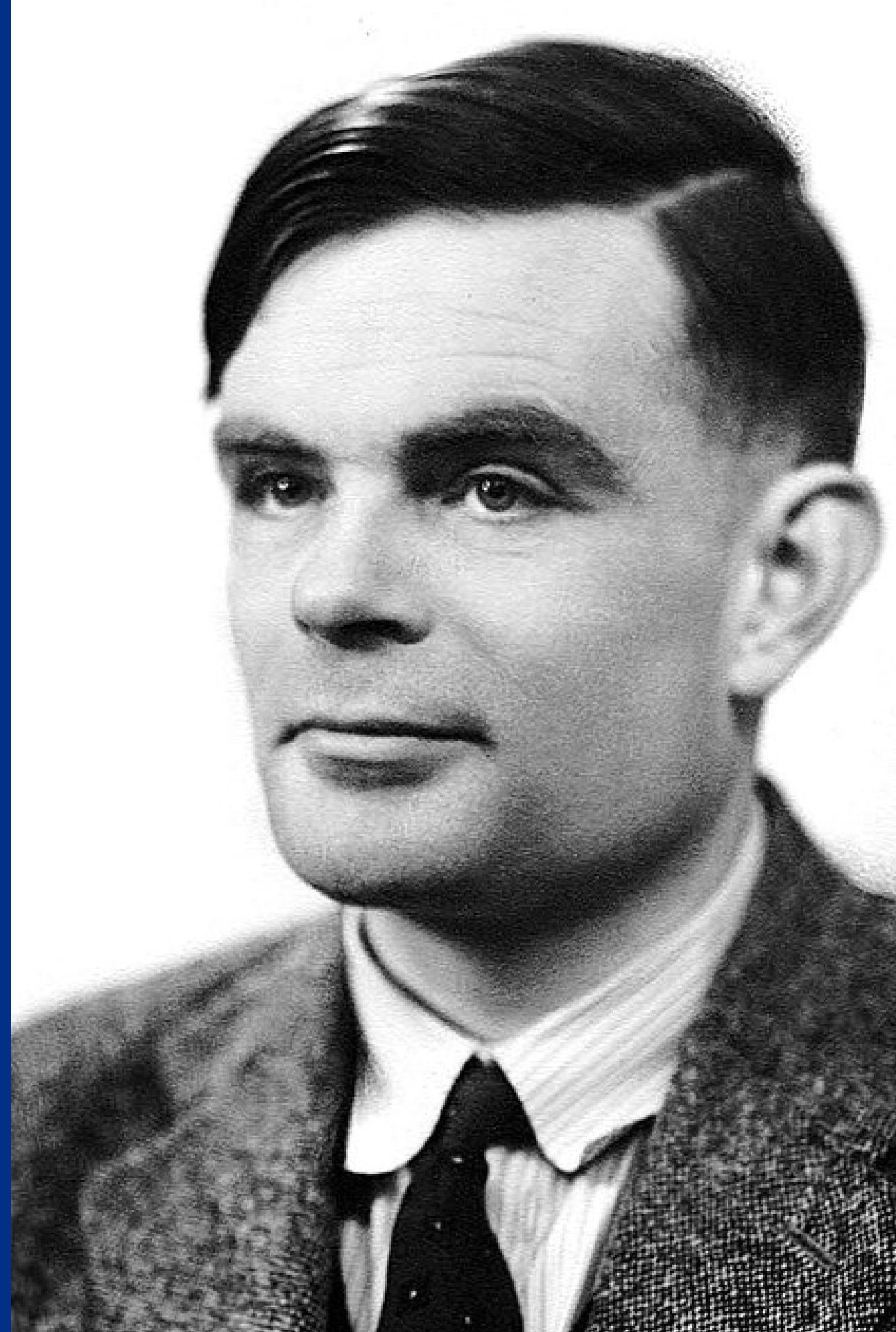


Introducción a la Teoría de la Computación



UNIVERSIDAD
DE CHILE

Arturo Merino



Plan

Plan

- **Plan:**
 - Cosas administrativas
 - Introducción
 - Lenguajes formales

Cosas administrativas

Administratrivia

- Secciones coordinadas
- 3 controles: S6 - S10 - S14
