
**UNIVERSIDAD DISTRITAL**
FRANCISCO JOSÉ DE CALDAS

# Números construibles y computabilidad

MONOGRAFÍA DE TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE MATEMÁTICO
PROYECTO CURRICULAR DE MATEMÁTICAS

Daniel Meshir Vargas
Dirigido por: Carolina Mejía

Bogotá DC
Octubre de 2020

**Resumen**

Los números construibles con regla y compás relacionan el Álgebra con la Geometría, las cuales son dos ramas muy importantes de las matemáticas, además en éste escrito conectamos por medio de los números construibles, la Geometría Euclideana con la teoría de la computación, a través de una máquina con el poder computacional de una máquina de Turing que se construye en éste trabajo, con la cual mostramos la íntima relación entre los números computables y los números construibles con regla y compás, vinculamos también lo no computable con las imposibilidades de construcciones geométricas, lo cual no se relaciona con el problema de la parada ni sus derivados, que frecuentemente se utilizan para comprender la naturaleza de lo no computable.
**Palabras clave**: No construible, no computable, problema indecidible.
**Clasificación AMS**: Primaria: 51M15 Secundaria: 03D99

**Agradecimientos**: *En primera instancia agradezco a la profesora Carolina Mejía por su orientación y sus sugerencias en el desarrollo de éste escrito. A mis formadores y compañeros, personas de gran sabiduría, quienes me han brindado a lo largo de estos años el conocimiento y apoyo para conseguir mis metas.*

*Agradezco el apoyo incondicional de mis padres y mi hermano, que son mi inspiración día a día, y además, la causa de mis alegrías.*

# 1. Introducción

El reto de encontrar un algoritmo general capaz de decidir si una fórmula del cálculo de primer orden es un teorema, fue planteado por David Hilbert y Wilhelm Ackermann en 1928 bajo el nombre de Entscheidungsproblem (Problema de decisión). La negativa a la existencia de este algoritmo capaz de decidir si una frase era o no un teorema, fue dada por Alan Turing al reducir el problema de decisión al problema de la parada que se describe detalladamente en [4] y en [11]. EL problema consiste en determinar si una máquina de Turing $M$ con una entrada inicial $\omega$ (palabra inicial $\omega$) se detiene en un número finito de pasos, demostrando Turing que este problema es indecidible, y mostrando así los límites teóricos de los ordenadores. El problema de la parada es uno de los más utilizados para comprender lo indecidible (no computable) o no recursivo, pero podemos ir a la naturaleza de la no computabilidad a través de un camino diferente, por medio de famosos problemas de construcción con regla y compás como el duplicar el cubo, la cuadratura del círculo o la trisección de un ángulo dado. Nuestro interés se enfocará en este último problema de construcción con regla y compás.

*Definición* 1 (Número Construible [5]). Un número real **c** es un **número construible** si podemos construir un segmento de linea de longitud |**c**| en un número finito de pasos a partir de la **unidad de longitud** con regla y compás.

A partir de esta definición, en [5] se muestra de manera detallada el siguiente teorema.

*Teorema* 1. El campo $F$ de todos los números construibles consiste precisamente de todos los números reales que pueden ser obtenidos tomando raíces cuadradas de números positivos del campo $\mathbb{Q}$, y aplicando un número finito de veces las operaciones de campo.

A partir de este resultado, se llega al siguiente corolario que nos ayuda a probar las imposibilidades clásicas con regla y compás por medio de su contrarrecíproca.

**Corolario 1**. *Sea $\alpha$ un número construible y $\alpha \notin \mathbb{Q}$, entonces es una secuencia finita de números reales, $\alpha_1, \alpha_2, ..., \alpha_n = \alpha$, tal que $\mathbb{Q}(\alpha_1, \alpha_2, ..., \alpha_i)$ es una extensión de $\mathbb{Q}(\alpha_1, \alpha_2, ..., \alpha_{i-1})$ de grado 2. En particular $[\mathbb{Q}(\alpha) : \mathbb{Q}] = 2^n$ para algún $n \in \mathbb{N}$, donde $[\mathbb{Q}(\alpha) : \mathbb{Q}]$ es el índice de la extensión.*

*Proposición* 1 (Trisecar un ángulo). Dado un ángulo, no siempre es posible trisecarlo con regla y compás.

*Demostración.* Tomemos el ángulo 60°, y mostraremos que no puede ser trisecado. Veamos que:

$$\begin{aligned} cos(3\theta) &= cos(2\theta + \theta) \\ &= (2cos^2(\theta) - 1)cos(\theta) - 2cos(\theta)(1 - cos^2(\theta)) \\ &= 4cos^3(\theta) - 3cos(\theta) \end{aligned}$$

Necesitamos que $\theta = 20°$, y sea $\alpha = cos(20°)$. Así obtenemos la ecuación

$$4\alpha^3 - 3\alpha = \frac{1}{2}$$

ya que $cos(60°) = \frac{1}{2}$. Con lo que $\alpha$ es raíz del polinomio $8x^3 - 6x - 1$; el cual es un polinomio irreducible en $\mathbb{Q}[x]$ ya que es un polinomio de grado 3 cuyas raíces no están en $\mathbb{Q}$. Con lo que $[\mathbb{Q}(\alpha) : \mathbb{Q}] = 3 \neq 2^n$ para todo $n \in \mathbb{N}$. Por lo que el ángulo de 60° no es trisecable con regla y compás. □

Esta última proposición la retomaremos en las siguientes secciones.
Podemos encontrar en [6] que todos los problemas de construcción que se resuelven con regla y compás, pueden resolverse utilizando solamente un compás; esto es conocido como Teorema de Mohr-Mascheroni.

Así nuestro objetivo es ver la relación entre los números computables y los números construibles con regla y compás a través de la creación de una máquina que solo utiliza estos dos instrumentos (regla y compás).

# 2. Máquina Euclidiana

En esta sección vamos a construir una máquina abstracta, que llamaremos como en [9] Máquina Euclideana. Nuestra máquina consistirá en una hoja de papel tan grande como sea necesaria, una regla sin marcas, y un compás que será ideal, es decir un compás capaz de abrir tanto como sea necesario.

## 2.1. Operadores de la Máquina Euclidiana

Ahora vamos a especificar una lista de operaciones posibles para la construcción de algoritmos que determinan las máquinas Euclideanas, serán listadas como en [9], con unas pequeñas variaciones para mayor comprensión del lector.

- P($p_1, ..., p_n$)- Dibujar un número finito de puntos distintos (en este caso dibujara $n$ puntos distintos)
- L($P, Q$)- Dibujar la línea que pasa a través de $P$ y $Q$.
- C($P, Q$)- Dibujar una circunferencia con centro en $P$ y que pasa por el punto $Q$.
- EC($P, Q; A$)- Etiquetar, nombrar con la letra $A$ la circunferencia con centro en $P$ y que pasa por el punto $Q$.
- EL($P, Q; A$)- Etiquetar, nombrar con la letra $A$ la línea que pasa por los puntos $P$ y $Q$

- EP($O_1, O_2; A$)- Etiquetar, nombrar con la letra $A$ el punto de intersección entre el objeto 1 y el objeto 2, (ya sean rectas o circunferencias).
- D($A$)- Borrar la etiqueta $A$.
- X ∈ C: n- Si el punto X está en el interior del círculo C o en la circunferencia C, entonces ejecuta la n-ésima instrucción; de otra forma continúa con la instrucción siguiente.
- Moff- Detiene el programa.

Un programa o algoritmo, será una lista de operaciones ordenada del tipo descrito arriba. Después de la $n$-ésima operación se ejecutará la $n+1$, a menos que se ejecute una operación X∈ C: $n$.

