UNIVERSITÀ DEGLI STUDI "ROMA TRE"
IN410 - CALCOLABILITÀ E COMPLESSITÀ
A.A. 2023-2024
PROF. M. PEDICINI

12/01/2024 PROVA IN ITINERE – DURATA 3H00

COGNOME _____ NOME _____ MATRICOLA _____

Esercizio 1. *(1) Spiegare perché una funzione* strettamente sublineare *(ovvero tale che $f(n) < n$ per ogni n) non può essere costruibile in tempo;*

Soluzione. un f-timer quando $f(n) < n$ non può esistere poiché deve terminare la computazione prima di leggere l'input, quindi esiste solo quando il valore di $f(n) \geq n$ oppure è indipendente da n. □

(2) Costruire esplicitamente un f-timer per la funzione

$$f(n) := 2n - 2;$$

Soluzione. È sufficiente costruire una macchina di Turing a due nastri che copia l'input sul secondo nastro a meno dei primi due caratteri e poi quando viene raggiunta la fine dell'input sul primo nastro ritorna all'inizio della parola sul secondo nastro.

Vedi in Figura 1, la specifica della macchina da utilizzare con i comandi della libreria sulle Macchine di Turing per Wolfram Mathematica fornita a lezione.

□

(3) Dimostrare che ogni funzione $g_{(a,b)}(n) := an - b$ è costruibile in tempo per ogni coppia di interi $(a, b) \in \mathbb{N}^2$ tale che $a \geq b$.

Soluzione. Per ipotesi $a \geq b$ se consideriamo $c = a - b$ possiamo scrivere $a = c + b$ e quindi $an - b = cn + bn - b$. Se b è pari, avremo che $d = b/2$ e quindi

$$an - b = cn + d(2n - 2)$$

Quindi possiamo ottenere un $(an - b)$-timer iterando per c volte la lettura di n e per d volte l'esecuzione del $(2n - 2)$-timer definito al punto precedente.

Se b non è pari, allora consideriamo $b + 1$ che lo è e poniamo $d = (b + 1)/2$ avremo che

$$d(2n - 2) = \frac{b+1}{2}(2n - 2) = (b + 1)n - b - 1 = bn - b + n - 1$$

e quindi

$$an - b = (c - 1)n + d(2n - 2) + 1.$$

Essendo a e b fissati possiamo adattare caso per caso il timer. Si noti che in funzione di a e b è possibile che per alcuni valori di n si potrebbe finire in un caso sub-lineare (ad esempio se $n = 1$ nel caso del punto 2, in tal caso ci si può fermare appena finito di leggere l'input), in sostanza la funzione che è costruibile in tempo è la funzione definita per casi:

$$g'_{(a,b)}(n) := \begin{cases} an - b & an - b >= n \\ n & an - b < n \end{cases}$$

```
timer = {
  (*copio l'input sul secondo nastro-
  tranne i primi quattro caratteri*)

  {q0, {1, blank}} -> {q1, {1, blank}, {1, 0}},
  {q0, {0, blank}} -> {q1, {0, blank}, {1, 0}},
  {q1, {1, blank}} -> {q2, {1, blank}, {1, 0}},
  {q1, {0, blank}} -> {q2, {0, blank}, {1, 0}},
  {q2, {1, blank}} -> {q2b, {1, blank}, {1, 0}},
  {q2, {0, blank}} -> {q2b, {0, blank}, {1, 0}},
  {q2b, {1, blank}} -> {q2c, {1, blank}, {1, 0}},
  {q2b, {0, blank}} -> {q2c, {0, blank}, {1, 0}},
  {q2c, {1, blank}} -> {q3, {1, 1}, {1, 1}},
  {q2c, {0, blank}} -> {q3, {0, 0}, {1, 1}},
  {q3, {1, blank}} -> {q3, {1, 1}, {1, 1}},
  {q3, {0, blank}} -> {q3, {0, 0}, {1, 1}},

  (* al blank blank torno indietro sul secondo nastro *)

  {q3, {blank, blank}} -> {q4, {blank, blank}, {0, -1}},
  {q4, {blank, 1}} -> {q4, {blank, 1}, {0, -1}},
  {q4, {blank, 0}} -> {q4, {blank, 0}, {0, -1}},
  {q4, {blank, blank}} -> {final, {blank, blank}, {0, 0}}
```

Comandi per eseguire la computazione

```
final = qF1;
TuringMachineGraph[timer]
blank = -1;
c0 = MultiInitialConfiguration[{0, 0, 0}, q0, blank, 2]
ManipulateMultiComputation[timer3, c0]
```

FIGURA 1. $(2n-2)$-timer: da ogni input binario scaturisce una computazione di lunghezza $2n-2$.

□

Esercizio 2. *(1) Sia $f(n)$ la successione di Fibonacci (ovvero definita come $f(0) := 0$, $f(1) := 1$, $f(n+1) := f(n) + f(n-1)$). Mostrare che l'insieme*

$$X := \{x | x = f(n) \cdot f(n+1) \text{ per qualche } n\}$$

è un insieme ricorsivo.

Soluzione. Per dimostrare che X è ricorsivo dalla definizione devo mostrare che la sua funzione indicatrice lo è. Mostriamo inizialmente una espressione ricorsiva per la funzione di Fibonacci: è facile da mostrare in quanto possiamo utilizzare la ricorsione generalizzata al caso con vettori binari. In tal caso avremo la funzione ausiliaria che restitiusce l'n-esima coppia di numeri di fibonacci consecutivi,

$$\texttt{auxfib} := \texttt{REC}^2(\texttt{fibBase}, \texttt{fibStep}) \qquad [[\texttt{auxfib}]] :: \mathbb{N} \to \mathbb{N}^2$$

dove

$$\texttt{fibBase} := \texttt{CONCAT}(\texttt{C}_{0,1}, \texttt{C}_{0,0}) \qquad [[\texttt{fibBase}]] : \mathbb{N}^0 \to \mathbb{N}^2$$

e

$$\texttt{fibStep} = \texttt{CONCAT}(\texttt{COMP}(\texttt{CONCAT}(\texttt{P}_{3,2}, \texttt{P}_{3,3}), \texttt{S}), \texttt{P}_{3,2}) \qquad [[\texttt{fibStep}]] : \mathbb{N}^3 \to \mathbb{N}^2$$

da cui ricaviamo l'n-esimo Fibonacci per proiezione della seconda componente:

$$\texttt{fib} := \texttt{COMP}(\texttt{auxfib}, \texttt{P}_{2,2}) \qquad \text{per cui } f = [[\texttt{fib}]].$$

Torniamo alla funzione caratteristica dell'insieme X. A questo scopo è sufficiente trovare n per cui $f(n)f(n+1) = x$, posso ricavare per minimalizzazione il più piccolo valore k per cui la differenza troncata $\Delta(x, f(k)f(k+1))$ si annulla (ovvero quando $x - f(k)f(k+1) \leq 0$), e infine rispondere al quesito definendo

$$\mathbf{1}_X(x) := \mathbf{1}_{\{0\}}(f(k)f(k+1) - x).$$

Dunque definisco la funzione da minimalizzare:

$$s(x, k) := \Delta(x, f(k)f(k+1))$$

per cui l'espressione ricorsiva per la funzione s è

$$\texttt{T} := \texttt{COMP}(\texttt{CONCAT}(\texttt{P}_{2,1}, \texttt{G}), \texttt{diff}) \qquad [[\texttt{T}]] = s : \mathbb{N}^2 \to \mathbb{N}$$

prendendo con `G` l'espressione ricorsiva del prodotto di due numeri di Fibonacci successivi:

$$\texttt{G} := \texttt{COMP}(\texttt{CONCAT}(\texttt{COMP}(\texttt{P}_{2,2}, \texttt{fib}), \texttt{COMP}(\texttt{COMP}(\texttt{P}_{2,2}, \texttt{succ}), \texttt{fib})), \texttt{mult})$$

Finalmente $\texttt{kappa} := \texttt{MIN}(\texttt{T})$ con $[[\texttt{kappa}]] : \mathbb{N} \to \mathbb{N}$. In modo analogo alla definizione di `T` definiamo l'espressione `R` per rappresentare la funzione $f(k)f(k+1) - x$ che utilizzeremo per verificare che effettivamente per il k trovato per minimalizzazione si abbia $f(k)f(k+1) = x$.