 - En horario auxiliar (en principio)
 - Reemplazado por examen en casos justificados
- 3 tareas:
 - 1 pregunta + presentación
 - Reemplazado por el control en casos justificados
- 1 examen: fecha a definir.
- Medio de comunicación: U-Cursos.
- `amerino.cl/teoria`

Cálculo de notas

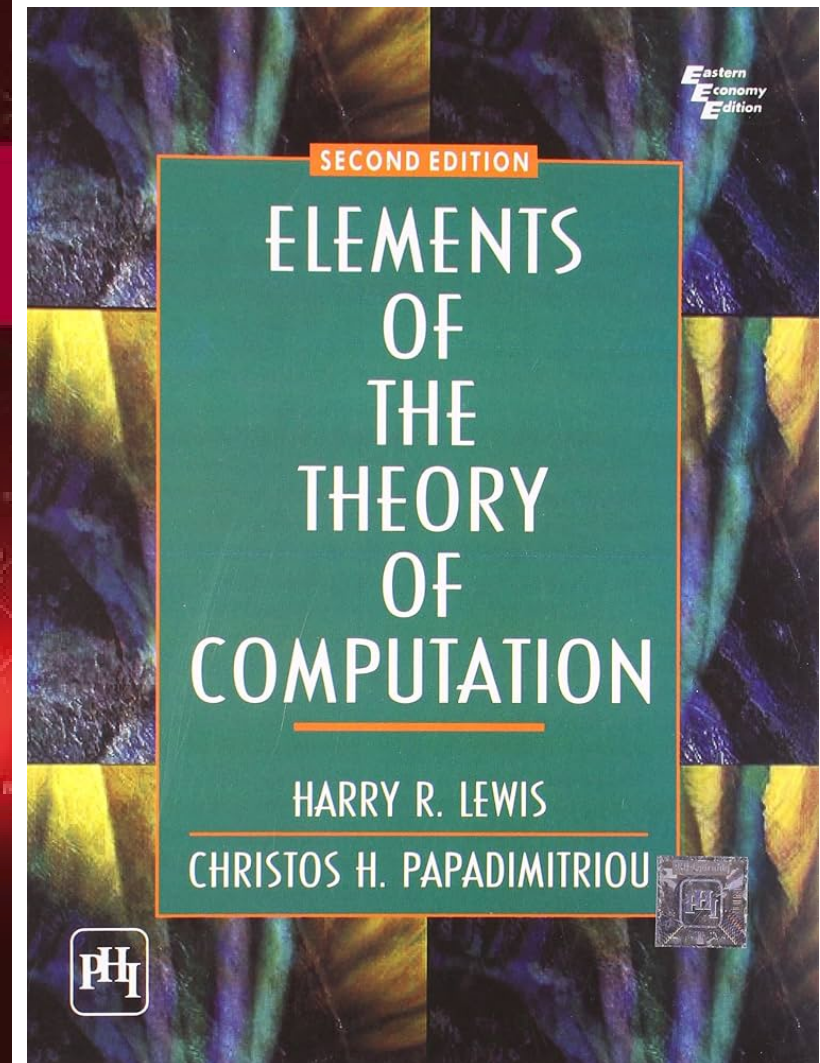
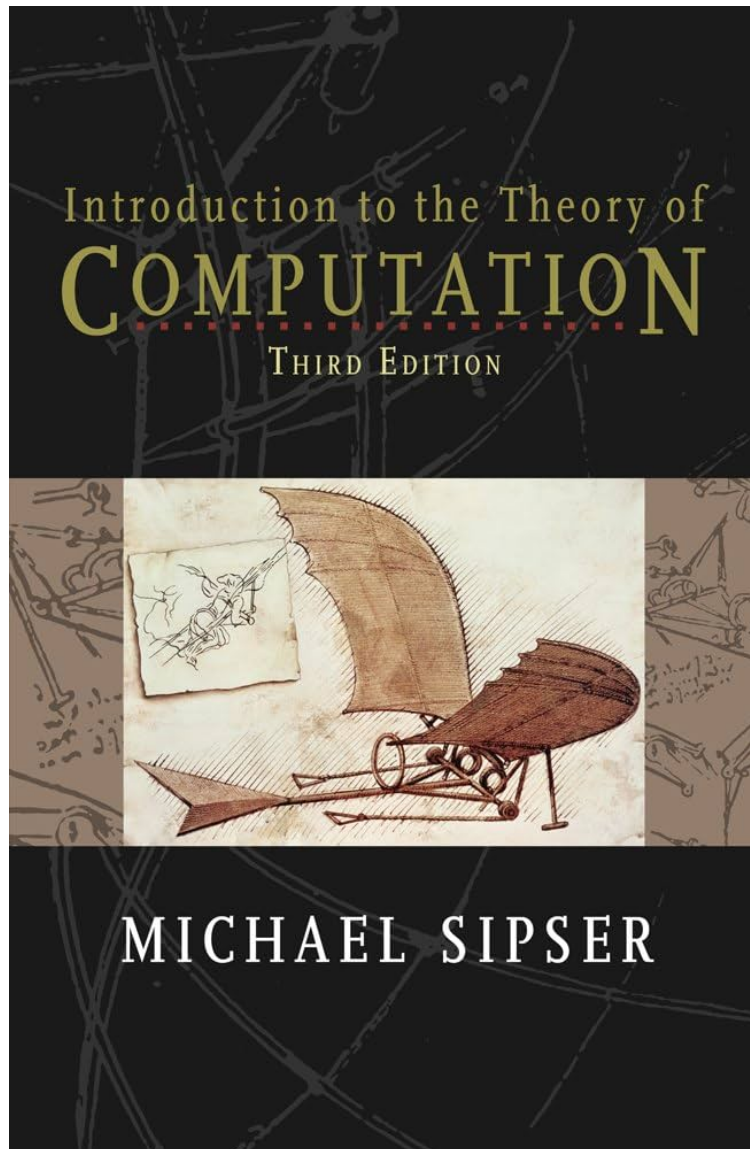
- **NT**: Promedio de las tareas
- **NCon**: Promedio de los controles
- Si **NCon** ≥ 5.5 , está exento del examen.
- **NCat**: $40\% \cdot \text{Examen} + 60\% \cdot \text{NCon}$.
- **Para aprobar**: Ambos **NE** ≥ 4.0 y **NCat** ≥ 4.0 .