Comentaremos algunos aspectos de la lista de operaciones. De la primera operación, al ser los puntos diferentes, estos siempre estarán etiquetados. Cada etiqueta puede identificar un único objeto, esto no quiere decir que un objeto no pueda tener más de una etiqueta. El número de puntos iniciales es arbitrario y depende directamente del algoritmo o programa que se requiera realizar. La operación EP debe etiquetar todos los puntos de intersección, por ejemplo, si una recta L y una circunferencia C se intersectan en dos puntos, podemos utilizar esta operación como EP($L, C, A_1, A_2$). Si alguna operación no puede ser ejecutada como por ejemplo L($Q, Q$), ya que para construir la línea se hacen necesarios dos puntos distintos, o si EP($L, C, A_{,1}, A_2$) no puede ser ejecutada ya que la recta y la circunferencia no se intersectan o se intersecta nada más en un punto, entonces nuestro programa se detiene incorrectamente. La operación X ∈ C: n- nos permite crear bucles anidados, y ser equivalente a un condicional. La última operación $Moff$ no es necesaria en ningún programa, y no alterará los resultados obtenidos.

Vamos a ver algunos ejemplos de programas.

## 2.2. Algunos algoritmos de máquinas Euclideanas

*Ejemplo* 1. Consideremos la construcción de una recta perpendicular a otra que pasa por un punto extremo. Para esto necesitamos 2 puntos iniciales distintos, entonces dibujaremos la línea a través de dos puntos y una línea perpendicular a la anterior que pase por un punto extremo.
Algoritmo PL($A, B; B, l$):

| | | | |
|---|---|---|---|
| 1:: P($A, B$) | 5:: EC($A, B; C_1$) | 9:: L($A, C$) | 13:: C($D, B$) |
| 2:: L($A, B$) | 6:: C($B, A$) | 10:: EL($A, C; l_2$) | 14:: EC($D, A; C_4$) |
| 3:: EL($A, B; l_1$) | 7:: EC($B, A; C_2$) | 11:: C($C, B$) | 15:: D($A$) |
| 4:: C($A, B$) | 8:: EP($C_1, C_2; C, D$) | 12:: EC($C, B; C_3$) | 16:: EP($l_2, C_3; G, F$) |

17:: $G \in C_4$: 21
18:: L($G, B$)
19:: EL($G, B; l$)
20:: Moff
21:: L($B, F$)
22:: EL($B, F; l$)

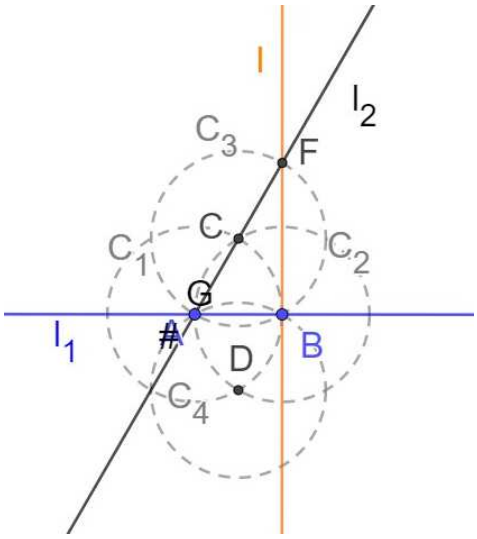


Figura 1: PL(A,B;B,l)

Al algoritmo anterior podríamos modificarlo de tal forma que la recta perpendicular pase por el punto $A$ en vez de por $B$, cambiando en algunas operaciones el orden de $A$ y $B$; podemos nombrar el algoritmo anterior con la idea de que lo podamos utilizar en futuros programas sin tener que reescribir todas sus líneas. En particular a este algoritmo será llamado como PL($A, B; B, l$), y construirá una recta perpendicular que pasa por el punto $B$, a la recta que pasa por $A$ y $B$, y la etiquetará como $l$.

Nuestros algoritmos no son deterministas, en el sentido de que dependen del orden en que los puntos sean etiquetados, es por esta razón que en el ejemplo 1 hay que considerar que se etiqueten en órdenes distintos el punto $F$ y $G$. Y aunque esto no es realmente relevante para las construcciones, podemos poner algunas condiciones sobre las rectas y circunferencias y esto puede ser resuelto. Elijamos un plano cartesiano sin marcas como marco de referencia; la instrucción $EP(O_1, O_2; A, B)$ será ejecutada de la siguiente forma:

- Si $O_1, O_2$ son líneas, entonces su intersección (si la hay) siempre será etiquetada.
- Si $O_1$ es una línea y $O_2$ una circunferencia, su intersección (si la hay) será etiquetada de izquierda a derecha y de arriba hacia abajo.
- Si $O_1, O_2$ son circunferencias, como en el ítem anterior su intersección (si la hay) será etiquetada de izquierda a derecha y de arriba hacia abajo.

*Ejemplo* 2. Construcción de un bisector de un ángulo dado tres puntos.
Algoritmo $Angle(A, B, C; l)$

1:: $P(A, B, C)$
2:: $L(A, B)$
3:: $L(B, C)$
4:: $EL(A, B, l_1)$
5:: $El(B, C, l_2)$
6:: $C(B, A)$
7:: $EC(B, A; c_1)$
8:: $EP(C_1, l_2; D, E)$
9:: $C(E, B)$
10:: $EC(E, B; c_e)$
11:: $C \in c_e : 30$
12:: $C(C, B)$
13:: $EC(C, B; c_c)$
14:: $E \in c_c : 30$
15:: $C(D, B)$
16:: $EC(D, B; c_2)$
17:: $C(A, B)$
18:: $EC(A, B; c_3)$
19:: $EP(c_2, c_3; F, G)$
20:: $F \in c_3 : 26$
22:: $L(B, F)$
23:: $EL(B, F, l)$
24:: $D(D, E, c_e, c_c, G)$
25:: $Moff$
26:: $L(B, G)$
27:: $EL(B, G, l)$
28:: $D(D, E, c_e, c_c, F)$
29:: $Moff$
30:: $C(E, B)$
31:: $EC(E, B; c_2)$
32:: $C(A, B)$
33:: $EC(A, B; c_3)$
34:: $EP(c_2, c_3; F, G)$
35:: $F \in c_3 : 40$
36:: $L(B, F)$
37:: $EL(B, F, l)$
38:: $D(D, E, c_e, c_c, G)$
39:: $Moff$
40:: $L(B, G)$
41:: $EL(B, G, l)$
42:: $D(D, E, c_e, c_c, F)$
43:: $Moff$

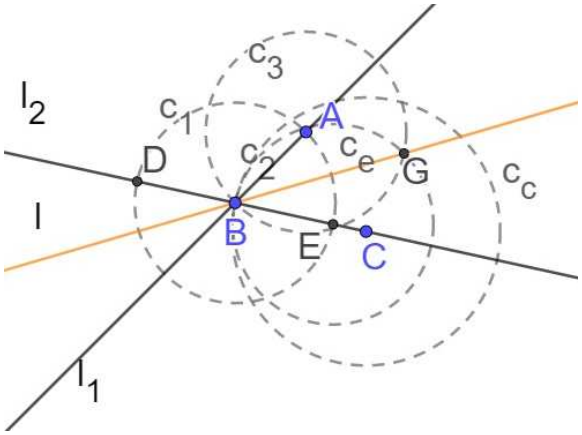


Figura 2: Angle(A,B,C,l)

Ahora vamos a ver la analogía que existe entre las máquinas Euclideanas y los teoremas de la geometría Euclideana.

*Ejemplo* 3. Consideremos la construcción del teorema de Tales que se realiza en [9, pag 9]; un ángulo inscrito en una semicircunferencia es recto. ¿Y cómo podemos verificar este teorema con una Máquina de Euclides?

