**Teoria de la calculabilitat**

Introducció

La teoria de la computació està formada per les teories:

* D’autòmats i llenguatges.
* De la calculabilitat.
* De la complexitat.

Pregunta comuna: quines són les capacitats i limitacions dels ordinadors?

Els models de càlcul formals representen el concepte de computació:

* Autòmats finits (FA).
* Gramàtiques incontextuals (CFG).
* Autòmats de pila (PDA).
* Màquines de Turing (TM).

Tots els models precedents es poden codificar amb cadenes de símbols sobre {0, 1}.

Teoria de conjunts

Cantor defineix un conjunt com una col·lecció d’objectes, que pot ser finita o infinita.

Desenvolupa la teoria de conjunts naïve on qualsevol col·lecció és un conjunt amb les operacions habituals: unió, intersecció, diferència, producte escalar i vectorial.

Mateix cardinalitat:

Donats dos conjunts A i B, |A| = |B| si existeix una bijecció f : A -> B.

Enumerabilitat:

Un conjunt A és enumerable si A és finit o |A| = |N|.

Conjunts enumerables:

Els conjunts Z, N × N i Q son enumerables.

Conjunts no enumerables:

Els conjunts P(N) i R no són enumerables.

R no és enumerable:

Si R fos enumerable, (0, 1) també. Per tant, es pot escriure

(0, 1) = {r0, r1, r2, r3, ...}

Sigui ri = 0, xi0, xi1, xi2, xi3, ... l’expansió decimal de ri.

Definim yi = 7, si xii != 7 o yi = 3, si xii = 7

I formem el nombre r = 0, y0 y1 y2 ...

Llavors, r != ri perquè difereixen en l’i-èssim lloc decimal.

Contradicció: r pertany (0, 1), però no és a la llista.

*Teorema de Cantor;*

Donat un conjunt C,

|C| < |P(C)|

*Corol·lari*

|N| < |P(N)| < |P(P(N))| < ...

Es pot demostrar que |P(N)| = |R|.

*Hipòtesi del continu (CH)*

No hi ha cap conjunt C tal que |N| < |C| < |R|.

No es pot demostrar ni refutar.

Màquina de Turing

Model teòric d’ordinador, equivalent a un sistema formal.

*Diagrama

Descripción generada automáticamente*

Definició de llenguatge reconegut: Diem que una TM M accepta un mot x ∈ Σ\* si, a partir de la configuració inicial i x escrit a la cinta d’entrada, arriba a una configuració final.

Notació:

* L(M) = {x | M accepta x}
* M(x)↓: M s’atura amb entrada x.
* M(x)↑: M no s’atura amb entrada x.

Aturada segura: Diem que una TM M és d’aturada segura si M s’atura per a tota entrada x, és a dir, si ∀x ∈ Σ\* M(x)↓.

En el cas que M sigui una TM d’aturada segura, diem que L(M) és el llenguatge decidit per M.

Un llenguatge és decidible si és decidir per alguna TM.

Un llenguatge és semidecidible si és reconegut per alguna TM.

Tot llenguatge decidible és semidecidible.

Codificació de les TM:

* Les TM es poden codificar com a mots sobre {0, 1}.
* Hi ha una bijecció entre {0, 1}\* i N.
* El natural associat a una TM se’n diu nombre de Gödel de la TM.

La TM amb nombre de Gödel i es representa amb Mi.

Teorema: Existeixen llenguatges indecidibles.

Demostració:

* El conjunt de TM és enumerable.
* El conjunt de llenguatges sobre {0, 1} no és enumerable.

Per tant, hi ha llenguatges que no són decidits per cap TM.

Definició: Diem que una TM Mi computa una funció f si, per a tot mot x ∈ Σ\*:

* Mi(x)↓ i deixa a la cinta f(x) si f està definida en x.
* Mi(x)↑ si f està indefinida en x.

La funció computada per la màquina Mi es representa amb ϕi.

Diagrama, Diagrama de Venn

Descripción generada automáticamente

Tesi de Church-Turing: qualsevol algorisme concebible es pot implementar en el model de màquina de Turing.

En particular, qualsevol algorisme codificat en qualsevol llenguatge de programació es pot convertir en una TM.

En les demostracions no construirem màquines de Turing sinó pseudocodi, sabent que es pot arribar a codificar com a TM.

Proposició: existeix una TM universal. És a dir, hi ha un nombre de Gödel u tal que

L(Mu) = { <i, j> | Mi accepta j}.

