Altres aproximacions a la computabilitat

- La màquina de Turing no és l'únic model de computació que es pot considerar per tal d'arribar a les mateixes conclusions.
- Tampoc no va ser històricament el primer. Aproximacions similars es van dur a terme des del punt de vista de la lògica.
- Particularment interessant per a la ciència de la computació és la teoria de les funcions recursives.
- Considerarem còmputs relacionats amb **enters**. Concretament,

  $$f : \mathbb{IN} \longrightarrow \mathbb{IN}$$

  És a dir, funcions d'enters en enters. La qual cosa és fàcilment extensible a funcions del tipus $f : \mathbb{IN}^m \longrightarrow \mathbb{IN}^n$

Càlcul lambda

El càlcul lambda és un formalisme lògic-matemàtic que serveix per representar còmputs i que es pot veure com un sistema de reescritura (com les gramàtiques).

- Només existeixen funcions que es representen com a **abstraccions lambda**.
- Una abstracció lambda té la forma: $\lambda x.$(expressió en funció de x). On l'expressió ha d'estar formada també per abstraccions lambda.

  Exemple: $\lambda x.x$, $\lambda x.(x+2)$, $\lambda y.\lambda x.(x+y)$
- Les abstraccions lambda es poden aplicar sobre un argument i donen com a valor una altra abstracció o un valor definitiu.

  $(\lambda x.x)3 = 3$, $(\lambda x.(x+2))f = f+2$, $\lambda y.\lambda x.(x+y)5 = \lambda x.(x+5)$

Càlcul lambda

Inicialment només es disposa de variables lliures $(x, y, z, ...)$ y el mecanisme d'abstracció. I a partir d'això es defineix tot: els enters, funcions aritmètiques, els valors vertader i fals, predicats, etc. i fins i tot, la recursió!

És a dir, qualsevol funció recursiva es pot expressar com una abstracció lambda.

La computabilitat es pot estudiar des d'aquest punt de vista i, històricament, el primer problema indedidible va ser la comprovació d'equivalència entre dues expressions lambda.

El càlcul lambda està intimament relacionat amb alguns llenguatges de programació funcionals com LISP i SCHEME i més llunyanament amb altres com ML, MIRANDA o HASKELL.

La teoria de funcions recursives

Com que el nostre objectiu és estudiar la computabilitat dels càlculs (recursius) sobre enters adoptarem un nivell d'abstracció diferent.

- considerarem donats els enters i les tuples d'enters ($\mathbb{N}^m$) i partirem d'un conjunt reduït de funcions **molt** senzilles **clarament** computables.
- definirem aleshores un conjunt (finit) de mecanismes de obtenció de funcions a partir de funcions per tal d'obtindre noves funcions cada vegada més complexes. Com que els mecanismes es podran caracteriztar metòdicament el resultat seran funcions computables.
- el resultat serà una jerarquia de famílies de funcions que cobrirà el conjunt de **totes** les funcions (computables) dels enters en els enters.
- al final demostrarem que el conjunt de funcions així obtingudes és equivalent al conjunt de funcions Turing-computables (una vegada aplicada la codificació-decodificació corresponent).

Les funcions inicials

funció zero: escriure un zero, $\zeta() = 0$

$$\zeta : \mathbb{N}^0 \longrightarrow \mathbb{N}$$

funció successor: següent valor segons l'enumeració natural, $\sigma(x) = x + 1$.

$$\sigma : \mathbb{N} \longrightarrow \mathbb{N}$$

funcions projecció: selecció dels elements d'una tupla, $\pi_i^m(x_1, \ldots, x_m) = x_i$.

$$\pi_i^m : \mathbb{N}^m \longrightarrow \mathbb{N}$$

$$\pi_0^m : \mathbb{N}^m \longrightarrow \mathbb{N}^0$$

$\pi_0^m(\mathbf{x}) = ()$.