1:: Dibujar tres puntos diferentes no colíneales, $A, O, X$
2:: Dibujar una circunferencia $C$ con centro en $O$ y que pase por $A$
3:: Dibujar una línea $L$ que pase por los puntos $A, O$
4:: Etiquetar el otro punto de intersección de $C$ y $L$ como $B$

5:: Dibujar la línea $L2$ que pasa por los puntos $O, X$
6:: Etiquetar los puntos de intersección de $C$ y $L2$ como $P, Q$
7:: Dibujar la línea $L3$ que pasa por $A, P$
8:: Dibujar la línea $L4$ que pasa por $B, P$
9:: Dibujar una línea perpendicular a $L4$ que pase por $P$
10:: Etiquetar la intersección de $L4$ con $C$ como $A'$

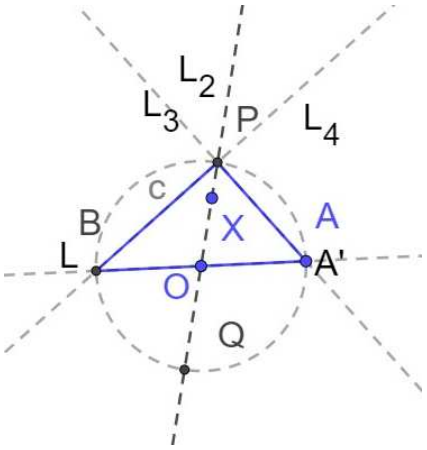


Figura 3: Teorema de Tales

El lector puede ver que es sencillo crear el algoritmo en las máquinas de Euclides, pasar a operaciones esta última lista. Construimos el ángulo $APB$ que esta inscrito en la circunferencia $C$, y luego la recta perpendicular a $l4$ que pasa por $P$, decir que el ángulo $APB$ es recto, equivale a decir que $A$ y $A'$ son iguales. Podemos utilizar una operación que nos ayudará a verificar esto, si asumimos que todo punto es una circunferencia de radio 0.

- $A \in A'$: $n$

Así podríamos hacer que el programa termine con un $Moff$ si la condición es verdadera o que entre en un bucle si la condición es falsa. O también podríamos etiquetar algún punto con un símbolo seleccionado, por ejemplo etiquetar el punto $O$ con 1 si la condición es verdadera o 0 si la condición es falsa.

Así, la prueba es correcta con las operaciones si para toda configuración inicial obtenemos el símbolo elegido especial de verdadero (1). Con esto se hace evidente que toda proposición de Euclides se puede verificar con una máquina de Euclides, pero aún falta ver todo el potencial de las máquinas que definimos en ésta sección, para esto, vamos a recurrir a la Máquinas de Registro Ilimitado que nos ayudarán en la conexión de la Turing completez de la máquinas de Euclides.

# 3. Máquina de registro ilimitada

La idea de esta sección es ver el comportamiento de las máquinas que llamaremos "Máquina de Registro Ilimitada", o por sus siglas en ingles URM. Mostraremos que son Turing completas, es decir que toda función recursiva parcial (ver [4]) es computable bajo URM. Esto con el fin de crear una relación entre las Máquinas Euclideanas y las URM para, en secciones futuras, mostrar la Turing completez de las Máquinas Euclideanas. Las Máquinas de Registro Ilimitado son máquinas abstractas que consisten en un número de ubicaciones llamadas registros numerados por $1, 2, 3, \ldots$ donde cada registro puede almacenar cualquier número natural $0, 1, 2, 3, \ldots$ Cada programa de una URM debe hacer uso de, únicamente un número finito de estos registros, el resto de registros deben permanecer vacíos o en otras palabras contener el 0 a través de toda la ejecución del algoritmo (es por esta razón que tomamos a 0 como un natural).

## 3.1. Instrucciones de las URM

Como en la sección anterior, definiremos unas instrucciones básicas (comandos básicos) u operaciones posibles para crear algoritmos de URM. Los registros serán notados como en [10] $\langle n \rangle$, $\langle n' \rangle$ el contenido del registro $n$ y $n'$ antes y después de ser cambiado respectivamente en una instrucción. El conjunto de instrucciones que elegimos a continuación, se toman con el fin de que sea más sencillo ver la computabilidad de las funciones recursivas parciales (ver [4]), y así poder ver la Turing completez de las URM. Las instrucciones son:

- P($n$)- Suma 1 al actual valor en el registro $n$, esto es $\langle n' \rangle = \langle n \rangle + 1$
- D($n$)- Resta 1 al número en el $n$-ésimo registro, esto es $\langle n' \rangle = \langle n \rangle - 1$. (con $\langle n \rangle \neq 0$).
- O($n$)- Vacía el registro $n$, esto es $\langle n' \rangle = 0$.
- C($m, n$)- Cambia lo que hay en el registro $n$ por lo que hay en el registro $m$, esto es $\langle n' \rangle = \langle m \rangle$
- J[$n$]- La máquina salta a la línea $n$.
- J($m$)[$n$]- Si el registro $m$ es vacío la máquina salta a la línea del programa $n$.

Las instrucciones en un programa deben leerse de manera gerárgica, comenzando en la línea 1. Nótese que la última operación nos ayuda a crear bucles, además cuando la máquina pasa a una línea que no existe se detendrá automáticamente. La última instrucción funciona también como un condicional, es decir si tomamos la instrucción $J(m)[n]$, y $m$ es un registro vacío, entonces pasará a la n-ésima línea, de lo contrario pasará a la siguiente línea.

*Ejemplo* 4 (Multiplicar por 2). Vamos a crear una máquina URM capaz de multiplicar cualquier natural por 2.

1:: $C(n_1, n_2)$
2:: $J(n_1)[7]$
3:: $P(n_1)$
4:: $D(n_2)$
5:: $J(n_2)[7]$
6:: $J[3]$.

$n_1$ y $n_2$ son registros donde $n_1$ debe ser el input del número natural que se quiere multiplicar, y $n_2$ al no ser un input (una entrada) estará vacío, es decir, al comienzo de la ejecución $n_2 = 0$. Por ejemplo, si tomamos $n_1 = 3$ como entrada, la primera línea del algoritmo hace que $n_2$ cambie de ser vacío a contener el número natural 3, es decir, $n_2 = 3$. En la segunda línea verificamos si $n_1 = 0$, esto ya que si $n_1 = 0$ el ouput del programa (el resultado de la multiplicación) debe ser 0, así simplemente el programa saltará a la línea 7 y como esta no existe, el programa parará. Como en este caso $n_1 \neq 0$ pasa a la línea 3 la cual hace que $n_1$ cambie de estado, así $n_1 = 4$, al ejecutar la cuarta línea obtenemos $n_2 = 2$, en la 5 línea se verifica si $n_2 = 0$, como $n_2 \neq 0$ entonces pasa a la línea 6 que nos reenvía a la línea 3, siguiendo este proceso obtenemos que $n_1 = 5$ y $n_2 = 1$, llegando de nuevo a la sexta línea, y repitiendo el proceso una vez mas, con lo que $n_1 = 6$ y $n_2 = 0$, como $n_2 = 0$ en la quinta línea salta a la línea inexistente 7 con lo cual el programa termina y su ouput será $n_1 = 6 = 2 \cdot 3$.

En los futuros programas de URM no utilizaremos letras para representar los registros, sino números de 1 al número de registros necesarios para nuestro programa. Así el programa anterior también puede escribirse como:

1. $C(1, 2)$
2. $J(1)[7]$
3. $P(1)$
4. $D(2)$
5. $J(2)[7]$
6. $J[3]$.

Los inputs son los primeros registros almacenados, es decir en el anterior ejemplo 1 es el primer registro que almacena el natural que se quiere multiplicar por 2, los demás registros que no sean inputs serán vacíos.

