7 Calcolabilita'

Questa sezione integra il capitolo 3 del Papadimitriou con ulteriori risultati attinenti al modello di calcolo basato sulle Turing Machines (TM).

Il modello di calcolo delle Macchine di Turing (TM) puo' essere utilizzato sia per risolvere problemi di decisione (o accettazione), sia per calcolare funzioni (parziali) su Σ^*. Per far questo e' sufficiente aggiungere un nastro di output (per semplicita' si puo supporre che il nastro di input e' read-only, quello di output write-only, e che su tali nastri si possano trovare solo caratteri in un alfabeto Σ prefissato, mentre sui nastri di lavoro si possono usare caratteri in un alfabeto $\Gamma \supseteq \Sigma$ che varia da TM a TM). Un modo conciso di esprimere una serie di risultati fondamentali sul modello di calcolo delle TM e' di considerare la struttura applicativa parziale $X = (\Sigma^*, \cdot)$ [SI TRATTA DI UNA PARTIAL COMBINATORY ALGEBRA (pCA), molti dei risultati che seguono si generalizzano ad un'arbitraria pCA], dove la **operazione binaria e parziale $\cdot$ e' data in termini di una codifica delle TM (canoniche) in Σ^*: $y \cdot x = z$ sse y codifica una TM M e M con input x termina e produce z in output**. Scriveremo M sia per indicare una TM che la sua codifica come stringa.

Indichiamo con X l'insieme Σ^* delle stringhe. La nozione di m-aria funzione Turing-calcolabile puo' essere data in termini di $\cdot$.

Definition 7.1 *Una funzione parziale $f: X^m \rightharpoonup X$ e'* **calcolabile** $\overset{\Delta}{\iff}$ *esiste $e \in X$ t.c. $f(x_1, \ldots, x_m) \simeq e \cdot x_1 \cdot \ldots \cdot x_m$, dove $x_i \in X$.*
La funzione **universale** *e' quella funzione $U_m: X^{m+1} \rightharpoonup X$ tale che $U_m(M, x_1, \ldots, x_m) = y$ sse M con input $x_1; \ldots; x_m$ termina e produce y in output. (Si suppone di estendere Σ con un simbolo separatore ";", p.e. il carattere blank).*

Valgono i seguenti risultati fondamentali.

Theorem 7.2 (TM universale) *La funzione U_m e' calcolabile, e la TM che la calcola e' detta TM universale.*

Theorem 7.3 (S-m-n) *Data $f: X^{m+n} \rightharpoonup X$ parziale calcolabile, esiste $g: X^m \to X$ totale calcolabile t.c. $U_n(g(\overline{x}), \overline{y}) \simeq f(\overline{x}, \overline{y})$ per ogni $\overline{x} \in X^m$ e $\overline{y} \in X^n$.*

Le nozioni di insieme decidibile o semi-decidibile e la relazione di riducibilita' possono essere formulate in termini di funzioni calcolabili:

- $L \subseteq X$ e' **decidibile** sse $\exists e: X. \forall x: X. (x \in L \supset e\ x = T) \land (x \notin L \supset e\ x = F)$
  (dove T e F sono due elementi distinti di X che rappresentano rispettivamente vero e falso)
- $L \subseteq X$ e' **semi-decidibile** sse $\exists e: X. \forall x: X. (x \in L \leftrightarrow e\ x \downarrow)$
- L_1 e' **riducibile** a L_2 (notazione $L_1 \leq L_2$) sse esiste $f: X \to X$ totale e calcolabile t.c. $\forall x: X. (x \in L_1 \leftrightarrow f(x) \in L_2)$.

L'indecidibilita' dell'halting problem $H \overset{\Delta}{=} \{(x, y) | x\ y \downarrow\}$ segue dal seguente risultato (e dalla proprieta' di chiusura per riduzione dei sottoinsiemi decidibili).

