Modelos de computación

Clase 31

IIC 1253

Prof. Pedro Bahamondes

Outline

¿Qué es un modelo de computación?

Un modelo de computación es una representación abstracta de cómo una computadora realiza tareas o cálculos, es decir, **formalizan el concepto de algoritmos**.

¿Por qué son importantes los modelos de computación?

- **Entender las limitaciones de la computación:** Nos permiten comprender qué problemas pueden ser resueltos por una computadora y cuáles no
- **Diseñar algoritmos:** Sirven como base para el diseño de algoritmos correctos y eficientes
- **Estudiar la complejidad computacional:** Nos permiten clasificar problemas según su dificultad de resolución
- **Comparar la eficiencia de distintas soluciones:** Permiten evaluar las diferentes arquitecturas o soluciones

Intentos de formalización: S. XX

Funciones parcialmente recursivas
por K. Gödel, J. Herbrand, S. Kleene.

λ-calculus
por Alonzo Church.

Sistemas de Post
por Emil Post.

Máquinas de Turing
por Alan Turing.

. . .

Spoiler: Todos estos métodos sirven y son equivalentes

Alfabetos, palabras, lenguajes...

Antes de entrar en detalles sobre los modelos de computación, nos interesa primero definir formalmente qué es un problema.

Definición

Un **alfabeto** Σ es un conjunto finito de símbolos. Cada símbolo en Σ es una **letra**.

Ejemplo

$$\Sigma_1 = \{A, G, T, C, U\}$$

$$\Sigma_2 = \{0, 1\}$$

Alfabetos, palabras, lenguajes...

Definición

Una **palabra** es una secuencia finita de símbolos de Σ.

Ejemplo

$$w_1 = AATGCGATCTAGCGATCGG$$

$$w_2 = 1010010010$$

Al conjunto de todas las palabras con símbolos en Σ se le denomina Σ^*

Alfabetos, palabras, lenguajes...

Definición

Un **lenguaje** es un conjunto de palabras (sobre un alfabeto dado).

Ejemplo

$$L = \{0^n 1^n \mid n \in \mathbb{N}\}$$

Al conjunto de todas las palabras con símbolos en Σ se le denomina Σ^*

Problemas de decisión

Definición

El problema de decisión asociado a un lenguaje L consiste en, dado $w \in \Sigma^*$, decidir si $w \in L$.

Ejemplo

Podemos ver el problema de satisfactibilidad como un problema de decisión. Suponga que

$$P = \{p, q\}$$

$$\Sigma = \{p, q, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$$

Nótese que solo algunas palabras de Σ^* representan fórmulas, mientras que otras tales como $\neg\neg$ y $p\neg q \wedge\wedge$ q no representan fórmulas. Definimos

$$SAT = \{w \in \Sigma^* \mid w \text{ representa una fórmula y } w \text{ es satisfactible}\}$$

Decimos que un lenguaje L puede ser solucionado eficientemente si existe un algoritmo eficiente que decide L. En caso contrario, diremos que L es difícil.

Problemas de decisión

Definición

El problema de decisión asociado a un lenguaje L consiste en, dado $w \in \Sigma^*$, decidir si $w \in L$.

Observación

La cardinalidad de Σ^* es enumerable. Por lo tanto, los problemas de decisión tienen una cardinalidad no enumerable. Como la cantidad de programas posibles es enumerable, ¡existen problemas indecidibles!

Objetivos de la clase

- Introducir la noción de modelos de computación
- Entender cómo se ve una máquina de Turing determinista y no determinista
- Conocer superficialmente otros modelos de computación

Outline

Máquinas de Turing deterministas

Definición

Una **máquina de Turing determinista** M es una 6-tupla $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ donde:

- Q es un conjunto finito de estados
- Σ es un alfabeto tal que $\mathrm{B} \notin \Sigma$
- Γ es un alfabeto tal que $\Sigma \cup \{\mathrm{B}\} \subseteq \Sigma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\}$ es la función (parcial) de transición
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

Ejecución de una máquina de Turing determinista

¿Cómo funciona una Máquina de Turing?