$$\text{NFinal} = 80\% \cdot \text{NCat} + 20\% \cdot \text{NE}.$$

Metodología

- Secciones coordinadas
- Cada clase: lecturas, slides, handouts.
- Estructura:
 - 12:00 - Catedra
 - 13:00 - Resolución de problemas y dudas
 - 13:25 - Discusión
- Handout - 3 problemas
 - P1 - Tipo
 - P2 - Asimilación o exploración
 - P3 - Mayor dificultad

Lecturas



+ Apunte G. Navarro

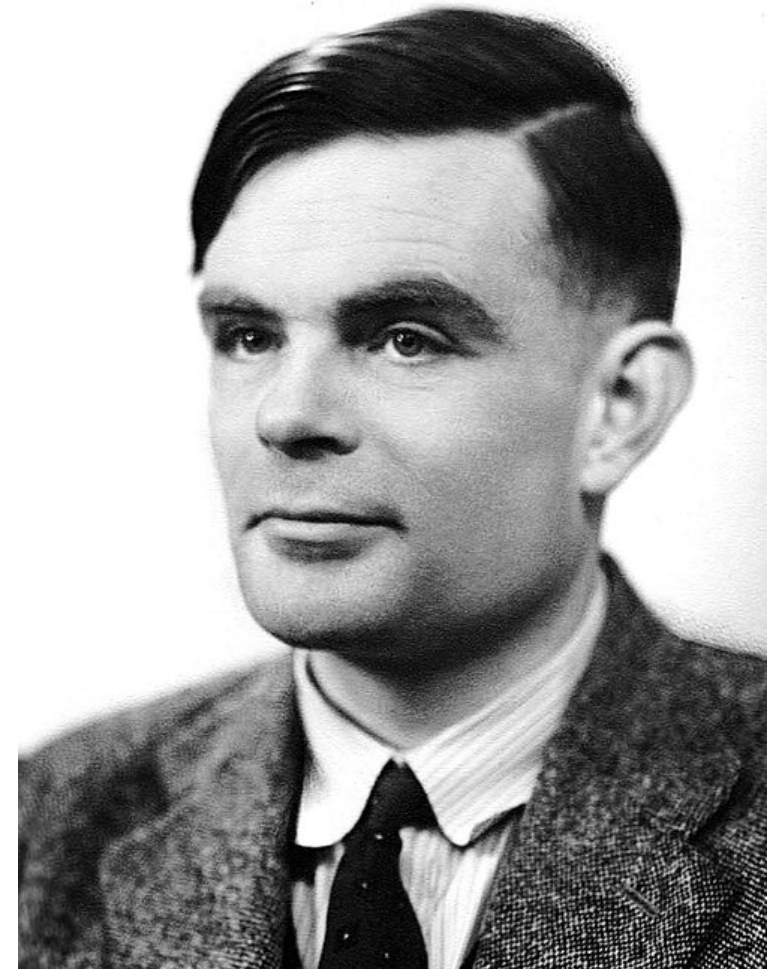
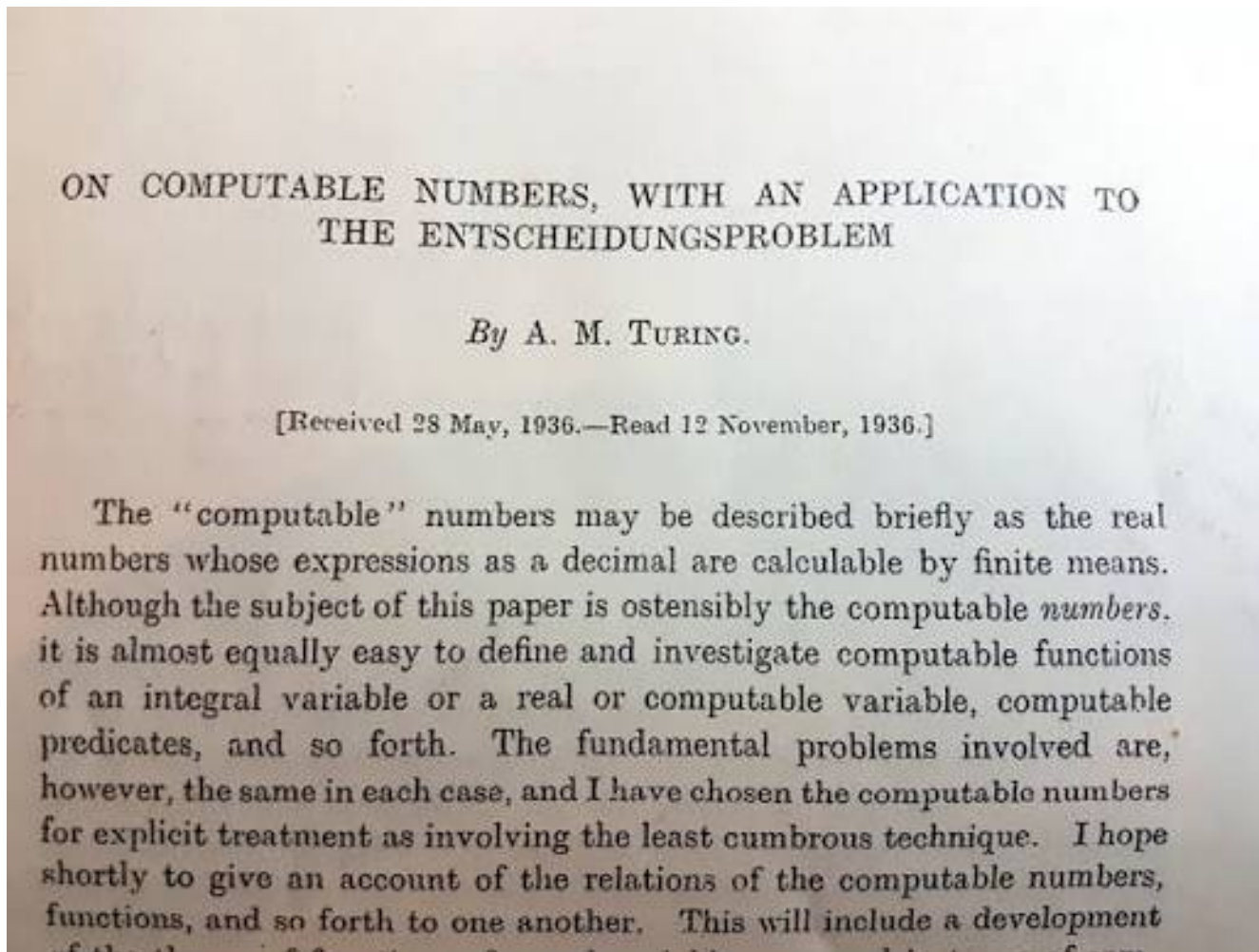
Introducción