Theorem 7.4 (indecidibilita' dell'halting problem) *L'insieme $D \overset{\Delta}{=} \{x \in X | x \cdot x \downarrow\}$ e'* indecidibile*, cioe' non esiste $d \in X$ t.c. $d \cdot x = T$ se $x \in D$ e $d \cdot x = F$ se $x \notin D$.*

Proof La dimostrazione procede per assurdo. Supponiamo che d esista, allora possiamo definire (mediante astrazione) $e \in X$ t.c. $e\ x = if(d \cdot x, \perp, I) \cdot x$, dove $\perp$ e I sono le TM che calcolano rispettivamente la funzione ovunque divergente e l'identita'. Vediamo che e non puo' appartenere ne' a D ne' al suo complementare:

- se $e \in D$ allora $e \cdot e \downarrow$, e quindi si deve avere $d \cdot e = F$, cioe' $e \notin D$ (assurdo);
- se $e \notin D$ allora $e \cdot e \uparrow$, e quindi si deve avere $d \cdot e = T$, cioe' $e \in D$ (assurdo).

∎

Proposition 7.5 *Se $L_1 \leq L_2$ e L_2 e' (semi-)decidibile, allora anche L_1 e' (semi-)decidibile.*

Il seguente teorema di Rice asserisce l'indecidibilita' delle proprieta' *estensionali* non banali dei programmi. Introduciamo un po' di terminolgia:

- $F(X) \subseteq X \rightharpoonup X$ e' l'insieme delle funzioni calcolabili in X

- fissato $A \subseteq F(X)$, il sottoinsieme $A' \subseteq X$ e' t.c. $x \in A'$ sse la funzione rappresentata da x e' in A.

Theorem 7.6 (Rice) *Se $A \subset F(X)$, allora A' e' decidibile sse A e' banale, cioe' $A = \emptyset$ o $A = F(X)$.*

Proof Si sfrutta la tecnica della riduzione, ovvero facciamo vedere che esiste una $g: X \to X$ totale e calcolabile t.c. $x \in D$ sse $g(x) \in A'$. Quindi se A' fosse decidibile lo sarebbe anche D.
Supponiamo che A non sia banale e che $\perp \notin A'$ (altrimenti si considera il complementare di A), inoltre sia $a \in A'$. Sia f la funzione parziale calcolabile t.c.

$$f(x,y) \simeq \text{if } (x \cdot x)\downarrow \text{ then } a \cdot y$$

e g una funzione totale calcolabile (data dal Teorema S-m-n) t.c. $U(g(x), y) \simeq f(x, y)$. Allora g e' una riduzione di D ad A'. Infatti, se $x \in D$ allora $g(x)$ rappresenta la stessa funzione rappresentata da a e quindi $g(x) \in A'$. Se invece $x \notin D$ allora $g(x)$ rappresenta la funzione ovunque divergente, e quindi $g(x) \notin A'$. ∎

Definition 7.7 *Fissata una codifica delle coppie per X (calcolata dalla TM Pair), per ogni $L \subseteq X$ definiamo*

$$\exists(L) = \{x: X | \exists y: X.(Pair\ x\ y) \in L\}$$

tale sottoinsieme e' detto **la quantificazione esistenziale** *di L. In modo analogo si definisce la quantificazione universale di L.*

Proposition 7.8 *Il* **predicato** T **di Kleene**, $T(M, x, y) \overset{\Delta}{\iff}$ *"y e' la computazione della TM M con input x", e' decidibile.*

Il predicato di Kleene e' utile per dimostrare alcuni dei seguenti risultati.

Theorem 7.9

- *L e' decidibile sse L ed il suo complementare $\overline{L}$ sono semidecidibili.*
- *L e' semidecidibile sse $L = \exists(R)$ per qualche R decidibile.*

Proof Le direzioni difficili di tali caratterizzazioni si dimostrano dando degli algoritmi che usano il predicato di Kleene.

- supponiamo che la TM $M1$ accetta L e la TM $M2$ accetta $\overline{L}$ allora una TM M che decide L e' data da

  ```
  M(x) = for y string do
           if T(M1,x,y) and y accepting then accept
           if T(M2,x,y) and y accepting then reject
  ```

  prima o poi troveremo un y per cui uno dei due if e' vero.

- supponiamo che M accetta L, allora definiamo

  ```
  R(x,y) = T(M,x,y) and y accepting
  ```

  chiaramente R e' decidibile e $L = \exists(R)$.

∎

Theorem 7.10 *I sottoinsiemi semidecidibili sono chiusi per intersezione, unione e quantificazione esistenziale.*

Proof Siano $M1$ ed $M2$ TM che accettano i linguaggi L_1 ed L_2 allora

- il linguaggio $L_1 \cap L_2$ e' accettato dalla TM `M(x) = if M1(x) then if M2(x) the accept`
- il linguaggio $L_1 \cup L_2$ e' accettato dalla TM

  ```
  M(x) = for y string do
           if T(M1,x,y) and y accepting then accept
           if T(M2,x,y) and y accepting then accept
  ```

  una definizione alternativa di M e' data dalla *composizione parallela* di $M1$ ed $M2$

Sia M una TM che accetta L, allora $\exists(L)$ e' accettato dalla TM

```
M'(x) = for (y,z) string do
         if T(M,(x,y),z) and z accepting then accept
```

si osservi che si ha un unico ciclo for, che enumera tutte le (codifiche) di coppie di stringhe. ∎

E' immediato vedere che i sottoinsiemi decidibili sono chiusi per intersezione, unione e complemento.

Theorem 7.11 *Le seguenti asserzioni su $L \subseteq N$ sono equivalenti*

- *L e' semidecidibile, cioe' il dominio di una funzione parziale calcolabile f*
- *L e' l'immagine di una funzione parziale calcolabile g*
- *L e' vuoto o e' l'immagine di una funzione totale calcolabile h.*

Proof Facciamo vedere una catena di implicazioni. Data f definiamo `g(x)=if f(x) defined then x`.
Data g calcolata dalla TM M e $z \in L$ definiamo `h(x,y)=if T(M,x,y) and y accepting then g(x) else z`.
Data h definiamo `f(x)=for y string do if h(y)=x then return(x)`. ∎

Theorem 7.12 *Le seguenti asserzioni su $L \subseteq X$ sono equivalenti*

- *L e' decidibile, cioe' la funzione caratteristica f e' calcolabile*
- *L e' finito o e' l'immagine di una funzione totale calcolabile h strettamente crescente [rispetto all'ordine lineare dato da $x \leq y$ sse $|x| < |y|$ o ($|x| = |y|$ e $x \leq_{lex} y$), dove $\leq_{lex}$ e' l'ordine lessicografico indotto da un qualche ordine lineare su Σ].*

Mentre la tecnica della riducibilita' serve essenzialemnte a dimostrare risultati negativi (non decidibilita' e non semidecidibilita' di problemi), le proprieta' di chiusura servono soprattutto a dimostrare la decidibilita' e semidecidibilita' di un problema, in alternativa alla descrizione di una TM (o di un algoritmo ad alto livello).

8 Complessita'

Questa sezione elenca i risultati visti a lezione. In alcuni casi i risultati vengono enunciati in modo leggermente diverso dal libro di testo, evitando riferimenti espliciti al modello di computazione.

- Modelli di calcolo. Abbiamo introdotto i seguenti modelli di calcolo: le TM deterministiche (DTM), the TM nondeterministiche (NDTM),e le Random Access Machines (RAM). Per le TM abbiamo introdotto alcune varianti: input read-only, output write-only, limitazione sul numero di nastri (di lavoro).
- Risorse di calcolo. Per le TM abbiamo introdotto la risorsa tempo (misurata in numero di passi di computazione) e la risorsa spazio (misurato in numero di celle visitate sui nastri di lavoro). Per le RAM abbiamo considerato solo la risorsa tempo.

I modelli di computazione formalizzano (in modi diversi) la nozione di *algoritmo*. Quello che si vuole studiare in complessita' computazionale non e' la complessita' degli algoritmi, ma quella dei *problemi*. E' quindi necessario fissare una corrispondenza del tipo "l'algoritmo M risolve il problema L". Abbiamo introdotto le seguenti corrispondenze:

- un linguaggio $L \subseteq \Sigma^*$ **e' deciso** da una DTM M con alfabeto di input Σ
- un linguaggio $L \subseteq \Sigma^*$ **e' accettato** da una DTM M con alfabeto di input Σ
- un linguaggio $L \subseteq \Sigma^*$ **e' deciso** da una NDTM M con alfabeto di input Σ
- una funzione parziale $f: \Sigma^* \rightharpoonup \Sigma^*$ **e' calcolata** da una DTM M con alfabeto di input/output Σ
- una funzione parziale $f: Z^* \rightharpoonup Z$ **e' calcolata** da una RAM M.