- Se inicia con una cinta infinita, que contiene solamente los caracteres B y un input en ella, que utiliza solo caracteres en Σ.
- Hay un cabezal en la primera posición del input, que determina el símbolo actual. El estado inicial es q_0.
- En cada iteración de la ejecución, nos fijamos en el símbolo actual $a \in \Gamma$ y el estado actual $q \in Q$ y aplicamos la función delta de manera que si $\delta(q, a) = (q', a', f)$ entonces el nuevo estado será q', se reemplaza el símbolo a por a' y nos movemos el cabezal en la dirección de f en la cinta.
- Se repite el proceso hasta llegar a un estado final (en cuyo caso decimos que la máquina acepta la palabra) o no encontrar una transición (en cuyo caso decimos que rechaza la palabra).

Ejemplo

Construya una máquina de Turing M que, dado un natural, determine si es un número par.

Máquinas de Turing deterministas: Aceptación

Definición

Dada una máquina de Turing determinista M, definimos $L(M)$, el **lenguaje de aceptación de M** como el conjunto de palabras que son aceptadas por M, es decir:

$$L(M) = \{w \in \Sigma^* \mid M \text{ acepta } w\}$$

Ejemplo

Muestre que las máquinas de Turing posibles son enumerables. Concluya que hay lenguajes que no son aceptados por ninguna máquina de Turing.

Ejemplo

¿Cuál es el lenguaje aceptado por la máquina de Turing M que construimos?

Máquinas de Turing deterministas: Complejidad

Observación

Notemos que una máquina de Turing **puede no detenerse** en algunos inputs.

¿Cómo se mide el tiempo de ejecución de un algoritmo?

Para una MT con alfabeto Σ:

- Paso de M: La ejecución de una única instrucción usando la función de transición.
- $tiempo_M(w)$: Número de pasos ejecutados por M con entrada $w \in \Sigma^*$

Máquinas de Turing deterministas: Complejidad

Definición

Dada una máquina de Turing determinista M con alfabeto Σ, definimos su **tiempo de ejecución** como:

$$t_M(n) = \max\{tiempo_M(w) \mid w \in \Sigma^* \wedge |w| = n)\}$$

Utilidad del modelo de MTD

¿Por qué creemos que las máquinas de Turing son una buena formalización del concepto de algoritmo?

- Es un modelo sencillo sobre el que es fácil razonar y demostrar
- Toda máquina de Turing puede ser implementada
- Los algoritmos conocidos pueden ser implementados en máquinas de Turing
- Los mejores intentos de formalización de los algoritmos resultaron ser equivalentes a las máquinas de Turing
- **Tesis de Church**: Algoritmo = Máquina de Turing

Outline

Extensiones de las MT: No determinismo

Definición

Una **máquina de Turing no determinista** M es una 6-tupla $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$ donde:

- Q es un conjunto finito de estados
- Σ es un alfabeto tal que $\mathrm{B} \notin \Sigma$
- Γ es un alfabeto tal que $\Sigma \cup \{\mathrm{B}\} \subseteq \Sigma$
- $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\}$ es la relación de transición
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

Ejecución de una máquina de Turing no determinista

¿Cómo funciona una Máquina de Turing no determinista?

- Se inicia con una cinta infinita, que contiene solamente los caracteres B y un input en ella, que utiliza solo caracteres en Σ.
- Hay un cabezal en la primera posición del input, que determina el símbolo actual. El estado inicial es q_0.
- En cada iteración de la ejecución, nos fijamos en el símbolo actual $a \in \Gamma$ y el estado actual $q \in Q$ y nos fijamos en la relación de transición. **Si** $(q, a, q', a', f) \in \Delta$ **entonces podemos pasar al nuevo estado** q' **reemplazando el símbolo** a **por** a' **y moviendo el cabezal en la dirección de** f **en la cinta.**
- Se repite el proceso hasta llegar a un estado final o no encontrar una transición.

Dada una MTND M y una palabra w, decimos que M acepta w si existe **alguna** ejecución que termina en estado final, y que la rechaza en caso contrario.

Máquinas de Turing no deterministas: Aceptación

Definición

Dada una máquina de Turing no determinista M, definimos $L(M)$, el **lenguaje de aceptación de M** como el conjunto de palabras que son aceptadas por M, es decir:

$$L(M) = \{w \in \Sigma^* \mid M \text{ acepta } w\}$$

Ejemplo

Muestre que las máquinas de Turing posibles también son enumerables. Concluya que hay lenguajes que no son aceptados por ninguna máquina de Turing no determinista.

Determinismo vs. No determinismo

¿Es posible aceptar más lenguajes con las MT no deterministas?

Teorema

Si un lenguaje L es aceptado por una MT no determinista M_1, entonces L es aceptado por alguna MT determinista M_2.

Ejercicio

Demuestre el teorema. ¿Cuál es la diferencia de complejidad entre M1 y M2?

Máquinas de Turing no deterministas: Complejidad

¿Cómo se mide el tiempo de ejecución de un algoritmo?

Para una MTND con alfabeto Σ:

- Paso de M: La ejecución de una única instrucción usando la relación de transición.
- $tiempo_M(w)$: Número de pasos ejecutados por M con entrada $w \in \Sigma^*$ **en la ejecución más corta que acepta a** w

Máquinas de Turing no deterministas: Complejidad

Definición

Dada una máquina de Turing no determinista M con alfabeto Σ, definimos su **tiempo de ejecución** como:

$$t_M(n) = \max\{n, \max\{tiempo_M(w) \mid w \in \Sigma^* \wedge |w| = n)\}\}$$

El problema P vs. NP

Recordemos una famosa pregunta abierta en computación: ¿P = NP?

Dicho de otro modo, ¿es cierto que para todo lenguaje L tal que existe una máquina de Turing no determinista M tal que $L(M) = L$ y que funciona en tiempo polinomial existe una máquina de Turing determinista M' tal que $L(M') = L$ y M' también funciona en tiempo polinomial?

¡Este es un problema abierto con importantes consecuencias en computación!

Extensiones de las MT: Varias cintas

Definición

Una **máquina de Turing con k cintas** M es una 7-tupla $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$ donde:

- Q es un conjunto finito de estados
- Σ es un alfabeto tal que $\mathrm{B} \notin \Sigma$
- Γ es un alfabeto tal que $\Sigma \cup \{\mathrm{B}\} \subseteq \Sigma$
- $\delta \subseteq Q \times \Gamma^k \to Q \times \Gamma^k \times \{\leftarrow, \square, \rightarrow\}^k$ es la función (parcial) de transición
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

Ejecución de una máquina de Turing determinista

¿Cómo funciona una Máquina de Turing?

- Se inicia con k cintas infinitas, que contienen solamente los caracteres B y un input sobre la primera de ellas, que utiliza solo caracteres en Σ.
- Hay un cabezal en la primera posición del input y en una posición arbitraria de cada una de las otras cintas, que determina los símbolos actual. El estado inicial es q_0.
- En cada iteración de la ejecución, nos fijamos en el símbolo actual en cada cinta $(a_1, ..., a_k) \in \Gamma$ y el estado actual $q \in Q$ y aplicamos la función delta de manera que si $\delta(q, (a_1, ..., a_k)) = (q', (a'_1, ..., a'_k), (f_1, ..., f_k))$ entonces el nuevo estado será q', se reemplaza cada símbolo a_i por a'_i y movemos el cabezal de cada cinta i en la dirección de f_i.
- Se repite el proceso hasta llegar a un estado final (en cuyo caso decimos que la máquina acepta la palabra) o no encontrar una transición (en cuyo caso decimos que rechaza la palabra).

Utilidad de las MT con k cintas

Teorema

Si un lenguaje L es aceptado por una MT con k cintas M_1, entonces L es aceptado por alguna MT determinista M_2.

¡Podemos simplificarnos la vida usando k-cintas cuando sea más fácil expresar así los cálculos!

Outline

Sistemas de Post

Definición

Un **sistema de Post** es un modelo formal de computación basado en reglas de reescritura de cadenas de caracteres. Está compuesto por:

- Un alfabeto finito de símbolos Σ
- Un conjunto finito de reglas de producción de la forma $u \to v$ donde u y v son cadenas sobre el alfabeto
- Una cadena inicial (input)
- Una o más cadenas de parada

Sistemas de Post

La ejecución de un sistema de Post se ve como una secuencia de producciones desde el input hasta llegar a una cadena de parada usando las reglas de producción, o no encontrar una regla de producción para la cadena actual. Una palabra es aceptada si se termina en una cadena de parada.

Teorema

Los sistemas de Post son un modelo de computación Turing-completo

¡Es decir, son equivalentes a las Máquinas de Turing!

Cálculo Lambda

Definición

El cálculo lambda es un modelo matemático de computación basado en la transformación de expresiones utilizando funciones y sustituciones. Se compone de

- Variables (x, y, z, ...)
- Abstracciones de la forma $\lambda x.E$ donde x es una variable y E es una expresión, que representa una función de x
- Aplicaciones de la forma $(E_1\, E2)$, que representan el uso de dichas funciones sobre inputs

Cálculo Lambda

La ejecución en cálculo lambda se basa en la reducción beta, donde una expresión de la forma $((\lambda x.E)V)$ se simplifica reemplazando x con V en E.

Ejemplo

Resuelva

$$(\lambda x.x + 1)2$$

Teorema

El cálculo lambda es un modelo de computación Turing-completo

¡Es decir, también es equivalentes a las Máquinas de Turing!

Computación randomizada

Podemos extender los modelos de máquina de Turing para incluir aleatoriedad. Una forma de hacerlo es asumir un **generador aleatorio** que en una iteración i de una ejecución entregue un bit de forma aleatoria. La función de transición dependerá del estado actual, del símbolo actual pero además del bit aleatorio obtenido en la iteración respectiva.

Computación randomizada

¿Para qué sirven los modelos de computación randomizada?

- Nos permiten razonar sobre algoritmos aleatorios o pseudoaleatorios
- Nos permiten comparar la complejidad de estos algoritmos, con clases de complejidad relevantes que consideran la aleatoriedad como *RP* (*Randomized Polynomial time*: Problemas donde se acepta la palabra correcta en tiempo polinomial con probabilidad $\geq 1/2$, pero nunca se aceptan palabras incorrectas) o *BPP* (*Bounded-error Probabilistic Polynomial time*: Problemas donde tanto las palabras correctas como incorrectas tienen probabilidades de error menores a un límite, típicamente 1/3).

Computación cuántica

En computación, típicamente trabajamos con *bits* (0 o 1).

Ejercicio

Demuestre que todo alfabeto puede es equinumeroso con un alfabeto donde los símbolos son palabras en el alfabeto $\{0, 1\}$.

En computación cuántica, se trabaja con **qubits**, que son superposiciones de 0s y 1s. Para n qubits:

- El estado actual es una superposición de las 2^n posibles palabras de n bits.
- Cada palabra de n bits está acompañada de una probabilidad (entre 0 y 1) de tal manera que la suma de las probabilidades da 1.
- Un estado cuántico es equivalente a un vector 2^n-dimensional con componentes entre 0 y 1 cuya suma de componentes da 1.
- Las operaciones utilizan cálculo tensorial y están representadas por matrices unitarias, que transforman los estados de los qubits.
- El resultado se obtiene por medición del estado final, obteniendo una palabra de n bits según las probabilidades que expresa el estado cuántico.

Computación cuántica

¿Son los computadores cuánticos más poderosos que los computadores clásicos?

Teorema

Todo algoritmo cuántico puede ser representado como un algoritmo clásico.

Omitimos la demostración pues requiere mayores formalizaciones, pero la idea central es que podemos usar 2^n bits para representar n qubits y realizar operaciones clásicas de formas equivalentes a las cuánticas.

Computación cuántica

¿Son los computadores cuánticos más poderosos que los computadores clásicos?

Teorema

Todo algoritmo cuántico puede ser representado como un algoritmo clásico.

Sin embargo, existen problemas que se pueden resolver con algoritmos cuánticos que son "polinomiales" en computación cuántica pero para los que solamente se conocen algoritmos exponenciales en computación cuántica. Un ejemplo es el **algoritmo de Shor**, que permite factorizar un número en sus factores primos, lo que tiene consecuencias importantes en criptografía asimétrica.

Objetivos de la clase

- Introducir la noción de modelos de computación
- Entender cómo se ve una máquina de Turing determinista y no determinista
- Conocer superficialmente otros modelos de computación