## 3.2. Computabilidad de funciones recursivas parciales bajo URM

A continuación vamos a mostrar que cualquier función recursiva parcial es calculable por una Máquina de Registro Ilimitado (URM) a través de una serie de proposiciones, en el sentido que existe una subrutina capaz de calcular cada una de las partes de la compleja definición de función recursiva parcial. Para esto, los registros input para iniciar el programa serán tomados como $1, 2, 3, ..., n$ donde $\langle 1 \rangle = x_1, \langle 2 \rangle = x_2, ..., \langle n \rangle = x_n$, los demás registros serán vacíos y el ouput será dado en el registro $n+1$.

*Definición* 2. Una función $f : \mathbb{N}^k \to \mathbb{N}$ se dice recursiva si puede obtenerse por una sucesión finita de aplicaciones de las reglas siguientes.

I. Las siguientes funciones, son conocidas como primitivas básicas

$Z(x_1) = 0$ función constante 0.
$S(x_1) = x + 1$ función sucesor.
$P_k^n(x_1, x_2, ..., x_n) = x_k$ función proyección, con $k \leq n$.

II. Composición.
Si $f(y_1, ..., y_n)$ y $g_1(x_1, ..., x_k), ..., g_n(x_1, ..., x_k)$ son recursivas entonces
$h(x_1, ..., x_k) = f(g_1(x_1, ..., x_k), ..., g_n(x_1, ...x_k))$ es recursiva.

III. Recurrencia primitiva.
Si $f(x_1, ..., x_n)$ y $g(x_1, ..., x_n, y, z)$ son *recursivas*, entonces la función $h(x_1, ..., x_k, y)$ definida por:

$$h(x_1, .., x_n, 0) = f(x_1, ..., x_n)$$
$$h(x_1, ..., x_n, y + 1) = g(x_1, ..., x_n, y, h(x_1, ..., x_n, y))$$

es recursiva.

Una función $f(x_1, ..., x_k, y)$ es regular para $y$ si para todo $n_1, ..., n_k$ existe $m$ tal que $f(n_1, ..., n_k, m) = 0$, podemos definir una nueva función $h((x_1, ..., x_k)) = \mu y(f(x_1, ..., x_l, y) = 0)$, donde $\mu y$ es el mínimo $y$ tal que $f(x_1, ..., x_k, y) = 0$.

IV. Minimización (de funciones regulares)
si $f(x_1, ..., x_k, y)$ es recursiva y regular para $y$ entonces $h(x_1, ..., x_k) = \mu y(f(x_1, ..., x_l, y) = 0)$ es recursiva.

Intuitivamente la minimización desea encontrar, comenzando desde 0, el argumento mas pequeño que hace que la función sea 0.

Con esta definición mostraremos la Turing completez de las URM.

*Proposición* 2. Todas las funciones recursivas primitivas básicas son calculables bajo las URM.

*Demostración.*

I. Funciones primitivas básicas

$Z(x_1) = 0$

1:: $O(1)$:

$S(x_1) = x_1 + 1$:

1:: $P(1)$
2:: $C(1, 2)$

$P_k^n(x_1, ..., x_n) = x_k$:

1:: $C(k, n + 1)$

De donde toda función básica recursiva primitiva es URM-calculable. □

***Proposición*** 3. Si $f(x_1,...,x_n)$ y $g_1(x_1,...,x_k), g_2(x_1,...,x_k),...,g_n(x_1,...,x_k)$ son funciones $URM-calculables$, entonces $h(x_1,...,x_k) = f(g_1(x_1,...,x_k),...,g_n(x_1,...,x_k))$ es $URM-calculable$

***Demostración.*** Como $g_1(x_1,...,x_k), g_2(x_1,...,x_k),...,g_n(x_1,...,x_k)$ son funciones $URM-calculables$ entonces para cada conjunto de números naturales $x_1, x_2,..., x_k$ existe una URM $A_i(N+i = g_i(x_1, x_2,..., x_k))$ donde $N+i$ es el registro ouput (el resultado) al calcular $g_i(x_1,...,x_k)$. Así:

II. Composición
$h(x_1,...,x_k) = f(g_1(x_1,...,x_k),...,g_n(x_1,...,x_k))$

1:: $A_1(N+1 = g_1(x_1,...,x_k))$
2:: $A_2(N+2 = g_2(x_1,...,x_k))$
$\vdots$ $\vdots$
n:: $A_n(N+n = g_n(x_1,...,x_k))$
n+1:: $A(n+1 = f(N+1,...,N+n))$

Note que los registros $1, 2,..,k$ almacenan los inputs $x_1,...,x_k$. □

***Proposición*** 4. Si $f(x_1,...,x_n)$ y $g(x_1,...,x_n,y,z)$ son $URM-calculables$, entonces

$$h(x_1,..,x_n,0) = f(x_1,...,x_n)$$

$$h(x_1,...,x_n,y+1) = g(x_1,...,x_n,y,h(x_1,...,x_n,y))$$

es $URM-calculable$

***Demostración.***

III. Recurrencia primitiva
$h(x_1,..,x_n,0) = f(x_1,...,x_n)$
$h(x_1,...,x_n,y+1) = g(x_1,...,x_n,y,h(x_1,...,x_n,y))$

1:: $A(y = f(x_1,...,x_n))$
2:: $O(N+1)$
3:: $A_{N+2}(N+2 = g(x_1,...,x_n,N+1,y))$
4:: $C(N+2,y)$
5:: $P(N+1)$
6:: $J(z)[10]$
7:: $D(z)$
8:: $J[3]$
9:: $C(N+1,z)$

□

***Proposición*** 5. Si $f(x_1,...,x_n,y)$ es $URM-calculable$ entonces la función

$$g(x_1,...,x_n) = \mu y(f(x_1,...,x_n,y) = 0)$$

es $URM-calculable$.

*Demostración.*

IV Minimización
$g(x_1,...,x_n) = \mu y(f(x_1,...,x_n,y) = 0)$

1:: $O(n+1)$
2:: $A(N = f(x_1,...,x_n,n+1))$
3:: $J(N)[6]$
4:: $P(n+1)$
5:: $J[2]$

□

Así por proposición 2,3,4 y 5 tenemos que toda función recursiva (parcial) es $URM-calculable$ y así obtenemos el siguiente corolario:

**Corolario 2** (Completez de $URM$). *Toda función calculable por una máquina de Turing es calculable por una Máquina de Registro Ilimitado* ($URM$)

*Demostración.* Por Corolario 4 de la sección 5.3 de [4] se tiene que toda función Turing calculable es recursiva, y por las proposiciones anteriores se tiene que toda función recursiva es calculable con una $URM$ por lo tanto, toda función Turing calculable es $URM-calculable$. □

### 3.2.1. Reducción de operaciones Básicas

Las Máquinas de Registros Ilimitados (URM) son muy similares a las máquinas descritas en [8], capítulo 11, por Marvin L. Minsky, en el que se logra realizar una reducción de operaciones. La idea es reducir el conjunto de operaciones de las $URM$ para, en futuras secciones, simular bajo las Máquinas de Euclides un conjunto mucho más pequeño de las operaciones de las Máquinas de Registro Ilimitado. La instrucción más evidente a reemplazar es $C(m,n)$, la cual copia lo que hay en el registro $m$ en el registro $n$. Mostraremos que existe un algoritmo para remplazar la instrucción $C(m,n)$ en términos de las demás instrucciones.

1. Subrutina para $C(m,n)$ en términos de las demás instrucciones: $C(m,n)$

   1:: $O(n)$
   2:: $O(N)$
   3:: $P(n)$
   4:: $P(N)$
   5:: $D(m)$
   6:: $J(m)[8]$
   7:: $J[3]$

   La siguiente instrucción que vamos a simplificar será $J[n]$ "saltar a la instrucción $n$".