Fissate la nozione di algoritmo (mediante la scelta di un modello di calcolo), una corrispondenza tra problemi ed algoritmi, ed una risorsa di calcolo (per il modello prescelto), resta definita una *classe di complessita'* (parametrizzata rispetto ad un upper-bound sulla risorsa di calcolo), ovvero l'insieme dei problemi L che sono *risolubili* con un algoritmo M che usa la risorsa entro il limite consentito. Le classi di complessita' comunemente considerate sono:

- $L \in TIME(f) \subseteq \Sigma^*$ sse esiste una DTM che decide L e che con input di lunghezza n lavora in tempo $\leq f(n)$ (in genere si suppone che $f: N \to N$ sia una funzione almeno lineare).
- $L \in NTIME(f) \subseteq \Sigma^*$ sse esiste una NDTM che decide L e che con input di lunghezza n lavora in tempo $\leq f(n)$, la differenza essenziale rispetto alla classe $TIME(f)$ e' l'uso del modello di calcolo non-deterministico, che richiede adattamenti sia alla definizione di M decide L che al calcolo della risorsa tempo.
- $L \in SPACE(f) \subseteq \Sigma^*$ sse esiste una DTM con input read-only che decide L e che con input di lunghezza n usa spazio di lavoro $\leq f(n)$.
- $L \in NSPACE(f) \subseteq \Sigma^*$ sse esiste una NDTM con input read-only che decide L e che con input di lunghezza n usa spazio di lavoro $\leq f(n)$.
- $h \in TIME(f) \subseteq \Sigma^* \to \Sigma^*$ sse esiste un DTM con che calcola h e che con input di lunghezza n lavora in tempo $\leq f(n)$.

  $h \in SPACE(f) \subseteq \Sigma^* \to \Sigma^*$ sse esiste un DTM con input read-only ed output write-only che calcola h e che con input di lunghezza n usa spazio di lavoro $\leq f(n)$.

  $h \in TIME^{RAM}(f) \subseteq Z^* \to Z$ sse esiste una RAM che calcola h e che con input di lunghezza n (usando la codifica binaria) lavora in tempo $\leq f(n)$. In questo caso e' molto importante precisare il tipo di operazioni aritmentiche disponibili su una RAM (si consente solo somma, sottrazione e divisione per 2).

In alcuni casi si vuole imporre delle limitazioni sul numero di nastri (di lavoro) di una TM, scriveremo $TIME_k(f)$ (e similmente per le altri classe di complessita' basate sul modello di calcolo delle TM) per indicare che si usano TM con $\leq k$ nastri (di lavoro).
Riportiamo i risultati fondamentali sulle classi di complessita' basate su modello di calcolo delle TM.

- riduzione a TM con un nastro (di lavoro).

  $TIME(f) \subseteq TIME_1(O(f^2))$ [se f e' almeno lineare], un risultato analogo vale per $NTIME$.

  $SPACE(f) \subseteq SPACE_1(f)$, un risultato analogo vale per $NSPACE$.

  L'idea e' di rappresentare i k nastri (di lavoro) di M con k tracce del nastro (di lavoro) di M', cambiando in modo opportuno l'afabeto di lavoro. A parte una fase di inizializzazione, M' richiede $O(f)$ mosse per simulare una mossa di M.
- $TIME_k(f) \subseteq SPACE_k(f)$ [se f e' almeno lineare].
- teorema di speed-up.

  $TIME(f) \subseteq TIME(g)$ se $f \in O(g)$ [e $g(n)/n \to \infty$], un risultato analogo vale per $NTIME$. L'idea e di simulare molti passi di M con pochi passi di M' usando un alfabeto di lavoro molto piu' grande.
- teorema di tape compression.