L’existència de la màquina universal situa automàticament el seu llenguatge L(Mu) – el llenguatge universal – entre els semidecidibles.

És a dir, es pot “semidecidir” si una màquina de Turing donada accepta una certa entrada L(Mu) = { <i, j> | Mi accepta j}.

El problema de l’aturada s’assembla a L(Mu), però consisteix només a decidir l’aturada d’una TM amb una entrada. Dues versions:

* Problema de l’aturada més general:

HALT = {<i, j> | Mi(j)↓}.

* Problema de l’autoaturada que consisteix a decidir si una TM s’atura quan se li proporciona la seva pròpia codificació com a entrada:

K = {i | Mi(i)↓}.

Problemes indecidibles

Com saber si un programa s’atura?

* Executant-lo?
* Inspeccionant-ne el codi?
* Amb un altre programa?

Veurem que no existeix cap programa que pugui decidir l’aturada.

Ex.: conjectura dels primers bessons, existeixen infinits primers bessons, és a dir, primers que difereixen en un valor de 2 (com 3 i 5, 5 i 7, 11 i 13, 17 i 19, ...).

Teorema: *K i HALT són semidecidibles*

Considerem la TM següent:

M(<i, j>)

1. Simular Mi(j).
2. Acceptar.

Com que la línia 2 s’executa només si la simulació de la línia 1 s’ha aturat, la màquina M accepta exactament les entrades <i, j> de HALT. Per tant, L(M) = HALT i, per tant, HALT és semidecidible.

De manera semblant, es pot veure que K és semidecidible.

Teorema: *K és indecidible.*

Per reducció a l’absurd i diagonalització. Suposem que K és decidible: existeix una TM d’aturada segura M tal que L(M) = K. Llavors, podríem definit la TM:

N(x)

1. Simular M(x).
2. Si accepta
3. Bucle infinit

∃k N = Mk. Per a tot x,

Mk(x)↓ ⇔ M rebutja x ⇔ Mx(x)↑,

Amb x = k, Mk(k)↓⇔ Mk(k))↑, contradicció!

Teorema: *HALT és indecidible.*

Per reducció a l’absurd. Si HALT fos decidible, existiria una TM d’aturada segura M tal que L(M) = HALT. Llavors, podríem definit la TM:

N(i)

1. Simular M(<i, i>).
2. Si accepta
3. Acceptar
4. Si no
5. Rebutjar

Per a tot x,

N accepta i ⇔ M accepta <i, i> ⇔ <i, i> ∈ HALT ⇔ i ∈ K.

N és una TM d’aturada segura que reconeix K, contradicció!

Teorema: *el complementari d’un llenguatge decidible és decidible.*

Sigui A un llenguatge decidible. Llavors, existeix una TM d’aturada segura M tal que L(M) = A. Sigui M’ la TM:

M’(x)

1. Simular M(x).
2. Si accepta
3. Rebutjar
4. Si no
5. Acceptar

M’ és una TM d’aturada segura tal que

L(M’) = {x | M(x) rebutja} = {x | x no pertany a A} = A complementari.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Com que K i HALT són semidecidibles, pel teorema del complementari, els problemes K i HALT complementari no són semidecidibles.

Llenguatges decidibles:

REC = {L | L és un llenguatge decidible}, la classe que conté tots els llenguatges decidibles, com ara el conjunt dels primers, el buit, el total, el llenguatge dels mots sobre {a, b} que contenen el mateix nombre d’as que de bes, un conjunt que representa els llistats de notes de la FIB en tota la seva història (és un conjunt finit) o qualsevol conjunt finit.

Llenguatges semidecidibles:

RE = {L | L és un llenguatge semidecidible}, la classe que conté tots els llenguatges semidecidibles, com ara K, HALT, el llenguatge universal, el problema de la correspondència de Post, qualsevol llenguatge decidible o el conjunt d’enunciats d’afirmacions matemàtiques demostrables.

Diagrama

Descripción generada automáticamente

Reduccions

Donats dos llenguatges A i B sobre un alfabet Σ, diem que A es redueix a B si existeix una funció computable f tal que, per a tot x ∈ Σ\*,

x ∈ A ⇔ f(x) ∈ B.

En aquest cas, escrivim A <=m B (via f) i diem que f es una reducció de A a B.

Texto, Carta

Descripción generada automáticamente

Si A <=m B i B és decidible, A també és decidible.

Si A <=m B i A és indecidible, B també és indecidible.

Si A <=m B i B és semidecidible, A també és semidecidible.

Si A <=m B i A no és semidecidible, B tampoc no és semidecidible.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente