10 Complessita'

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

- Modelli di calcolo. Abbiamo introdotto i seguenti modelli di calcolo: le TM deterministiche (DTM), the TM non-deterministiche (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.

La definizione di TM differisce dal libro di testo, in quanto si distingue l'alfabeto di input/output (che e' associato al problema/linguaggio e non ad un particolare automa/macchina) e l'alfabeto di lavoro. Diamo la definizione di DTM ad un nastro, e l'automa ad essa associato.

Definition 10.1 *Una DTM M con alfabeto di input Σ e' specificata da una quintupla $(\Gamma, K, s, \flat, \delta)$ dove*

- *Γ e' un insieme finito, detto alfabeto di lavoro, t.c. $\Sigma \subset \Gamma$*
- *K e' un insieme finito, detto insieme degli stati*
- *$\flat \in \Gamma - \Sigma$ e' il carattere blank*
- *$s \in K$ e' lo stato iniziale*
- *$\delta\colon (K \times \Gamma) \to (\overline{K} \times \Gamma \times \{\leftarrow, -, \rightarrow\})$ e' la funzione di transizione, mentre $\overline{K}$ e' K esteso con degli stati speciali $\{yes, no\}$ (quando una DTM e' usata come traduttore, basta avere lo stato speciale h per fermare la computazione).*

L'automa A_M associato alla DTM M ha come insieme di input $I = \Sigma^*$, come insieme di output $O = \{yes, no\}$, come insieme di stati interni S l'insieme ID_M delle configurazioni instantanee di M ovvero $\alpha q \beta$ con $q \in K$ e $\alpha, \beta \in \Gamma^\omega$ sequenze infinite quasi ovunque $\flat$: α e' il contenuto del nastro a sinistra della testina, mentre β il contenuto del nastro a destra della testina (incluso il carattere sotto la testina). La relazione di transizione $\xrightarrow[M]{} \subset (I + S) \times (S + O)$ e' definita nel modo seguente

- $\alpha \xrightarrow[M]{} \flat^\omega s \alpha \flat^\omega$ *se* $\alpha \in I$
- $\alpha q b \beta \xrightarrow[M]{} yes$ *se* $\delta(q, b) = (yes, \_, \_)$
- $\alpha q b \beta \xrightarrow[M]{} no$ *se* $\delta(q, b) = (no, \_, \_)$
- $\alpha a q b \beta \xrightarrow[M]{} \alpha q' a b' \beta$ *se* $\delta(q, b) = (q', b', \leftarrow)$
- $\alpha q b \beta \xrightarrow[M]{} \alpha q' b' \beta$ *se* $\delta(q, b) = (q', b', -)$
- $\alpha q b \beta \xrightarrow[M]{} \alpha b' q' \beta$ *se* $\delta(q, b) = (q', b', \rightarrow)$

Remark 10.2 Molte varianti di TM si ottengono modificando il requisito su δ, per esempio

- una DTM a $m > 1$ nastri corrisponde ad una funzione di transizione $\delta\colon (K \times \Gamma^m) \to (\overline{K} \times (\Gamma \times \{\leftarrow, -, \rightarrow\})^m)$, quindi ognuna delle m testine opera in modo indipendente dalle altre.
- una TM read-only corrisponde ad una δ che riscrive il carattere letto, mentre una TM write-only corrisponde ad una δ che non permette di spostare la testina a sinistra $\leftarrow$.
- un traduttore e' una DTM con input (il primo nastro) read-only, l'output (l'ultimo nastro) write-only su cui si puo' scrivere solo caratteri in Σ, ed uno stato speciale h (invece di $\{yes, no\}$) che indica la fine della computazione. L'automa associato ad un traduttore va modificato per tenere conto che l'insieme ddi output e' $O = \Sigma^*$.

- una NDTM corrisponde ad una funzione di transizione $\delta: (K \times \Gamma) \to \mathcal{P}(\overline{K} \times \Gamma \times \{\leftarrow, -, \rightarrow\})$ cioe' ad ogni passo si ha un certo numero (necessariamente finito) di possibili transizioni.

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 $M \stackrel{\Delta}{\iff}$ l'automa associato A_M calcola la funzione caratteristica di L, cioe' $f(x) = yes$ se $x \in L$, no altrimenti

- un linguaggio $L \subseteq \Sigma^*$ **e' accettato** da una DTM $M \stackrel{\Delta}{\iff}$ l'automa associato A_M calcola la funzione semi-caratteristica di L, cioe' $f(x) = yes$ se $x \in L$, indefinita altrimenti.

- un linguaggio $L \subseteq \Sigma^*$ **e' deciso/accettato** da una NDTM $M \stackrel{\Delta}{\iff}$ l'automa non-deterministico associato A_M gode della seguente proprieta' $\forall x \in \Sigma^*. x \in L \leftrightarrow x \xrightarrow[M]{} yes$, cioe' esiste almeno una computazione accettante.

- una funzione parziale $f: \Sigma^* \rightharpoonup \Sigma^*$ **e' calcolata** da un traduttore $M \stackrel{\Delta}{\iff}$ l'automa associato A_M calcola f.

- una funzione parziale $f: Z^* \rightharpoonup Z$ **e' calcolata** da una RAM $M \stackrel{\Delta}{\iff}$ [usare automa associato a RAM].

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:

- $L \in TIME(f) \subseteq \mathcal{P}(\Sigma^*)$ 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), cioe' $\forall x \in \Sigma^*. \forall m \in N. \forall id \in ID_M. x \xrightarrow{m} id \supset m \leq f(|x|)$.

- $L \in NTIME(f) \subseteq \mathcal{P}(\Sigma^*)$ sse esiste una NDTM che decide L e che con input di lunghezza n lavora in tempo $\leq f(n)$, la definizione di "lavora in tempo $\leq f(n)$" per le DTM si applica anche alle NDTM, ed in tal caso richede che tutte le computazioni devono terminare in $\leq f(n)$ passi.

- $L \in SPACE(f) \subseteq \mathcal{P}(\Sigma^*)$ sse esiste una DTM con input read-only che decide L e che con input di lunghezza n **usa spazio di lavoro** $\leq f(n)$. Occorre usare delle transizioni pesate che misurano le celle usate sui nastri di lavoro (se si assume che le testine dei nastri di lavoro scrivono sempre caratteri diversi da blank, basta contare quanti di tali caratteri ci sono nella configurazione istantanea id).

- $L \in NSPACE(f) \subseteq \mathcal{P}(\Sigma^*)$ sse esiste una NDTM con input read-only che decide L e che con input di lunghezza n usa spazio di lavoro $\leq f(n)$.

- $h \in TIME(f) \subseteq \mathcal{P}(\Sigma^* \to \Sigma^*)$ sse esiste un traduttore (cioe' una DTM con input read-only ed output write-only) che calcola h e che con input di lunghezza n lavora in tempo $\leq f(n)$.

  $h \in SPACE(f) \subseteq \mathcal{P}(\Sigma^* \to \Sigma^*)$ sse esiste un traduttore che calcola h e che con input di lunghezza n usa spazio di lavoro $\leq f(n)$.

  $h \in TIME^{RAM}(f) \subseteq \mathcal{P}(Z^* \to Z)$ 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 volte 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

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

  dove M_U e' una opportuna variante della DTM universale che si aspetta come primo argomento la codifica $\langle M\rangle$ di una DTM con input read-only ed 1 nastro di lavoro (e il nastro di input di M_U fa anche da input per M). Per dimostrare il risultato si deve sfruttare la seguente relazione tra lo spazio di lavoro usato da M_U e da M: $s_{M_U}(\langle M\rangle, x) \leq c_M * s_M(x)$, dove $s_M(x)$ e' lo spazio di lavoro usato dalla computazione di M con input x.
- 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

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

  La diversa formulazione del teorema di gerarchia tempo e' dovuta al costo quadratico per ridursi a DTM ad un solo nastro. M_U e' la DTM universale. Per dimostrare il risultato si deve sfruttare la seguente relazione tra il tempo di lavoro di M_U e di M: $t_{M_U}(\langle M\rangle, x) \leq c_M * t_M(x)$, dove $t_M(x)$ e' il numero di passi della computazione di M con input x.
- 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 in tempo 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.

  - $H^P = \{(\langle M\rangle, x, t)|\ M_U(\langle M\rangle, x)$ accetta in tempo $\leq |t|\}$ e' P-completo per la riduzione $\leq_L$. La stringa t e' usata come timer.
  - $H^{NP} = \{(\langle M\rangle, x, t)|$ esiste una computazione di $M_U(\langle M\rangle, x)$ che accetta in tempo $\leq |t|\}$ e' NP-completo per la riduzione $\leq_L$. M_U e' una variante della TM universale, che si aspetta come primo argomento la codifica $\langle M\rangle$ di una NDTM ad 1 nastro. Poiche' M e' non-deterministica, anche M_U e' non-deterministica.
  - $H^{PSPACE} = \{(\langle M\rangle, x, t)|\ M_U(\langle M\rangle, x)$ accetta ed usa un spazio di lavoro $\leq |t|\}$ e' $PSPACE$-completo per la riduzione $\leq_L$. La stringa t e' usata per il marking (del nastro di lavoro).
  - $H^{NL} = \{(\langle M\rangle, x, t)|$ esiste una computazione di $M_U(\langle M\rangle, x)$ accetta ed usa un spazio di lavoro $\leq \log |t|\}$ e' NL-completo per la riduzione $\leq_L$. M_U e' una variante della TM universale, che si aspetta come primo argomento la codifica $\langle M\rangle$ di una NDTM con input read-only ed 1 nastro di lavoro.

• 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 $\exists_{\log}(L) \stackrel{\Delta}{=} \{x|\exists y.\langle x, y\rangle \in L \wedge |y| \leq \log |x|\}$. 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 – Conjunctive Normal Form) 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

- Macchina di Turing M (inclusi dettagli "irrilevanti"), configurazione istantanea (finita ed infinita) ID_M, relazione di transizione $\rightarrow_M$; 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 $SPACE(O(\log n))$.

- $TIME_k(f) \subseteq TIME_1(O(f^2))$ [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 $g(n)/n \rightarrow \infty$, 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 $\rightarrow_{\Pi,I}$, 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.