Estudio teoría

- ¿Qué es la teoría de la computación?
- Busca entender las capacidades y limitaciones de la **computación**.
- ¿Qué significa computar? ¿Qué es un algoritmo?
¿Qué es un computador?
- ¿Qué cosas puedo computar eficientemente?
¿Qué cosas no puedo computar eficientemente?
- ¿Por qué no puedo computar algunos problemas eficientemente?
- ¿Puedo computar todas las cosas?
- ¿Qué rol juegan partes del computador?

El modelo de computación

- Queremos un modelo que no dependa del computador, ni lenguaje
- Modelo robusto ante cambios
- Que represente de manera fidedigna los componentes del computador
- Concepto de *software*, data de lógicos de 1930.



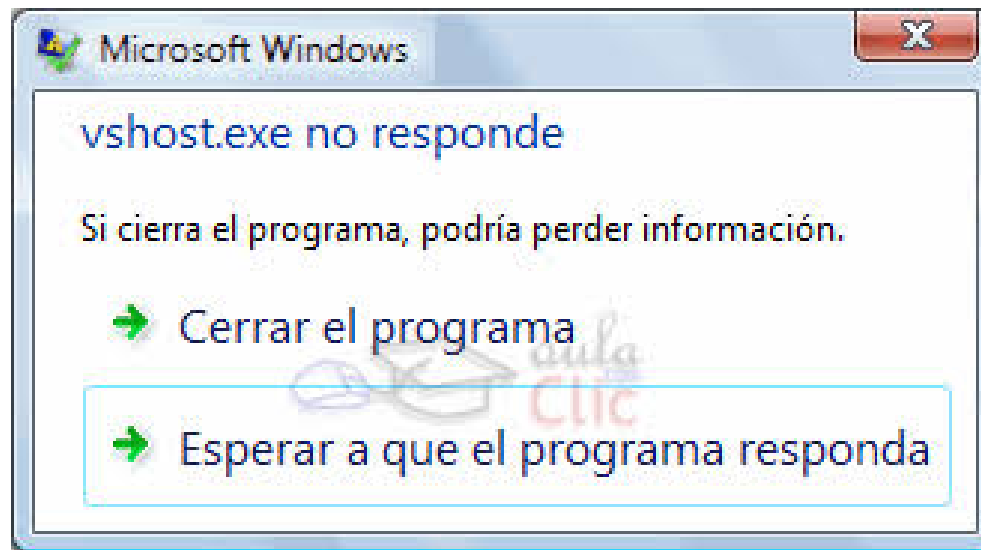
Complejidad computacional

- ¿Qué cosas se pueden hacer rápido en nuestro modelo?
- Ordenar, encontrar un árbol generador
- Modelo word-RAM $\rightarrow$ ¿Algo más robusto?
- Euleriano v/s Hamiltoniano
- NP-Complejidad, P v/s NP

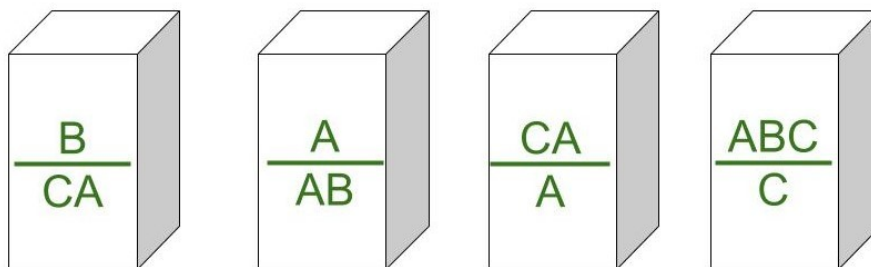


Computabilidad

- ¿Hay algo que no se puede hacer en nuestro modelo?
- ¿Hay algo **útil** que no se puede hacer en nuestro modelo?

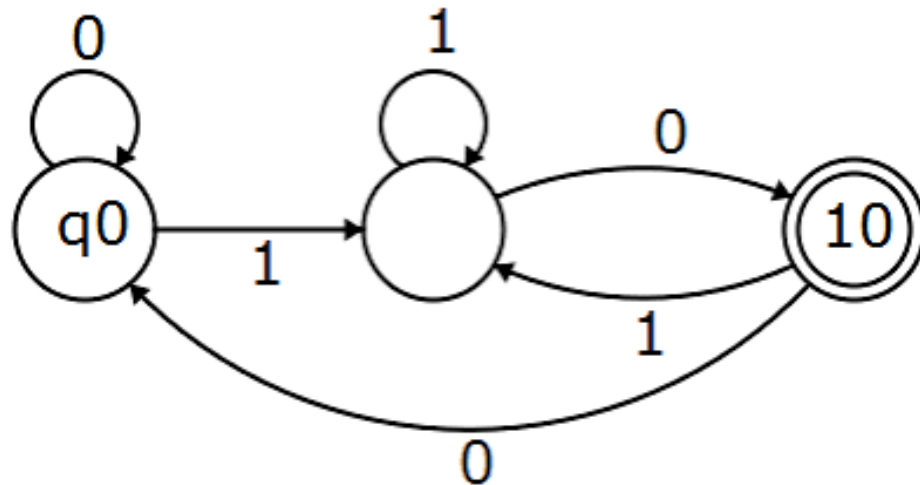


- ¿Hay algo **simple** que no se puede hacer en nuestro modelo?

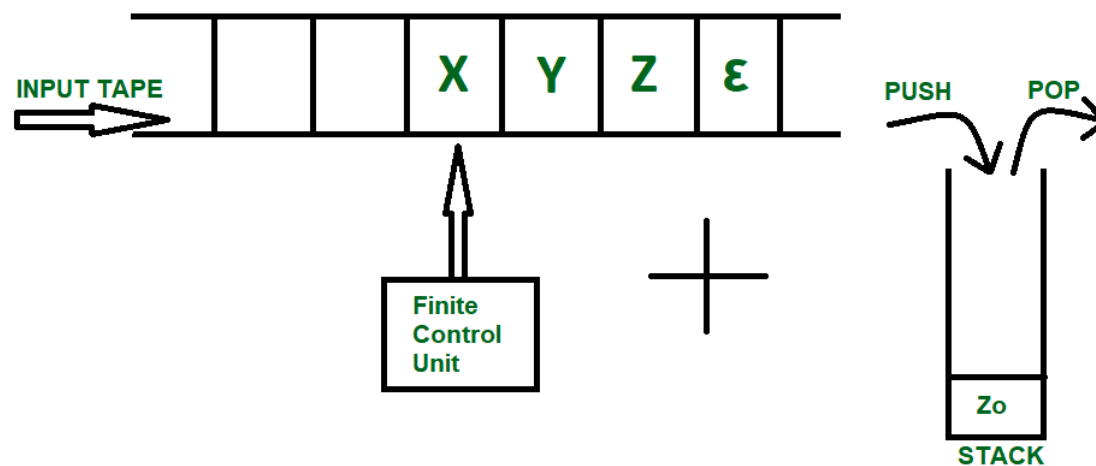


Teoría de automatas

- ¿Qué rol juega la RAM?



- ¿Que rol juega una pila?



- Camino hacía la máquina de Turing, con aplicaciones interesantes

Temas del Curso

- Automatas regulares (computadores sin RAM)
- Automatas de pila (computadores con acceso a pila)
- Máquinas de Turing (computadores)
- (In)computabilidad.
- Complejidad

Nuestras herramientas

- Análisis formal matemático
- Técnicas de matemáticas discretas
- Inspiración algorítmica.

¿Por qué estudiar teoría?

- Razonar sobre la computación requiere un modelo
- Ideas prácticas: automatas, gramáticas, reducciones, etc...
- Entender los límites de la computación
- Dual a algoritmos
- Alimenta a las áreas de la computación con modelos, limites, herramientas de análisis, etc...
- Bonito

Lenguajes formales

Problemas de decisión

- Dos objetivos: modelo de computación + clasificar problemas.
- Se puede, se puede de forma eficiente, etc...
- ¿Cómo representamos un problema?
- Problemas de decisión: `string` $\rightarrow$ `bool`
 - ¿Ordenar una lista?
 - ¿Camino más corto?
- Múltiples descripciones

Input: Un natural $n \in \mathbb{N}$.

Decidir: ¿ n es par?

Input: Un natural $n \in \mathbb{N}$.

Decidir: ¿ n^2 es par?

Problemas de decisión

- Múltiples descripciones

Input: Un natural $n \in \mathbb{N}$.

Decidir: ¿ n es par?

Input: Un natural $n \in \mathbb{N}$.

Decidir: ¿ n^2 es par?

- Un problema es representado por sus instancias positivas

- ¿El problema anterior?

- Ej:

Input: $a, b, c \in \mathbb{Z}$

Decidir: ¿ $a + b = c$?

Input: Un grafo G .

Decidir: ¿ G es conexo?

- ¿Por qué problemas de decisión?

- Teoría limpia

- Bastante generales $\rightarrow$ búsqueda, optimización.

Strings (cadenas)

- Estudiaremos conjuntos de strings

Def.

Sea Σ un alfabeto (conjunto finito no vacío).

Definimos los *strings de largo k sobre Σ* como el conjunto Σ^k de manera recursiva

- $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \Sigma$.
- $\Sigma^k = \Sigma \times \Sigma^{k-1}$, para todo $k > 1$.

Los *strings sobre Σ* son

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{k \geq 0} \Sigma^k.$$

- **Ej:** $\Sigma = \{0, 1\}$.
 - ¿ Σ^3 ? ¿ Σ^* ?
- **Notación:** Omitiremos la notación de tupla: $(0, 1, 0) \rightarrow 010$.
- **Notación:** $|w|$ representa el largo de $w \in \Sigma^*$.

Concatenación y lenguaje

Def.