2. $J[n]$
   1:: $O(N)$
   2:: $J(N)[n]$

Así tenemos un conjunto un poco más pequeño de operaciones de las $URM$, que en la siguiente sección simularemos bajo las máquinas de Euclides.
**Operaciones de las $URM$ Simplificadas**

- P($n$)- Suma 1 al actual valor en el registro $n$, esto es $\langle n' \rangle = \langle n \rangle + 1$
- D($n$)- Resta 1 al número en el $n$-ésimo registro, esto es $\langle n' \rangle = \langle n \rangle - 1$ (con $\langle n \rangle \neq 0$).
- O($n$)- Vacía el registro $n$, esto es $\langle n' \rangle = 0$.
- J($m$)[$n$]- Si el registro $m$ es vacío la máquina salta a la línea del programa $n$.

# 4. Turing completez de las máquinas Euclidianas

En la sección anterior mostramos la Turing completez de las Máquinas de Registro Ilimitado. A continuación, mostraremos que cualquier URM puede ser simulada por una máquina Euclidiana y por lo tanto cualquier función $URM - calculable$ será $Euclidean - calculable$.

## 4.1. Emulación de registros

Echemos un vistazo a cómo emular los registros que utilizan las $URM$ en una máquina Euclidena. Vamos a utilizar rectas para recordar los valores actuales de los registros. Cada recta va a tener los valores de un único registro para poder desplazarse sobre las rectas sin causar confusión en el registro que se está modificando mediante una operación. En la Figura 4 la recta $R_0$ se crea a través de las operaciones $P(P, Q_1)$, $L(P, Q_1)$, $EL(P, Q_1, R_0)$, donde el punto $Q_1$ será la representación vacía del registro $X$ es decir $Q_1 = \langle X \rangle = 0$.

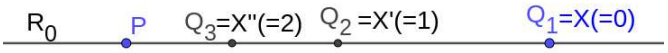


Figura 4: Registro $R_0$

Ahora si dividimos el segmento $\overline{PQ_1}$, obteniendo el punto $Q_2$, este representará el valor 1 del registro $X$ es decir $Q_2 = \langle X \rangle = 1$ como se ve en la Figura 4 Si continuamos con el proceso tenemos que la recta $R_0$ puede recordar cualquier valor natural que se encuentre en el registro $X$. A continuación crearemos un algoritmo capaz de crear $n$ registros distintos para un $n \in \mathbb{N}$ fijo.

(Primer entorno de algoritmo para crear $n - registros$ diferentes con $n \geq 2$). $MR(n)$ (Make registers)

1:: $P(P, X_1)$
2:: $L(P, X_1)$
3:: $EL(P, X_1, R_1)$
4:: $PL(P, X_1; P, R_n)$
5:: $C(P, X_1)$
6:: $EC(P, X_1; c_1)$
7:: $EP(R_n, c_1; X_n, Y_n)$
8:: $Angle(X_n, P, X_1, R_{n-1})$
9:: $EP(R_{n-1}, c_1, X_{n-1}, Y_{n-1})$
10:: $Angle(X_{n-1}, P, X_1, R_{n-2})$
11:: $EP(R_{n-2}, c_1, X_{n-2}, Y_{n-2})$
$\vdots$ $\vdots$
2n+2:: $Angle(X_3, P, X_1, R_2)$
2n+3:: $EP(R_2, c_1, X_2, Y_2)$

Donde $PL(P, Q, P, L)$ y $Angle(A, B, C, L)$ son los algoritmos descritos en los ejemplos 1 y 2 respectivamente de la sección 2.2
Este algoritmo crea $n$ registros diferentes donde cada $X_n$ inicia con un valor vacío y todos pertenecen a la misma circunferencia $c_1$.

*Ejemplo* 5 (Creación de $n = 5$ registros).

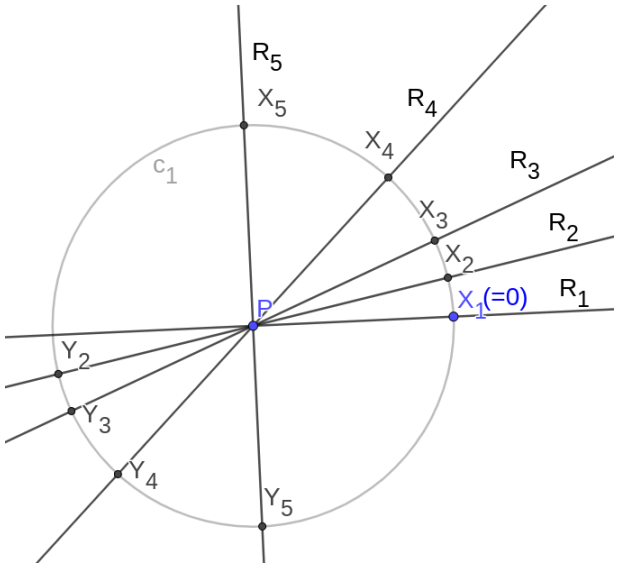


Figura 5: MR(5)

## 4.2. Translación de instrucciones de URM a máquinas Euclídeas

En [9] toman un conjunto más reducido de operaciones de las $URM$; pero siguiendo nuestra construcción realizada en la sección anterior vamos a emular el conjunto de 4 operaciones que obtuvimos, para poder mostrar la Turing completez de las máquinas Euclideanas. Notaremos por (operación(n)) al algoritmo que traslada la operación de las URM a las máquinas de Euclides.

(O(n)). Mover el punto $X_n$ a la intersección de la recta $R_n$ con $c_1$

1:: $D(X_n)$ 2:: $EP(R_n, c_1; X_n, Y_n)$ 3:: $D(Y_n)$

(P(n)). Bisecar el segmento $\overline{PX_n}$, y nombrar el punto de la mitad $X_n$

1:: $C(P, X_n)$
2:: $EC(P, X_n; C_1)$
3:: $C(X_n, P)$
4:: $EC(X_n, P; C_2)$
5:: $EP(C_1, C_2; P_1, P_2)$
6:: $L(P_1, P_2)$
7:: $EL(P_1, P_2, L)$
8:: $D(X_n)$
9:: $EP(L, R_n; X_n)$
10:: $D(P_1, P_2, C_1, C_2, L)$

(D(n)). Si $X_n$ no es el punto de intersección entre $c_1$ y $R_n$ duplicar el segmento $\overline{PX_n}$ y nombrar el punto $X_n$

1:: $C(X_n, P)$
2:: $EC(X_n, P; C_1)$
3:: $EP(R_n, C_1; P_1, P_2)$
4:: $C(P, X_n)$
5:: $EC(P, X_n; C_2)$
6:: $P_1 \in C_2 : 12$
7:: $D(X_n)$
8:: $PL(P, P_1, P_1; L)$
9:: $EP(R_n, L, X_n)$
10:: $D(P_1, P_2, L, C_1, C_2)$
11:: $Moff$
12:: $D(X_n)$
13:: $PL(P, P_2, P_2; L)$
14:: $EP(R_n, L; X_n)$
15:: $D(P_1, P_2, L, C_1, C_2)$

(J(m)(k)). $X_n$ un registro vacío, vamos a verificar $X_n$ se encuentra en el circulo $C(P, X_m)$ al ser $X_n$ vacío, el circulo $C(P, X_n)$ tiene el radio mayor, y si $X_n$ esta en el circulo $C(P, X_m)$ los circulos deben tener el mismo radio y por lo tanto $X_m$ es vacío.

