n) := f(\langle n, g(0), g(1), \dots, g(r-1) \rangle) - 1$$
$$r = \mu r. f(\langle n, g(0), g(1), \dots, g(r-1) \rangle) > 0$$

[2]Históricamente Java ha sido considerado un lenguaje de programación orientado en objetos, paradigma diferente al funcional.

en general se esta definiendo una aplicación parcial $\mathbb{N} \to \mathbb{N}$ pues el operador de minimización podría no estar definido para ciertas funciones. La estructura $(\mathbb{N}^{\mathbb{N}}, \odot)$ es conocida como el segundo modelo de Kleene y su definición esencialmente nos indica la forma de computar sobre datos (f, g) potencialmente infinitos en un número finito de pasos (r). Un hecho importante del modelo es que al dotar a $\mathbb{N}^{\mathbb{N}}$ con la topología estándar de Baire conecta las nociones de continuo y computación.

Una de las desventajas notables de esta aproximación es el uso de tipos, ya que carga de notación las expresiones y hace complejo el manejo o tratamiento de las mismas. Por otro lado, algunos de los modelos concretos como K_2 son difíciles de implementar en las máquinas con las que disponemos, en particular surge la necesidad de pensar estructuras de datos para este tipo de objetos.

3.2. Cálculo Pi

Milner y Parrow [MPW92] introducen en 1992 el cálculo π, en palabras de los autores *Se presenta el cálculo de transmisiones para los sistemas móviles, en el cual es posible modelar operaciones que tienen una estructura cambiante. En este marco de trabajo las operaciones del sistema no están enlazados arbitrariamente, sino que la comunicación entre ellos permite cambiar dichos enlaces.*

Este modelo ha permitido aproximarse a la formalización de los sistemas móviles, entendiendo la movilidad como la capacidad de transmitir datos de una operación a otra o la capacidad de crear enlaces entre dos operaciones. Visión que ha hecho posible modelar matemáticamente algunos fenómenos computacionales como lo son las redes de computo o el trabajo concurrente dentro de una computadora personal.

El cálculo π se inspira en la gramática del cálculo λ para determinar su propia gramática.

$$\begin{aligned} \pi &:= \bar{x}y \parallel x(z) \parallel \tau \parallel [x = y]\pi. \\ P &:= M \parallel P|Q \parallel \nu z.P \parallel !P \\ M &:= 0 \parallel \pi.P \parallel M + M' \end{aligned}$$

Como es de esperar los términos producidos por esta gramática suelen ser bastante complejos y por lo mismo el tratamiento de los mismos es complejo. De manera análoga, para este cálculo se definen reducciones análogas a las β para el cálculo λ y también resultan con propiedades deseables como la propiedad de Church-Rosser.

4. Conclusiones

Computar es una necesidad del hombre y ha surgido en distintos procesos a lo largo de la historia, no sólo porque le permite optimizar su tiempo de vida sino también porque ha permitido cambiar la manera en la cual nos relacionamos con nuestro ambiente.

Las matemáticas y la computación tienen una relación cercana, tanto así que en ocasiones no es claro sobre cuál de ellas estamos hablando. Su colaboración continua ha permitido la exploración de nuevas ideas y métodos que han aportado al avance de la sociedad y han motivado el estudio formal de cada una y de sus interacciones.

Las discusiones de la sección 2 permiten afirmar por un lado, que la computación hoy en día escapa a la matemática, en gran medida por la naturaleza de las preguntas que surgen, el nivel de profundidad de las mismas y la carencia de marcos de trabajo que sean capaces de capturar aspectos propios de las preguntas. Por otro lado, que existe una latente necesidad de estructurarla como una ciencia con cuerpo y características propias.

Han emergido distintas visiones sobre el concepto de computación y su relación con otras ciencias, Figura 1, en cada una se plasma la manera particular en la cual percibimos el mundo. Esta variedad aporta a la construcción de una ciencia de la computación.

Por último, es necesario comprender la computación como una ciencia social, en particular las redes e Internet han cambiado radicalmente la forma en la cual interactúan los seres humanos emergiendo fenómenos particulares. Así, por ejemplo hoy en día existen grupos de personas trabajando alrededor de la computación colaborativa o computación humana, los cuales buscan aumentar significativamente el

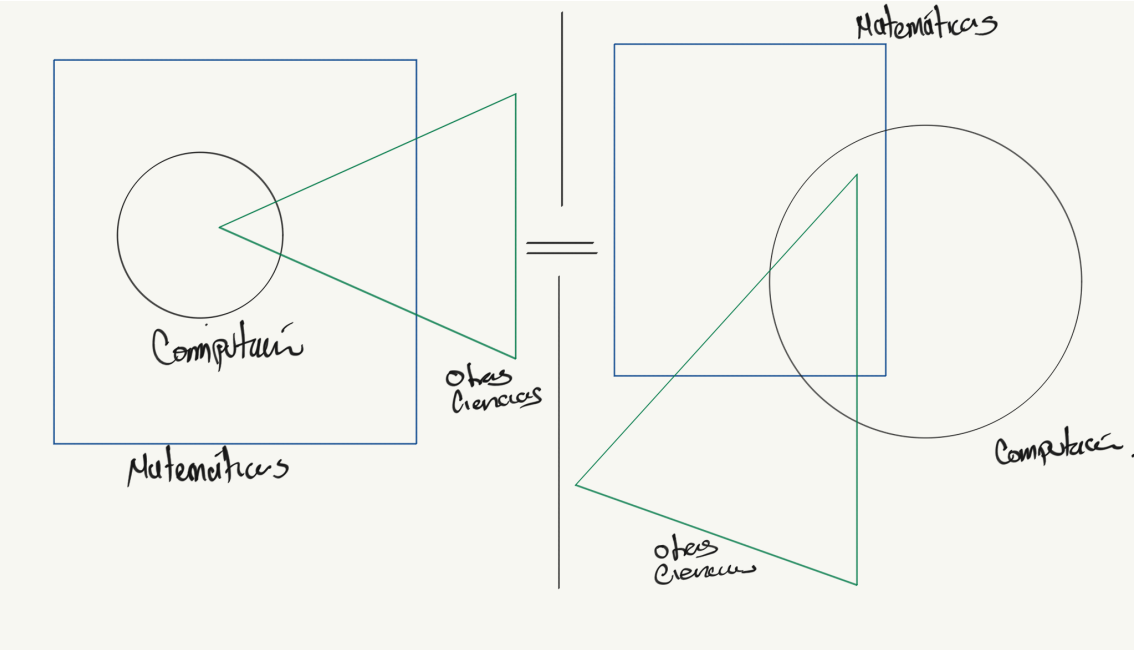

Figura 1: Visiones de sobre la computación.

poder computacional del hombre sumando 'pequeñas' unidades de computo alrededor del mundo y con la finalidad de resolver problemas a gran escala. Ejemplos de este tipo de proyectos son:

- IBM: World Community Grid
- Folding at Home
- World Community Grid

Referencias

[Aar03] Scott Aaronson. Is p versus np formally independent. *Bulletin of the European Association for Theoretical Computer Science*, 81:109–136, 2003.

[Bal60] W.W.R. Ball. *A Short Account of the History of Mathematics*. Dover Books on Mathematics Series. Dover Publications, 1960.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936.

[Coo03] S.B. Cooper. *Computability Theory*. Chapman Hall/CRC Mathematics Series. Taylor & Francis, 2003.

[EGW04] Eugene Eberbach, Dina Goldin, and Peter Wegner. *Turing's Ideas and Models of Computation*, pages 159–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Gö31] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[Háj79] P. Hájek. Arithmetical hierarchy and complexity of computation. *Theor. Comput. Sci.*, 8:227–237, 1979.

[HH76] J. Hartmanis and J. E. Hopcroft. Independence results in computer science. *SIGACT News*, 8(4):13–24, October 1976.

[HS08] J.R. Hindley and J.P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, 2008.

[Kin96] David King. Is the human mind a turing machine? *Synthese*, 108(3):379–389, 1996.

[Kle36] S. C. Kleene. λ -definability and recursiveness. *Duke Math. J.*, 2(2):340–353, 06 1936.

[LN15] J. Longley and D. Normann. *Higher-Order Computability*. Theory and Applications of Computability. Springer Berlin Heidelberg, 2015.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1):1–40, September 1992.

[Poo14] Bjorn Poonen. Undecidable problems: a sampler, 2014.

[Rei19] Andrea Reichenberger. From solvability to formal decidability: Revisiting hilbert's "non-ignorabimus". *Journal of Humanistic Mathematics*, 9:49–80, 01 2019.

[Rob15] B. Robič. *The Foundations of Computability Theory*. Springer Berlin Heidelberg, 2015.

[Sip12] M. Sipser. *Introduction to the Theory of Computation*. Introduction to the Theory of Computation. Cengage Learning, 2012.

[SU06] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics)*. Elsevier Science Inc., USA, 2006.

[Syr10] Apostolos Syropoulos. *Hypercomputation: Computing Beyond the Church-Turing Barrier*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[Tur37] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 01 1937.