  $SPACE(f) \subseteq SPACE(g)$ se $f \in O(g)$, un risultato analogo vale anche per $NSPACE$. L'idea e' di rappresentare una sequenza di caratteri di lavoro di M con un carettere di lavoro di M'. (Al contrario del teorema di speed-up, il teorema di tape compression vale anche per classi di complessita' di funzioni.)
- classi naturali di complessita'
  - $L = SPACE(\log n)$
  - $NL = NSPACE(\log n)$
  - $P = \cup_{k \in N} TIME(n^k)$
  - $NP = \cup_{k \in N} NTIME(n^k)$
  - $PSPACE = \cup_{k \in N} SPACE(n^k)$
  - $NPSPACE = \cup_{k \in N} NSPACE(n^k)$
  - $EXP = \cup_{k \in N} TIME(2^{(n^k)})$
- relazioni tra RAM e TM (e robustezza della classe P)

  $TIME^{TM}(f) \subseteq TIME^{RAM}(O(f))$ se f e' almeno lineare [modulo biiezione tra stringhe ed interi].

  $TIME^{RAM}(f) \subseteq TIME^{TM}(O(f^3))$ se f e' almeno lineare [modulo biiezione tra stringhe ed interi]. L'idea e' di usare un nastro della TM come log file, in cui si riportano in ordine cronologico tutte le modifiche effettuate allo stato della RAM, e che viene consultato tutte le volete che si deve simulare un accesso ad un registro della memoria (quelli di input si trovano invece sul nastro di input read-only). Tale nastro contiene al massimo $f(n)$ records.

Per dimostrare che ogni record ha dimensione $O(f)$ si usa il seguente Lemma sulle RAM: dopo t passi i registri di memoria contengono interi la cui rappresentazione e' limitata da una funzione lineare in t e la dimensione degli input. (tale Lemma non vale se si include la moltiplicazione tra le operazioni aritmetiche di una RAM).

- teorema di gerarchia spazio

  $SPACE(g) \not\subseteq SPACE(f)$, se $g \notin O(f)$ e g proper (e $g \geq \log$). Dove proper vuol dire che esiste una TM con input read-only che per ogni input di dimensione n usa esattamente $g(n)$ celle sul nastro di lavoro. Il controesempio a $SPACE(g) \subseteq SPACE(f)$ e' dato dal linguaggio

  $$D_g^{space} = \{w | \exists M, k. w = M; 1^k \wedge\ U(M, w) \text{ rigetta in spazio} \leq g(|w|)\}$$

- teorema di gerarchia tempo

  $TIME(g) \not\subseteq TIME(f)$, se $g \notin O(f^2)$ e g proper. Dove proper vuol dire che esiste una TM con che per ogni input di dimensione n termina in esattamente $g(n)$ passi. Il controesempio a $TIME(g) \subseteq TIME(f)$ e' dato dal linguaggio

  $$D_g^{time} = \{w | \exists M, k. w = M; 1^k \wedge\ U(M, w) \text{ rigetta in tempo} \leq g(|w|)\}$$

- relazioni di inclusioni tra classi di complessita'

  - $TIME(f) \subseteq NTIME(f)$ e $SPACE(f) \subseteq NSPACE(f)$. Dimostrazione immediata.
  - $NTIME(f) \subseteq SPACE(f)$. Si esaminano tutte le possibili computazioni della NDTM M che lavora intempo f usando un contatore fino a $d^{f(n)}$, dove d e' il grado di non-determinismo di M, per risolvere le scelte non-deterministiche.
  - $NSPACE(f) \subseteq \cup_k TIME(k^{(\log n)+f(n)})$. Si usa il reachability method. Cioe' si riformula il problema di determinare l'accettazione di una stringa x da parte di una NDTM M con nastro read-only che lavora in spazio f, al problema dell'esistenza di un cammino dalla configurazione iniziale alla configurazione finale (canonica) nel grafo delle ID di M di dimensione $\leq f(n)$.
  - teorema di Savitch

    $NSPACE(f) \subseteq SPACE(f^2)$ (se $f \geq \log$). Si usa il reachability method.

- relazioni di inclusione tra classi naturali di complessita'

  $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXP$, le uniche inclusioni strette (che si riescono a dimostrare usando i teoremi di gerarchia) sono $NL \subset PSPACE$ e $P \subset EXP$.

- riducibilita' logspace $\leq_L$ e polytime $\leq_P$

  Proprieta' riflessiva e transitiva delle nozioni di riducibilita' (per dimostrare la transitivita' di $\leq_L$ occorre usare un algoritmo demand driven per implementare la composizione delle funzioni di riduzione log-space).

  Chiusura delle classi naturali di complessita' rispetto a $\leq_L$, cioe' $L_1 \leq_L L_2 \in C$ implica $L_1 \in C$. Un risultato analogo vale per $\leq_P$ se $C \supseteq P$.

- Problemi C-completi e C-hard (rispetto ad una riduzione $\leq$)

  H e' RE-completo per $\leq_L$ (addirittura per riduzioni in tempo lineare). Non esistono problemi R-completi per $\leq_P$. Problemi C-completi per $\leq_L$ per le classi naturali di complessita', ottenuti come varianti dell'halting problem.

- proprieta' di chiusura delle classi di complessita' rispetto ad operazioni logiche

  Chiusura delle classi di complessita' per intersezione. Chiusura delle classi $TIME(f)$ e $SPACE(f)$ per complementazione (se $f \geq \log$). Chiusura delle classi $NTIME(f)$ e $NSPACE(f)$ per disgiunzione (se $f \geq \log$). Le dimostrazioni sono adattamenti delle analogo proprieta' di chiusura di R ed RE.

  Teorema di Immerman: chiusura di $NSPACE(f)$ per complementazione (se $f \geq \log$ e proper).

- proprieta' di chiusura delle classi naturali di complessita' rispetto a quantificazione limitata

  L, NL e P sono chiuse per quantificazione esistenziale log-limitata, cioe' $L \in P$ implica $\exists_{\log}(L) \triangleq \{x | \exists y. \langle x, y \rangle \in L \wedge |y| \leq \log |x|\}$. NP, $PSPACE$ ed EXP sono chiuse per quantificazione esistenziale poly-limitata.

- caratterizzazione di NP in termini di P.

  Un linguaggio sta in NP sse e' la quantificazione esistenziale poly-limitata di un linguaggio in P. Data una NDTM M che lavora in polytime L_M si ottiene come quantificazione esistenziale poly-limitata del linguaggio $\{\langle x, y\rangle | y$ codifica una computazione accettante di M con input $x\}$.

  Questo risultato si puo' riformulare dicendo che i problemi NP si possono vedere come dei problemi di esistenza di una soluzione in uno spazio di possibili soluzioni di cardinalita' esponenziale nella dimensione dell'istanza del problema, pero' data una soluzione potenziale e' facile erificare se e' una soluzione accettabile.

- esempi di problemi completi.

  CVP (Circuit Value Problem) e' P-completo, SAT (Satisfability of boolean formulas in CNF) e' NP-completo. Dimostrazione mediante codifica di tabelle di computazione.

  REACHABILITY e $2-SAT$ sono NL-completi. $HORNSAT$ e' P-completo, $3-SAT$ e $CIRCUIT\ SAT$ sono NP-completi, QBF (verita' di una Quantified Boolean Formula) e' $PSPACE$-completo.

- Altri esempi di problemi NP-completi (dimostrazione per riduzione).

Contenuto

Problemi ed algoritmi, macchine di Turing e random access machines (tesi di Church), calcolabilita' (macchine universali, problemi indecidibili, riduzioni), classi di complessita' (P, NP), riduzioni e completezza, problemi P-completi e NP-completi.
Capitoli del libro di testo: 1, 2, 3 (escluso recursive inseparability), 7 (integrato con note su teoremi di gerarchia), 8 (escluso 8.3), 9.

References

[Pap94] C.M. Papadimitriou, Computational Complexity. Addison-Wesley, 1994.

Tale libro e' disponibile in biblioteca, e puo' essere sostituito da altri libri che coprano gli argomenti trattati nei seguenti capitoli: 1-5, 7-9, 11, 13, 15.

Argomenti delle lezioni (Moggi)

- Macchina di Turing M (inclusi dettagli "irrilevanti"), configurazione istantanea (finita ed infinita) ID_M, relazione di transizione $\to_M$; nozioni derivate: M decide L, M accetta L, M calcola f.

  O-notazione: $O(f)$, $\Omega(f)$, $\theta(f)$. codifica di numeri, matrici e grafi; irragionevolezza della codifica unaria dei numeri. Analisi degli algoritmi e TM (problema, codifica e trattabilita').

- CH 2 Varianti della TM: a k nastri, input read-only, output write-only; M richiede tempo u con input x, classi di complessita' tempo $TIME_k(t)$ e $TIME(t)$; M richiede tempo u con input x (varianti), classi di complessita' spazio $SPACE(s)$ e $SPACE_k(s)$. palindrome in $TIME(O(n))$ e $SPACE(O(logn))$.

- $TIME_k(f) \subseteq TIME_1(O(f^2))$ [Thm 2.1], tecnica della simulazione $SPACE_k(f) \subseteq SPACE_1(f)$; $TIME_k(f) \subseteq SPACE_k(O(f))$.

  Speed-up [Thm 2.2] $TIME(f) \subseteq TIME(g)$ se $f \in O(g)$ e $g(n)/n \to \infty$, simulare molti passi con pochi, trattare gli input di dimemsione $\leq n_0$ usando lo stato finito di una TM, la classe naturale P.

  Tape-compression [Thm 2.3] $SPACE(f) \subseteq SPACE(g)$ se $f \in O(g)$, (al contrario dello speed-up) vale anche per macchine che calcolano funzioni.

- Corroborazione della Tesi di Church (estesa): le funzioni calcolabili in tempo polinomiale usando TM e RAM coincidono.

  Descrizione di una RAM: programma, istruzioni (read, store, load, add, sub, half, jump, halt), program counter, inputs, accumulatore e registri, indirizzi diretti ed indiretti. Semantica di un operando ed una istruzione, stato della RAM (k, R), relazione di transizione $\to_{\Pi,I}$, funzione da sequenze di interi ad interi calcolata da una RAM.

  Dimensione dell'input di una RAM e complessita' tempo (istruzioni a costo unitario); il problema causato dalla moltiplicazione: $x \mapsto x^x$ e' calcolabile in tempo polinomiale da una RAM con moltiplicazione (ma non da una TM), tuttavia la moltiplicazione e' realizzabile in tempo polinomiale.

  Lemma di upper-bound sul contenuto dei registri dopo t passi $l(r) \leq t + l(I) + l(\Pi)$ [non vale in presenza della moltiplicazione].

- Simulazione di TM tramite RAM [Thm 2.4] $L \in TIME(f)$ implies ϕ_L calcolata da una RAM in tempo $O(f)$, codifica numerica di Σ, un registro per ogni casella della TM (accesso diretto alle caselle).

  Simulazione di RAM tramite TM [Thm 2.5] ϕ calcolabile da una RAM in tempo f implica ϕ calcolabile da una TM in tempo $O(f^3)$, codifica binaria di input e registri, nastro con il log delle modifiche ai registri [dimensione $O(f^2)$] e determinazione del contenuto di un registro (backup incrementale).

  Le TM non-deterministiche, M accetta L, M richiede tempo/spazio $\leq u$ con input x, (differenze ed analogie con DTM); classi di complessita' non-deterministiche $NTIME(f)$ e $NSPACE(f)$; risultati di base per le classi non-deterministiche, ottenuti riciclando le dimostrazione degli stessi risultati per le classi deterministiche, la classe NP, esempi TSP(D).

- Relazioni tra classi deterministiche e non-deterministiche $TIME(f) \subseteq NTIME(f) \subseteq SPACE(f), \cup_{c>1} TIME(c^f)$ [Thm 2.6] ricerca esaustiva.

La classe $NSPACE(\log)$, esempi reachability (come forzare la terminazione).

Esercizi: Dimostrare che DTM ed NDTM accettano la stessa classe di linguaggi.

CH 3 Codifica di una DTM ad un nastro M ed una stringa $x \in \Sigma^*$ (come stringhe di un alfabeto prefissato), DTM universale U (per semplicita' a piu' nastri). U=processore, M=programma, x=dati.

Linguaggi/problemi decidibili R (ricorsivi), semidecibili RE (ricorsivamente enumerabili). Esistenza di problemi non-decidibili (e non-semidecidibili) per motivi di cardinalita'; il problema H dell'halt, semidecidibilita' di H [Prop 3.1], non-decidibile di H [Thm 3.2] tecnica della diagonalizzazione.