Analogamente alla definizione di `T` sarà

$$\texttt{R} := \texttt{COMP}(\texttt{CONCAT}(\texttt{G}, \texttt{P}_{2,1}), \texttt{diff}) \qquad [[\texttt{R}]] : \mathbb{N}^2 \to \mathbb{N}$$

Quindi se $x \notin X$ avremo un valore diverso da zero □

(2) Fornire l'espressione ricorsiva di $\mathbf{1}_X$ e delle eventuali funzioni ausiliarie utilizzate.

Soluzione. Finalmente, possiamo definire l'espressione ricorsiva per la funzione caratteristica di X:

$$\texttt{isx} := \texttt{COMP}(\texttt{COMP}(\texttt{CONCAT}(\texttt{COMP}(\texttt{P}_{2,2}, \texttt{kappa}), \texttt{P}_{2,1}), \texttt{R}), \texttt{iszero}).$$

Infine ricordiamo le espressioni ricorsive per le funzioni ausiliarie utilizzate:

(a) $\texttt{succ} := \texttt{COMP}(\texttt{CONCAT}(\texttt{P}_{1,1}, \texttt{C}_{1,1}), \texttt{S})$
(b) $\texttt{mult} := \texttt{REC}(\texttt{C}_{1,0}, \texttt{COMP}(\texttt{CONCAT}(\texttt{P}_{3,1}, \texttt{P}_{3,3}), \texttt{S}))$
(c) $\texttt{iszero} := \texttt{REC}(\texttt{C}_{0,1}, \texttt{COMP}(\texttt{P}_{2,2}, \texttt{D}))$
(d) $\texttt{diff} := \texttt{REC}(\texttt{P}_{1,1}, \texttt{COMP}(\texttt{P}_{3,3}, \texttt{D}))$

□

Esercizio 3. *Sia* $h : \mathbb{N} \to \mathbb{N}$ *la seguente funzione*

$$h(x) := \begin{cases} f(x)^2 + 1 & \textit{se } x \textit{ è pari} \\ \lfloor x/2 \rfloor & \textit{se } x \textit{ è dispari.} \end{cases}$$

con $f(x)$ *l'*x*-esimo numero della successione di Fibonacci (come all'esercizio precedente).*

(1) Fornire un lambda termine H *che rappresenta la funzione* h*.*

Soluzione. Per ottenere il lambda termine che rappresenta la funzione di Fibonacci quando gli interi sono rappresentati con i numerali di Church, costruiamo l'iteratore della funzione ausiliaria che abbiamo definito nell'esercizio precedente (modellando la ricorsione generalizzata alla dimensione 2) quindi avremo che il lambda-termine per la funzione di Fibonacci sarà:

$$F := \lambda n\,(((n)\texttt{step})\texttt{base})\mathtt{P}_{2,2}$$

dove il lambda termine

$$\texttt{base} := \langle \underline{1}, \underline{0} \rangle$$

rappresenta la coppia dei due primi numeri di Fibonacci (e, in particolare, la seconda componente è $f(0)$).

Il lambda termine `step` dipenderà da un parametro corrispondente alla coppia calcolata all'iterazione precedente e dunque usando per i parametri la notazione a pattern avremo:

$$\texttt{step} := \lambda\langle a_1, a_2\rangle\langle((\texttt{add})a_1)a_2, a_1\rangle$$

e dunque esplicitando le operazioni sulle coppie:

$$\texttt{step} := \lambda c\lambda p((p)((\texttt{add})(c)P_{2,1})(c)P_{2,2})(c)P_{2,1}$$

dove i lambda termini proiezione sono definiti come

$$P_{n,j} := \lambda x_1 \ldots \lambda x_n\, x_j$$

inoltre $\texttt{add} := \lambda n\lambda m((m)\texttt{succ})n$ dove

$$\texttt{succ} := \lambda n\lambda f\lambda x(f)((n)f)x;$$

Torniamo ora alla funzione h poiché si tratta di una funzione definita per casi, possiamo utilizzare il termine `if_then_else` per selezionare i due casi a patto di avere un lambda-termine che si valuta a `true` sui pari:

$$H := \lambda n(((\texttt{if\_then\_else})(\texttt{iseven})n)((\texttt{add})\underline{1})(\lambda x((\texttt{mult})x)x)n)(\texttt{half})n$$

Infine ricordiamo la definizione dei termini che abbiamo utilizzato:

(a) $\texttt{if\_then\_else} := \lambda b\lambda x_1\lambda x_2((b)x_1)x_2$;

(b) $\texttt{half} := \lambda n((n)\texttt{step}_{\texttt{div2}})\texttt{base}_{\texttt{div2}})P_{2,2}$ dove

$$\texttt{base}_{\texttt{div2}} := \langle \texttt{true}, 0\rangle \qquad \texttt{step}_{\texttt{div2}} := \lambda\langle b, x\rangle\langle(\texttt{not})b, ((b)(\texttt{succ})x)x\rangle;$$

(c) $\texttt{mult} := \lambda n\lambda m((m)\texttt{step}_{\texttt{mult}})\texttt{base}_{\texttt{mult}}$ dove

$$\texttt{base}_{\texttt{mult}} := n \qquad \texttt{step}_{\texttt{mult}} := \lambda s((\texttt{add})n)s;$$

(d) $\texttt{iseven} := \lambda n((n)\texttt{not})\texttt{true}$;

(e) $\texttt{true} := \lambda x\lambda y x$ (e anche $\texttt{false} := \lambda x\lambda y y$);

(f) $\texttt{not} := \lambda b((b)\texttt{false})\texttt{true}$ (e anche $\texttt{and} := \lambda a\lambda b((a)b)\texttt{false}$ e inoltre $\texttt{or} := \lambda a\lambda b((a)\texttt{true})b$, e infine $\texttt{xor} := \lambda a\lambda b((a)(\texttt{not})b)b$);

□

(2) Definita g nel modo seguente:

$$g(x) := \begin{cases} k & \text{se } k \text{ è il primo valore per cui } h^k(x) = 2 \\ \bot & \text{se } h^k(x) \neq 2 \text{ per ogni } k \end{cases}$$

dove $h^k(x)$ è x se $k = 0$ ed è $h(h^{k-1}(x))$ se $k > 0$.

Fornire un lambda termine G che rappresenta la funzione g.
(suggerimento scrivere il lambda termine che rappresenta l'iterazione di h fino a quando il risultato non è 2, e risolvere l'equazione ricorsiva utilizzando il teorema di punto fisso).

Soluzione. Poiché non sappiamo quante volte dobbiamo iterare la funzione base sarà necessario utilizzare un'equazione di punto fisso controllata da una stop-condition:

$$D := \lambda d\lambda s(((\texttt{if\_then\_else})(\texttt{stop\_condition})s)(\texttt{output})s)((d)d)(\texttt{step})s$$

per cui il punto fisso si ottiene come

$$((D)D)\texttt{base}.$$

L'idea è che lo stato ad una certa iterazione sarà dato dalla coppia costituita dall'indice del numero di iterazioni i già effettuate e dal valore di $h^i(x)$, pertanto il termine

$$G := \lambda x((D)D)\texttt{base} \qquad \text{dove } \texttt{base} := \langle \underline{0}, x \rangle.$$

La funzione da iterare, deve prendere la coppia che rappresenta lo stato al passo precedente e ne incrementa la prima componente e applica H alla seconda, dunque si rappresenta con il lambda-termine:

$$\texttt{step} := \lambda\langle i, x\rangle\langle(\texttt{succ})i, (H)x\rangle$$

La funzione che fa il test di arresto controlla se la seconda componente della coppia vale 2, dunque si rappresenta con il lambda-termine:

$$\texttt{stop\_condition} := \lambda\langle i, x\rangle(((x)\lambda\langle d_1, d_2, d_3\rangle\langle\texttt{false}, d_1, d_2\rangle)\langle\texttt{true}, \texttt{false}, \texttt{false}\rangle)P_{3,3}$$

Infine la funzione di output deve solo proiettare la prima componente dello stato (il numero di iterazioni) una volta che la seconda componente soddisfa la condizione di arresto:

$$\texttt{output} := \lambda\langle i, x\rangle i = \lambda c(c)P_{2,1}$$

□