Sea Σ un alfabeto y $x, y \in \Sigma^*$ dos strings.

Si $x = x_1 \dots x_n$ e $y = y_1 \dots y_m$, entonces su *concatenación* es

$$x \cdot y = xy = x_1 \dots x_n y_1 \dots y_m$$

La *k-ésima potencia* de x es el string x^k dado por $\underbrace{xx \dots x}_{k \text{ veces}}$.

- **Ej:** $x = \text{hello}$, $y = \text{world}$.
 - ¿ xy , yx , x^4 ?

Def.

Un *lenguaje (sobre Σ)* es un subconjunto de Σ^* .

- **Ej:** $\Sigma = \{a, b, c, \dots, z\}$
 - ¿Lenguajes sobre Σ ?

Operaciones de lenguajes

Def.

Sea Σ un alfabeto y $L_1, L_2 \subseteq \Sigma^*$ dos lenguajes.

Definimos las siguientes operaciones entre lenguajes

- *Unión:* $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ o } w \in L_2\}$.
- *Intersección:* $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ y } w \in L_2\}$.
- *Diferencia:* $L_1 \setminus L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ y } w \notin L_2\}$.
- *Complemento:* $L_1^c = \Sigma^* \setminus L_1 = \{w \in \Sigma^* \mid w \notin L_1\}$.

- **Ej:** $\Sigma = \{a, b, c, \dots, z\}$, $L_1 = \{w \in \Sigma^* \mid w \text{ es palabra válida en español}\}$, $L_2 = \{\text{casa, perro, asdf}\}$.
 - ¿ $L_1 \cup L_2$? ¿ $L_1 \cap L_2$? ¿ $L_2 \setminus L_1$? ¿ L_1^c ?

Operaciones de lenguajes

Def.

Sea Σ un alfabeto y $L_1, L_2 \subseteq \Sigma^*$ dos lenguajes.

Definimos las siguientes operaciones entre lenguajes

- *Concatenación*: $L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ y } y \in L_2\}$.
- *Potencia*: $L_1^0 = \{\varepsilon\}$ y $L_1^k = L_1^{k-1} \cdot L_1$, para todo $k \geq 1$.
- *Estrella de Kleene*: $L_1^* = \bigcup_{k \geq 0} L_1^k$.

- **Ej:** $\Sigma = \{a, b, c, \dots, z, 0, 1\}$,
 $L_1 = \{w \in \Sigma^* \mid w \text{ es palabra válida en español}\}$, $L_2 = \{0, 1\}$.
 - ¿ $L_1 \cdot L_2$? ¿ L_2^2 ? ¿ L_2^* ? ¿ $(L_1 \cdot L_2)^*$?

Dudas/Preguntas

Problema del día

2. Sea $G = (V, E)$ un grafo simple no dirigido. Un *matching* de G es un conjunto $M \subseteq E$ de aristas que no comparten extremos, y

$$\nu(G) = \max\{|M| : M \text{ es un matching de } G\}$$

denota la cardinalidad de un matching máximo.

- a) ¿Que problemas modela el problema de matching máximo? ¿Qué aplicaciones podría tener?
- b) Trate de describir el problema de decisión asociado al problema de optimización de matching máximo.
- c) Suponga que tiene un algoritmo de decisión para MATCHING. ¿Cómo puede usarlo para calcular $\nu(G)$? ¿Cuántas llamadas al algoritmo de decisión realiza su procedimiento?
- d) Una vez conocido $r = \nu(G)$, queremos recuperar un matching de cardinalidad r . ¿Cómo puede usar el algoritmo de decisión para reconstruir un matching máximo? ¿Cuántas llamadas al algoritmo de decisión realiza su procedimiento?
- e) Discuta la relación entre los problemas de decisión, optimización y búsqueda.

Wrap-Up

- **Hoy:**

- Cosas administrativas
- Introducción
- Lenguajes formales

- **Prox. clase:**

- Automatas finitos