Combinació i composició de funcions

combinació: juxtaposició dels resultats de dues funcions.

$$f : \mathbb{N}^k \longrightarrow \mathbb{N}^m$$
$$g : \mathbb{N}^k \longrightarrow \mathbb{N}^n$$

$$f \times g : \mathbb{N}^k \longrightarrow \mathbb{N}^{m+n}$$

$$(f \times g)(\mathbf{x}) = (f(\mathbf{x}), g(\mathbf{x}))$$

composició: aplicació d'una funció sobre el resultat d'una altra.

$$g : \mathbb{N}^k \longrightarrow \mathbb{N}^m$$
$$f : \mathbb{N}^m \longrightarrow \mathbb{N}^n$$

$$f \circ g : \mathbb{N}^k \longrightarrow \mathbb{N}^n$$

$$(f \circ g)(\mathbf{x}) = f(g(\mathbf{x}))$$

Curiositat. Algunes funcions en càlcul lambda

$\mathsf{id} = \lambda x.x$

$\mathsf{undef} = (\lambda x.xx)(\lambda x.xx)$

$0 = \zeta() = \lambda f.\lambda x.x$

$n = \lambda f.\lambda x.ff \ldots fx = \lambda f.\lambda x.f^n x$

$\sigma = \lambda f.\lambda x.f(nfx)$

$\pi_i^m = \lambda x_1. \ldots \lambda x_m.(x_i)$

Recursivitat primitiva

$$\begin{array}{l} g : \mathbb{N}^k \longrightarrow \mathbb{N}^m \\ h : \mathbb{N}^{n+k+1} \longrightarrow \mathbb{N}^m \end{array} \qquad \begin{cases} f(\mathbf{x}, 0) = g(\mathbf{x}) \\ f(\mathbf{x}, y+1) = h(\mathbf{x}, y, f(\mathbf{x}, y)) \end{cases}$$

Definició de la funció f mijantçant recursivitat primitiva a partir de g i h.

$\mathcal{F}_{RP}$: conjunt de les funcions **recursives primitives**, que són aquelles que es poden obtindre mijantçant un nombre finit de combinacions, composicions i recursivitats primitives a partir de funcions inicials.

Funcions recursives primitives. Exemples

Funcions constants: $\mathsf{e3}() = \sigma(\sigma(\sigma(\zeta()))) = (\zeta \circ \sigma^3)() = 3$

Funció identitat: $\mathsf{id}(x) = \pi_1^1(x) = x.$

Funció suma: $\begin{cases} \mathsf{suma}(x, 0) & = x \\ \mathsf{suma}(x, y+1) & = \sigma(\mathsf{suma}(x, y)) \end{cases}$

Funció multiplicació: $\begin{cases} \mathsf{mult}(x, 0) & = 0 \\ \mathsf{mult}(x, y+1) & = x + \mathsf{mult}(x, y) \end{cases}$

Funció predecessor: $\begin{cases} \mathsf{pred}(0) & = 0 \\ \mathsf{pred}(x+1) & = x \end{cases}$

Funció resta (positiva): $\begin{cases} x \dot{-} 0 & = x \\ x \dot{-} (y+1) & = \mathsf{pred}(x \dot{-} y) \end{cases}$

Predicats. Exemples

Els predicats són funcions que s'evaluen a 0 o 1:

$$(x > y) = x \dot{-} y$$

$$(x \neq y) = (x \dot{-} y) + (y \dot{-} x)$$

$$\mathsf{neg}(x) = 1 \dot{-} x$$

$$(x = y) = \mathsf{neg}(x \neq y)$$

$$(x \leq y) = \mathsf{neg}(x > y)$$

Funcions condicionals. Exemples

La funció sialeshoressino:

$$\mathsf{sialeshoressino}(x, y, z) = (x = 1) \cdot y + (x = 0) \cdot z$$

Funcions tabulars:

$$\begin{cases} g_1(x) & \underline{\texttt{si}} & p_1(x) \\ g_2(x) & \underline{\texttt{si}} & p_2(x) \wedge \mathsf{neg}(p_1(x)) \\ \quad \vdots & & \\ g_k(x) & \underline{\texttt{si}} & p_k(x) \wedge \ldots \\ g_0(x) & \underline{\texttt{sino}} & \end{cases}$$

exercici!

Funcions recursives primitives. Propietats

- Les funcions de $\mathcal{F}_{RP}$ són computables.
- Les funcions de $\mathcal{F}_{RP}$ són totals.
- Hi ha funcions totals que **no** són recursives primitives!

$$\begin{cases} A(0,y) & = & y+1 \\ A(x+1,0) & = & A(x,1) \\ A(x+1,y+1) & = & A(x,A(x+1,y)) \end{cases}$$

Es pot demostrar que aquesta funció creix més ràpid que qualsevol funció de $\mathcal{F}_{RP}$.