Altri problemi non-decidibili [Prop 3.2], riduzione tra problemi $L_1 \leq L_2$, proprieta' di $\leq$, completezza di H tra i problemi semidecidibili.

Relazioni tra problemi decidibili e semidecidibili [Prop 3.3, 3.4]: $R \subseteq RE$, L in R implica $\bar{L}$ in R, L e $\bar{L}$ in RE implica L in R (tecnica dell'interleaving).

- Caratterizzazione di RE in termini di R, il predicato T di Kleene.

  Caratterizzazione alternativa di RE [Prop 3.5] (tecnica del dovetailing).

  Linguaggio $L(M)$ accettato da una TM M, proprieta' "estensionale" di TM $\{M|L(M) \in C\}$, Teorema di Rice [Thm 3.2].

  Altri risultati (senza dimostrazione): Teorema s-m-n [Ex 3.4.6], controparte a livello di linguaggi di programmazione. Esempi di linguaggi in Σ_1 e Π_2.

- Proper complexity function [Def 7.1], timers and markers. Semplificazione delle dimostrazioni di alcuni risultati restringendosi a proper complexity functions.

  Teorema di gerarchia spazio [enunciato forte]: $SPACE(g) \not\subseteq SPACE(f)$, se $g \notin O(f)$ e g proper (e $g \geq \log$). Teorema di gerarchia tempo [enunciato forte]: $TIME(g) \not\subseteq TIME(f)$, se $g \notin O(f^2)$ e g proper.

  Classi di complessita' spazio, tempo, deterministiche e non. *coC* di una classe di linguaggi. Proprieta' di chiusura facili (intersezione, unione, complemento), per analogia con R [Ex 7.4.5]. Inclusioni tra classi facili da dimostrare [Thm 7.4]: $X(f) \subseteq NX(f)$, $NTIME(f) \subseteq SPACE(f)$, $NSPACE(f) \subseteq \cup_k TIME(k^{(\log n)+f(n)})$ tramite reachability method!.

- Teorema di Savitch [Thm 7.5]: $reachability \in SPACE(\log^2 n)$, $NSPACE(f) \subseteq SPACE(f^2)$ (se f proper e $f \geq \log$), $PSPACE = NPSPACE$.

  Classi naturali di complessita': L, NL, P, NP, $PSPACE$, $NPSPACE$, $EXP = \cup_k TIME(2^{n^k})$.

  Dimostrazione dei teoremi di gerarchia (note aggiuntive): codifica con padding delle TM ad un nastro con alfabeto di input prefissato (quello della macchina universale), metodo di diagonalizzazione corretto (nessuna codifica dell'input), uso di una proper g come marker/timer.

  Raffinamenti della nozione di riduzione $L_1 \leq L_2$ per classificare i problemi nelle classi naturali di complessita': log-space [Def 8.1] e poly-time, problemi complete ed hard. H e' RE-complete rispetto a riduzioni log-space.

  $\leq_L \subseteq \leq_P$ [Prop 8.1]. Transitivita' di $\leq_L$ [Prop 8.2].

- Ogni L non banale e' R-hard rispetto a $\leq$. Chiusura di classi naturali rispetto a $\leq_L$. Esempi di problemi logici [vedi CH 4]: SAT, circuit value CVP, circuit SAT. Problemi P-completi ed NP-completi tramite varianti di H.

- Rappresentazione di computazioni mediante tabelle. P-completezza di CVP e NP-completezza di SAT.

- Caratterizzazione di NP con P [Prop 9.1] (analogia con RE).

  Varianti di SAT: $SAT \leq 3SAT \in NP$, $2SAT \in NL \subseteq P$, $HORNSAT \in P$ [Thm 4.2] usa modello di Herbrand, CIRCUIT SAT (riduzione a SAT). $SAT \leq NAESAT \in NP$ (variante di $SAT \leq 3SAT$).

  Problemi su grafi NP-completi: CLIQUE (e simili), MAXCUT (MINCUT e' in P), HAMILTON PATH, TSP(D), 3-COLORABILITY.

  Problemi su interi: INTEGER PROGRAMMING (linear programming e' in P), KNAPSACK.