1:: $D(X_n)$
2:: $EP(R_n, c_1; X_n, Y_n)$
3:: $D(Y_n)$
4:: $C(P, X_m)$
5:: $EC(P, X_m; C_1)$
6:: $X_n \in C_1 : 9$
7:: $D(C_1)$
8:: $Moff$
9:: $D(C_1)$
10:: $P \in c_1 : k$

Así tenemos que toda máquina de registro ilimitado puede ser emulada por una máquina de Euclides, y así obtenemos el siguiente teorema:

*Teorema* 2 (Completez de las Maquinas Euclideanas). Toda función calculable por una máquina de Turing, es calculable por una máquina de Euclides.

*Demostración.* Por Colorario 2 de la sección 3.2 tenemos que toda función calculable por una máquina de Turing es calculable por una URM; y por lo anterior tenemos que toda URM puede ser emulada por una máquina de Euclides, se tiene que, toda función calculable es calculable por una máquina de Euclides. Por lo tanto, la máquina de Eclides es Turing-completa. □

Aunque no hemos definido formalmente que es una función calculable por una máquina de Euclides, ya que no hay necesidad, podemos preguntarnos por si toda función calculable por una máquina de Euclides, lo es también por una máquina de Turing; la respuesta a esta pregunta es sí, por la Tesis de Church [2], sin embargo nuestra máquina de Euclides requiere de entradas con una exactitud infinita, y por lo tanto simular sus operaciones a través de una máquina de Turing o por las mismas URM, resulta de gran complejidad, y en consecuencia también lo es para una prueba formal.

# 5. Computabilidad de los números construibles

A continuación vamos a ver una conexión entre los números computables bajo las máquinas de Turing y los números construibles con regla y compás. Asociaremos también los puntos construidos por una máquina de Euclides con un campo de números computables, algebráico y enumerable, que nos ayudará a darle coordenadas a los puntos construidos bajo las máquinas de Euclides.

## 5.1. Coordenadas de puntos

En la sección anterior mostramos que cualquier máquina de registro ilimitado puede ser simulada mediante una máquina de Euclides, es decir se tiene una codificación adecuada de los algoritmos de las URM que existe en el plano Euclideano. Aunque no hemos definido explícitamente funciones Euclídeo-computables, podríamos ver este concepto directamente sobre el plano Cartesiano, y esto es lo que evidenciará esta sección.

Los algoritmos con máquinas de Euclides requieren de puntos en el plano Euclideano, vamos a llevar estos puntos a un comportamiento más algebraico; las máquinas de Euclides pueden simular el plano cartesiano tomando la operación $P(P_1, P_2)$, donde, sin pérdida de generalidad, tenga coordenadas $P_1 = (0, 0)$ en el plano cartesiano y $P_2 = (1, 0)$, luego construimos la recta $L(P_1, P_2)$ la etiquetamos bajo $EL(P_1, P_2, x)$ y obtenemos una perpendicular que pase por $P_1$ y se etiquetará con el nombre $y$. Así estamos interesados en que estos puntos tengan coordenadas con un buen comportamiento (hablando desde el punto computacional). Por conveniencia vamos a tomar el campo de los números algebraicos que son computables y enumerables $\mathbb{A}$.

Cuando utilizamos la operación $P(A, B)$ para las máquinas de Euclides, estos puntos $A = (x_1, y_1)$ y $B = (x_2, y_2)$ los ubica aleatoriamente en el plano Cartesiano; de esta manera es posible que las coordenadas de $A$ o $B$ no se encuentren en el campo $\mathbb{A}$, entonces como necesitamos una elección completamente libre de puntos que se encuentren en un campo, podemos hacer una extensión finita del campo $\mathbb{A}$, es decir, para una máquina de Euclides que inicie con los puntos $P_1 = (x_1, y_1)$,...,$P_n = (x_n, y_n)$ podemos obtener la extensión del campo $\mathbb{A}$ como $\mathbb{A}(x_1, y_1, ..., x_n, y_n)$. Este campo es numerable ya que $\mathbb{A}$ es numerable y $\mathbb{A}(x_1, y_1, ..., x_n, y_n)$ es una extensión finita; por lo tanto, podemos enumerar los elementos de $\mathbb{A}(x_1, y_1, ..., x_n, y_n)$ usando números naturales, de donde cualquier problema de construcción de puntos en el plano Euclideano puede ser visto como algún cálculo de números naturales. Por lo que, para cualquier punto con sus coordenadas en $\mathbb{A}(x_1, y_1, ..., x_n, y_n)$, le corresponde un número natural el cual llamamos índice.

*Proposición* 6 (Clausura). Para cualquier máquina de Euclides, con puntos iniciales $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$,..., $P_n = (x_n, y_n)$, todos los puntos que construye la máquina, tienen coordenadas en el campo $\mathbb{A}(x_1, y_1, ..., x_n, y_n)$

*Demostración.* Sea $E$ una máquina de Euclides cuyos puntos iniciales son $P_1 = (x_1, y_1), ..., P_n = (x_n, y_n)$; la máquina construye puntos a través de la operación $P(A, B)$ y $EC(O_1, O_2; A, B)$. Esto es, que toda construcción de puntos hecha por $E$ distinta a los puntos iniciales, es realizada encontrando intersección de circunferencias y rectas que se obtienen a partir de puntos ya construidos. De donde, podemos obtener las coordenadas de estos nuevos puntos construidos de los puntos construidos anteriormente solucionando sistemas de ecuaciones de, a los más, segundo grado, y por lo tanto los nuevos puntos pertenecen también a $\mathbb{A}(x_1, y_1, ..., x_n, y_n)$. □

Así obtenemos el siguiente corolario.

**Corolario 3** (Computabilidad de los construibles bajo las máquinas de Euclides). *Para todo número construible con regla y compás existe una máquina de Euclides que lo obtiene (construye). Esto es, todo número construible es Euclídeo-calculable.*

Recordemos que por la Turing-Completez de las máquinas de Euclides, si un número es computable bajo las máquinas de Turing lo es también bajo las máquinas de Euclides. Así, si tomamos $Comp(\mathbb{R})$ el conjunto de todos los números computables por las máquinas de Turing y $F(\mathbb{R})$ el conjunto de todos los reales construibles con regla y compás, por el corolario anterior se tiene que $F(\mathbb{R}) \subset Comp(\mathbb{R})$; es fácil ver que $Comp(\mathbb{R}) \not\subset F(\mathbb{R})$ ya que por ejemplo $\pi$ es un número computable el cual no es construible. Este resultado es evidente desde un principio, pero a través de lo realizado se expone un poco más formal. Igualmente, si $\alpha$ es un real no computable por una máquina de Turing entonces $\alpha$ no es construible con regla y compás. Tenemos una relación más entre números computables con máquinas de Turing y números construibles con regla y compás derivada de la definición de número computable con máquinas de Turing aún sin definir explícitamente una función Euclídeo-computable.

**Corolario 4** (Número computable). *Sea $x \in \mathbb{R}$, si $x$ es un número Turing-computable entonces existe una función $f$ computable con una máquina de Euclides tal que dado un $\epsilon > 0$ la función $f$ calcula un número construible $\alpha$ tal que $|x - \alpha| \leq \epsilon$.*

*Demostración.* Si $x$ es Turing computable, entonces existe una función $f$ Turing computable tal que dado $\epsilon > 0$, $f$ calcula un racional $q$ tal que $|x - q| \leq \epsilon$. Por la completez de la máquinas de Euclides entonces $f$ es Euclídeo-computable, y ya que todo racional es construible con regla y compás, entonces $q$ es un número construible con regla y compás, por lo tanto existe $f$ Euclídeo-computable tal que dado un $\epsilon > 0$ la función $f$ construye un número $q = \alpha$ tal que $|x - \alpha| \leq \epsilon$ □

# 6. Problemas no decidibles con Máquinas Euclideanas

Daremos uno de los resultados más importantes en esta sección que nos permitirá observar problemas que no son decidibles con una perspectiva geométrica diferente a la perspectiva del problema de

la parada, que se usa con frecuencia en este contexto. Para esto definiremos relaciones Euclídeo-decidibles y veremos cómo la trisección de un ángulo, uno de los problemas de construcción con regla y compás clásicos, no es decidible con máquinas de Euclides.

Un problema de decisión consiste en una pregunta de sí o no (Falso o verdadero) que va a ser decidible si existe un algoritmo finito que nos resuelva la pregunta correctamente, y no decidible (indecidible) si no existe tal algoritmo que me resuelva el problema. Vamos a definir formalmente cuando una relación es Euclídeo-decidible tal como en [9].

*Definición* 3 (Euclídeo-decidible). Diremos que una relación $\mathcal{R}$ sobre $\mathbb{N}$ es Euclídeo-decidible si existe una máquina $E_{\mathcal{R}}$ tal que para todo $k_1, k_2, ..., k_n, k \in \mathbb{N}$ la relación $\mathcal{R}(k_1, ..., k_n, k)$ se cumple, si y solo si, $E_{\mathcal{R}}$, con puntos iniciales $P_{k_1}, P_{k_2}, ..., P_{k_n}$, construye el punto $P_k$. Tenemos que $P_n$ denota el punto en el plano cartesiano con coordenadas en $\mathbb{A}(x_{k_1}, y_{k_1}, ..., x_{k_n}, y_{k_n})$ con índice $n$.

## 6.1. Trisección de un ángulo

Vamos a retomar parte de lo expuesto en la introducción acerca de uno de los problemas clásicos, la imposibilidad de construcción con regla y compás.

Como vimos, trisecar un ángulo con regla y compás no siempre es posible, esto lo vimos para el ángulo de 60°. Pero ¿qué significa esa imposibilidad en términos computacionales?. La respuesta a esta pregunta nos conduce hacia los siguientes teoremas.

*Proposición* 7. El problema de trisecar el ángulo no puede ser solucionado por ninguna máquina de Euclides.

*Demostración.* Supongamos que existe una máquina de Euclides $E$ tal que dado cuales quiera puntos iniciales $P_n, P_m, P_r$ cuyas coordenadas están en el campo $\mathbb{A}(x_n, y_n, x_m, y_m, x_r, y_r)$, la máquina $E$ construye un punto $P_s$ el cual cumple que $\angle P_m P_n P_r = 3\angle P_m P_n P_s$. Como los puntos iniciales son cualesquiera, entonces se tiene la posibilidad de que $\angle P_m P_n P_r = 60°$; de donde existe un algoritmo finito compuesto por las operaciones de las máquinas de Euclides tal que calcula un punto $P_s$ que cumple que $60° = 3\angle P_m P_n P_s$, contradiciendo así la proposición 1 de la sección 1. Por lo que tal máquina $E$ no existe. □

Este problema de la insolubilidad lo podemos ver como un problema de la no decibilidad bajo la definición 3.

*Teorema* 3. Si definimos la relación de trisección $T : \mathbb{N}^4 \to \mathbb{N}$ si y solo si

$$\angle P_m P_n P_r = 3\angle P_m P_n P_s$$

donde $P_m, P_n, P_r, P_s$ corresponden a los puntos de los índices $m, n, r, s$ respectivamente, entonces $T$ no es Euclídeo-decidible.

Para mostrar este teorema la idea es ver que la relación $T$ se cumple para todo punto cuyas coordenadas están en el campo de los números algebraicos y entonces así mostrar que para toda máquina $E_T$ existen $m,n,r,s$ tal que $T(m,n,r,s)$ se cumple y $E_T$ con puntos iniciales $P_m, P_n, P_r$ no puede construir $P_s$.

*Proposición* 8. Sean $A,O,B$ puntos cualesquiera con coordenadas algebraicas; entonces existe el punto $C$ con coordenadas algebraicas, tal que

$$\angle AOB = 3\angle AOC$$

*Demostración.* Sea $A,O,B$ puntos con coordenadas algebraicas, sin perdida de generalidad supongamos que $O$ tiene coordenadas $(0,0)$, ya que siempre podemos usar una translación con parámetros algebraicos para obtener tal situación. Sean $(x_1,y_1),(x_2,y_2)$ las coordenadas de $A$ y $B$ respectivamente, construimos las líneas $\overline{OA},\overline{OB}$, esto es $(x_1t,y_1t),(x_2t,y_2t)$ con $t\in\mathbb{R}$. Si tomamos $\alpha=\angle AOB$ entonces se tiene que

$$cos(\alpha)=\frac{(x_1x_2)+(y_1y_2)}{((x_1^2+y_1^2)(x_2^2+y_2^2))^{1/2}}$$

Por propiedades de coseno podemos encontrar $cos(\alpha/3)$ de la ecuación

$$cos(\alpha)=4cos^3(\frac{\alpha}{3})-3cos(\frac{\alpha}{3})$$

Lo cual significa que $cos(\frac{\alpha}{3})$ es un número algebraico.
Por otra parte tenemos que $sin^2(\frac{\alpha}{3})=1-cos^2(\frac{\alpha}{3})$; de donde $sin(\frac{\alpha}{3})$ es un número algebraico; así el punto $C$ que necesitamos encontrar, debe tener coordenadas

$$x_C=x_1cos(\frac{\alpha}{3})-y_1sin(\frac{\alpha}{3})$$
$$y_C=x_1sin(\frac{\alpha}{3})+y_1cos(\frac{\alpha}{3})$$

Donde al ser $x_1,y_1,cos(\alpha/3),sin(\alpha/3)$ números algebraicos, entonces $x_C,y_C$ son números algebraicos □

Si tomamos $O=(0,0)$, $A=(1,0)$ y construimos las circunferencias $C_1$ con centro en $O$ y que pasa por $A$ y $C_2$ con centro en $A$ que pasa por $O$ hallamos los puntos $B,B'$ con coordenadas algebraicas como intersección de $C_1$ con $C_2$; además se tiene que $\angle AOB=60°$ por lo que por la proposición anterior existe $C$ con coordenadas algebraicas, tal que $\angle AOB=3\angle AOC$. Ahora si suponemos que los índices de $O,A,B,C$ son $o,a,b,c\in\mathbb{N}$ respectivamente, entonces la relación $T(o,a,b,c)$ se cumple. Pero no existe una máquina de Euclides tal que con los puntos iniciales $O,A,B$ pueda construir el punto $C$ por la proposición 8. Siendo esto lo que se quería para mostrar el Teorema 3.

*Teorema* 4 (La no decibilidad de la trisección). La relación $T$ definida en el teorema 3 no es $URM - computable$ (tampoco Turing-Computable)

*Demostración.* Vamos a mostrar que si $T$ fuera decidible, entonces podríamos decidir la igualdad entre dos números algebraicos mediante un procedimiento, es decir, que existe una máquina de Turing capaz de calcular toda su expresión decimal para comparar los dos números. Pero este procedimiento no puede ser realizado en un tiempo finito.