Funcions recursives primitives. Propietats

Són recursives primitives altres funcions recursives "rares" que creixen molt ràpid com la de Fibonacci?

$$\begin{cases} F(0) &= 0 \\ F(1) &= 1 \\ F(x) &= F(x-1) + F(x-2) \end{cases}$$

Depén. Aquesta és recursiva primitiva!

$$f(x) = \mathsf{sialeshoressino}(x = 0, (0, 1), (F(x), F(x+1)))$$

Aleshores ens hem de creure que hi ha funcions totals no recursives primitives?

Vale. Ahí va un altre exemple.

Com que les funcions de $\mathcal{F}_{RP}$ es poden descriure de forma finita (una determinada seqüència de combinacions, composicions i recursivitats primitives aplicades sobre determinades funcions inicials), aquestes admeten una codificació i, per tant, una **enumeració**.

$$\mathcal{F}_{RP} = \{f_i\}_{i=1}^{\infty}$$

Definim una nova funció clarament computable i total: $f(x) = f_x(x) + 1$. Però f no pot ser recursiva primitiva ja que si ho fóra li correspondria un determinat índex, y, en l'enumeració de $\mathcal{F}_{RP}$. Aleshores,

$$f(y) = f_y(y) + 1 = f(y) + 1 \qquad \text{impossible!}$$

Funcions μ-recursives o recursives parcials

S'introdueix un nou mecanisme de construcció de funcions: la **minimalització**.

Donada $g : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$, es defineix $f : \mathbb{N}^n \longrightarrow \mathbb{N}$ com

$$f(\mathbf{x}) = \min\{y \mid \ g(\mathbf{x}, y) = 0\}$$

$$\mathsf{indef}(\mathbf{x}) = \mu t[1 = 0]$$

$$\mathsf{id}(\mathbf{x}) = \mu y[x \dot{-} y = 0]$$

$$\mathsf{div}(x, y) = \mu d[(x + 1) \dot{-} y(d + 1) = 0]$$

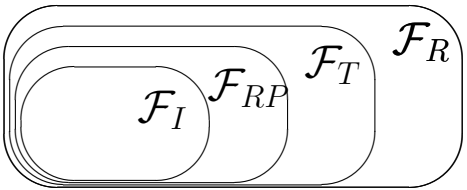
Funcions μ-recursives o recursives parcials

Una funció es diu **minimalitzable** si està definida per als valors menors que el primer valor per al quals s'anul·la

Les funcions definides mijantçant minimalització són computables si:

- g és minimalitzable.
- g és total.

Funcions μ-recursives o recursives parcials



El conjunt de les funcions μ-recursives està format per aquelles funcions que es poden definir mijantçant un nombre finit de combinacions, composicions, recursivitats primitives i minimalitzacions de funcions minimalitzables.

Si les minimalitzacions es restringeixen a funcions totals, el conjunt que s'obté és el mateix.

També es pot demostrar que tota funció μ-recursiva es pot expressar fent servir una única minimalització (necessàriament d'una funció total!).

Les funcions Turing-computables són μ-recursives

Considerem una màquina M que computa una funció parcial de $\mathbb{IN}$ en $\mathbb{IN}$ definida en un determinat subconjunt $X \subseteq \mathbb{IN}$.

$$M = \langle Q, \{1\}, \{1, 0\}, \delta, q_1, 0, \{q_0\} \rangle$$

Suposarem cinta semiinfinita, un únic estat de parada, q_0, i la codificació binaria invertida dels enters on el zero i el blanc són el mateix símbol.

Es tractarà ara de construir una funció recursiva que compute $f(x)$ per a tot x de X.

Les funcions Turing-computables són μ-recursives

- funció moviment següent:

$$\mathsf{mov}(i,x) = \begin{cases} 2 \ \underline{\texttt{si}} & \exists q_j \in Q, y \in \{1,0\} : \delta(q_i,x) = (q_j,y,\rightarrow) \\ 1 \ \underline{\texttt{si}} & \exists q_j \in Q, y \in \{1,0\} : \delta(q_i,x) = (q_j,y,\leftarrow) \\ 0 \ \underline{\texttt{sino}} & \end{cases}$$

- funció símbol següent:

$$\mathsf{sim}(i,x) = \begin{cases} y \ \underline{\texttt{si}} & \exists q_j \in Q, D \in \{\leftarrow,\rightarrow\} : \delta(q_i,x) = (q_j,y,D) \\ x \ \underline{\texttt{sino}} & \end{cases}$$

- funció estat següent:

$$\mathsf{est}(i,x) = \begin{cases} j & \underline{\texttt{si}} \quad \exists D \in \{\leftarrow,\rightarrow\}, y \in \{1,0\} : \delta(q_i,x) = (q_j,y,D) \\ |Q| & \underline{\texttt{sino}} \end{cases}$$

Les funcions Turing-computables són μ-recursives

Les configuracions de M es correspondran amb ternes (w, i, n) on w és el contingut de la cinta, i és l'estat actual i n la posició del capçal (si el capçal se n'eix, $n = 0$). Aleshores, si $\boxed{\mathsf{actsim}(w, i, n) = \mathsf{div}(w, 2^{n-1}) \dot{-} \mathsf{mult}(2, \mathsf{div}(w, 2^n))}$ posició següent:

$$\mathsf{posseg}(w, i, n) = n \dot{-} \mathsf{igual}(\mathsf{mov}(i, \mathsf{actsim}(w, i, n)), 1) + \mathsf{igual}(\mathsf{mov}(i, \mathsf{actsim}(w, i, n)), 2)$$

estat següent:

$$\mathsf{estseg}(w, i, n) = \mathsf{est}(i, \mathsf{actsim}(w, i, n)) + \mathsf{mult}(|Q|, \mathsf{no}(\mathsf{posseg}(w, i, n)))$$

contingut de cinta següent:

$$\mathsf{cintaseg}(w, i, n) = w \dot{-} \mathsf{mult}(2^n, \mathsf{actsim}(w, i, n)) + \mathsf{mult}(2^n, \mathsf{sim}(i, \mathsf{actsim}(w, i, n)))$$

Les funcions Turing-computables són μ-recursives

Ara es pot definir:

$$\mathsf{pas} : \mathbb{IN}^3 \longrightarrow \mathbb{IN}^3$$

$$\mathsf{pas} = \mathsf{cintaseg} \times \mathsf{estseg} \times \mathsf{posseg}$$

Per definició, aquesta funció compleix que

$$\mathsf{pas}(w, i, n) = (w', j, n') \iff (w, q_i, n) \vdash_M (w', q_j, n')$$

Aleshores,

$$\begin{cases} \mathsf{comp}(w, i, n, 0) & = (w, i, n) \\ \mathsf{comp}(w, i, n, t+1) & = \mathsf{pas}(\mathsf{comp}(w, i, n, t)) \end{cases}$$

$\mathsf{comp}(w, 1, 1, t)$ calcula la configuració de M amb w després de t moviments.

Les funcions Turing-computables són μ-recursives

Com que $\pi_2^3(\mathsf{comp}(w,1,1,t))$ ens dóna l'estat després de t moviments, i q_0 és l'únic estat de parada, podem definir aleshores, per minimalització,

$$\mathsf{parada}(w) = \mu t[\pi_2^3(\mathsf{comp}(w,1,1,t)) = 0]$$

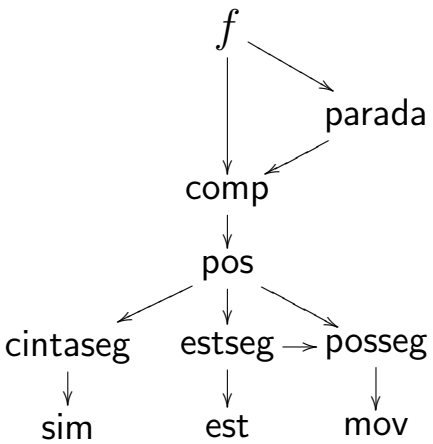
Amb la qual cosa, la funció que calcula M es pot escriure com a

$$f(w) = \pi_1^3(\mathsf{comp}(w,1,1,\mathsf{parada}(w)))$$

Amb la qual cosa queda demostrat que f és μ-recursiva!

Les funcions Turing-computables són μ-recursives

Sumari de les principals funcions definides i les seues dependències:



Només faltaria actsim que és una operació aritmètica sobre w utilitzada per cintaseg, estseg i posseg.