Para mostrar esto, vamos a tomar dos puntos con coordenadas algebŕaicas cualesquiera $P_n$ y $P_m$, con índices $n, m \in \mathbb{N}$ respectivamente. Tomemos $P_{u_1}$ y $P_{u_2}$ con índices $u_1, u_2$, los mínimos puntos tales que su trisección puede ser hecha por $P_m$; esto puede ser representado por

$$u_1 = (\mu_u T((u)_1, 0, (u)_2))_1,$$
$$u_2 = (\mu_u T((u)_1, 0, (u)_2))_2.$$

Ahora la condición $T(u_1, 0, u_2, n)$ nos dice si el punto $P_n$ triseca el ángulo $\angle P_{u_1} P_0, P_{u_2}$; además note que la condición $T(n, 0, m, m)$ se cumple si $P_m$ se encuentra en la misma recta que $P_0, P_n$. Por lo que si las dos condiciones anteriores se cumplen tenemos que $P_n$ y $P_m$ se encuentran en la misma recta que pasa por el punto $P_0$; pero esto no nos está queriendo decir que $P_m$ y $P_n$ son el mismo punto; así que necesitamos otra condición. Si suponemos además que $P_n$ y $P_m$ se encuentran en la recta $P_{u_1}, P_n$, si la relación $T(n, u_1, m, m)$ se cumple, entonces $P_m = P_n$, es decir $P_m = P_n$, si y solo si, las tres condiciones anteriores se cumplen simultáneamente. De esto último, si suponemos que $T$ es una relación decidible, entonces podemos decidir cuando dos números algebraicos son idénticos ya que se tiene $P_n = P_m$ si, y solo si, las condiciones $T(u_1, 0, u_2, n) \wedge T(n, 0, m, m) \wedge T(n, u_1, m, m)$ se cumplen. Lo cual nos indica que, una máquina de Turing sería capaz de calcular toda la expresión de un número algebraico para compararlo con otro en un tiempo finito y esto es una contradicción. Con lo que $T$ no es decidible. □

# 7. Conclusión

Hemos mostrado que saber si un ángulo puede o no ser trisecado es un problema no decidible bajo las máquinas de Turing y bajo las máquinas definidas de Euclides, para ello encontramos un problema clásico del álgebra abstracta que puede ser expresado como un problema de no decibilidad de la teoría de computación el cual no tiene relación con el problema de la parada. También hemos visto que la relación intuitiva entre los números computables y los números construibles puede ser enunciada de una manera formal, y hemos visto su relación íntima.

La Turing-completez de las máquinas de Euclides nos confirma el poder computacional de los instrumentos regla y compás; lo que podría ser utilizado para estudiar las órbitas de los planetas, calcular

áreas e incluso calcular series de Fourier al tratar las series de Fourier como suma de círculos cuyos radios se reducen tal como en [3] y en la figura 6, donde por ejemplo, tomando la serie de Fourier $\sum_{n=0}^{\infty} \frac{4}{(2n+1)} sin[(2n+1)x]$ y realizando las primeras tres iteraciones para calcular $f(\pi/4)$, con los puntos iniciales de la máquina de Euclides $A, X$, podemos construir los segmentos rojos y los segmentos verdes que se encuentran en la figura 6. Sumando la longitud de los rojos y restando la longitud de los verdes por medio de la suma de construibles, se obtiene la longitud del segmento naranja $\ell = 0{,}891$ la cual es una aproximación al valor de la suma de Fourier evaluado en el punto $\pi/4$.

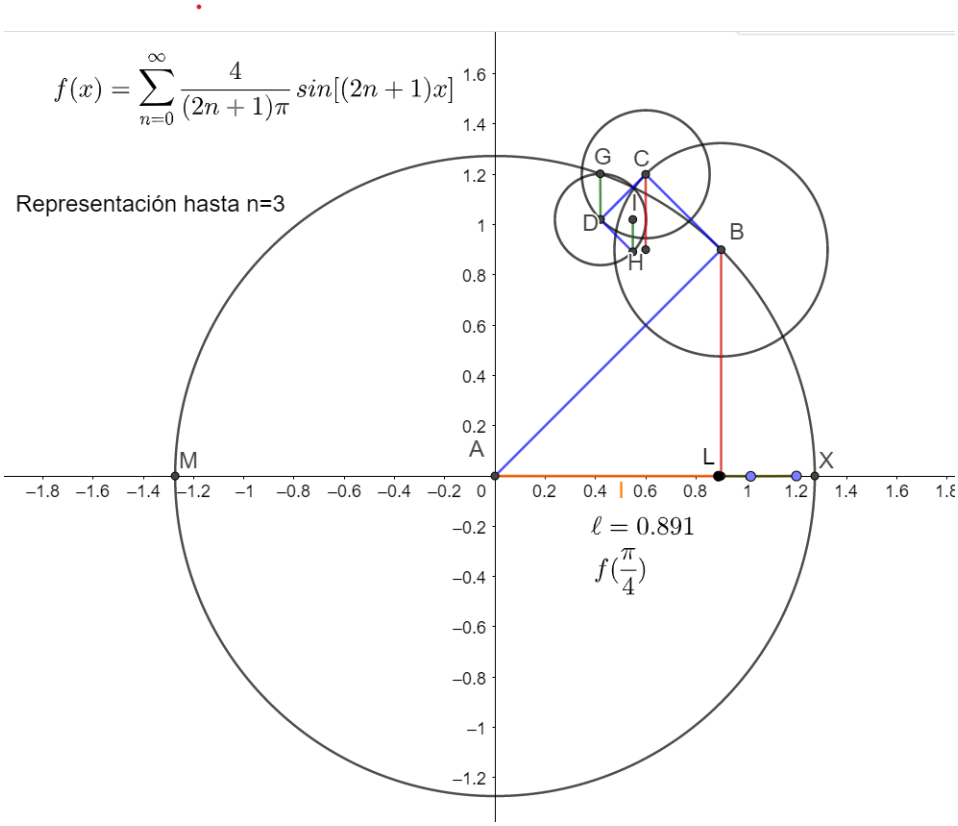


Figura 6: Serie de Fourier en ME

Con esta última aplicación de las máquinas de Euclides en la construcción de las Series de Fourier damos fin a la discusión acerca de éstas máquinas. Sin embargo quedan preguntas naturales como ¿qué ocurre si cambiamos los instrumentos de la regla y compás por regla y, por ejemplo, un hiperbolígrafo (un instrumento capaz de construir hipérbolas), y construimos una máquina que trabaje con estos instrumentos? ¿Sería posible que el problema de la trisección de un ángulo sea decidible bajo esta nueva máquina?, con esto culminamos nuestro trabajo.

# Referencias


[1] George S. Boolos. *Computability and logic*. Cambridge, 2002.

[2] B. Jack Copeland. «The Church-Turing Thesis». En: *The Stanford Encyclopedia of Philosophy*. Ed. por Edward N. Zalta. Summer 2020. Metaphysics Research Lab, Stanford University, 2020.

[3] Bilgecan Dede. *FFuyye Serisi ve Görüntüden Ses Çıkarmak*. Bilim Ne GuzelLan, feb. de 2017. url: https://bilimneguzellan.net/fuyye-serisi/ (visitado 31-07-2020).

[4] Xavier Caicedo F. *Elementos de lógica y calculabilidad*. Departamento de matemáticas Universidad de los Andes, 1989.

[5] John B Fraleigh. *First course in abstract algebra 7th edition*. ADDISON WESLEY, 2003.

[6] A.N Kostovski. *Construcciones geométricas mediante un compás, segunda edición*. Editorial MIR Moscú, 1984.

[7] Maria Juanes Linares. *Libro I de los elementos de Euclides*. 2013. url: http://newton.matem.unam.mx/geometria/menulibro_m.html (visitado 05-01-2020).

[8] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Pretice-Hall, INC, 1967.

[9] Jerzy Mycka, José Félix Costa y Francisco Coelho. «The Euclid abstract machine». En: *Jornal of Unconventional Computing* (dic. de 2008), págs. 6-27. url: https://www.researchgate.net/publication/210305222_The_Euclid_Abstract_Machine.

[10] J. C. Shepherdson y H.E Sturgis. «Computability of Recursive Functions». En: *University of Bristol, England* (1964).

[11] A. M. Turing. «ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM». En: (nov. de 1936), págs. 230-265. url: https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf.