Elementi di Teoria della Computazione

Prof.ssa M. Anselmo e Prof. R. Zaccagnino

a.a. 2022/23

Classe 3: matricole congrue 2 modulo 3

28 febbraio 2023

Presentazioni

- M. Anselmo:

  https://docenti.unisa.it/001367

- Orario di ricevimento: da concordare via email (anche in gruppi) nel mio studio (o eventualmente via Teams)

- Il mio studio è il n° 57 (o meglio 101) al 4° piano della Stecca 7 (fra l'aula F8 e Farmacia)

R. Zaccagnino (da giovedì 2/3/23)

Informazioni e materiale del corso

- Informazioni pubbliche generali:

  https://docenti.unisa.it/001367

  Didattica: Scheda dell'insegnamento

  Materiale e Risorse: Programma, compiti di esame, ...

- Informazioni per voi del corso: sulla piattaforma http://elearning.**informatica**.unisa.it/el-platform/

  Contenuto lezioni; esercizi da svolgere, …

O via email istituzionale!

- Inoltre trovate orario e calendario della vostra classe su

  http://corsi.unisa.it/05121/didattica/orari (settimana per settimana)

  https://easycourse.unisa.it/AgendaStudenti/

  https://corsi.unisa.it/informatica/didattica/calendari

Svolgimento del corso

9 CFU di lezione frontale ed esercitazioni = 72 ore

- Il corso prevede **72 ore** di lezione frontale di carattere teorico o esercitativo, che saranno svolte secondo l'orario previsto, oppure in eventuali ore di recupero, di cui si darà notizia in classe e sui canali soliti di comunicazione.
- M. Anselmo (6 CFU = 48 ore), R. Zaccagnino (3 CFU = 24 ore).
- Orario:
  - martedì ore 11 – 13 F8
  - giovedì ore 15:30 – 17:30 F8
  - venerdì ore 11 – 13 F8

- Vacanze di Pasqua: 6 - 11 aprile (giovedì – martedì)
- **Ultima lezione** prevista: giovedì 1 giugno 2023

- Altre festività:

  giovedì 16 marzo e martedì 25 aprile (1 maggio è lunedì)

Libri di testo

I libri di **testo di riferimento** sono:

[Sip] Michael Sipser, INTRODUZIONE ALLA TEORIA DELLA COMPUTAZIONE, Apogeo education, Maggioli editore (traduzione italiana di Introduction to the Theory of Computation, 3rd Edition).

[HMU] J. Hopcroft, R. Motwani, J. Ullman, AUTOMI, LINGUAGGI E CALCOLABILITÀ, Addison Wesley Pearson Education Italia s.r.l, terza Edizione 2009 (Capitolo 2, sezioni 2.2, 2.3 e 2.5. Si suggerisce lettura del Capitolo 1).

Altri libri di consultazione sono:

[CLRS] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduzione agli Algoritmi*, seconda edizione, McGraw Hill.

[KT] Kleinberg, Tardos. *Algorithm Design*. Pearson Addison Wesley.

Esami

- L'esame consiste di una **prova scritta** e di un **orale** cui si accede solo dopo il superamento di quella scritta con circa la metà del punteggio totale. La prova scritta è composta da domande a risposta aperta. Un esempio di compito con soluzione sarà disponibile.

- Sono previste 2 prove intercorso:
  1. **dopo Pasqua (Prof. Zaccagnino)**
  2. **a fine corso**: 9 giugno ore 15, aule P3+P4

E' richiesto superamento di una soglia nella prima prova per poter accedere alla seconda. In alcuni casi potrà essere richiesto un orale finale (da noi o da voi).

- Gli studenti interessati alle prove di esame devono prenotarsi su **Esse3** entro il termine utile (5 giorni prima). Ricordo inoltre che è possibile e doveroso cancellare la propria prenotazione qualora si decida di non partecipare, per evitare un inutile spreco di risorse.

Date esami

Disponibili su Easycourse e sulla Bacheca di Esse3

- **9 giugno** ore 15 P3+P4:

  pre-appello e seconda prova intercorso

- **26 giugno** ore 9 P4
- **19 luglio** ore 9 P3
- **6 settembre** ore 9 P3

Consigli per lo studio

- **Lezioni** (attive, …..)
- Slides, internet, appunti, ma… libro è il riferimento
- **Esercizi** (da soli, in gruppi,…)
- **Ricevimento**
- Tutorato
- Organizzare gli esami dei vari corsi **in ordine**
- Non dimenticare obiettivi e motivazioni
- I concetti del corso richiedono un po' di tempo per essere assimilati, non rimanete indietro!

ETC

Elementi di Teoria della Computazione

Computazione:

Calcolare, fare il conto di qualche cosa.

Processare l'informazione mediante l'applicazione di un insieme finito di operazioni o regole

Teoria:

Formulazione logicamente coerente di un insieme di definizioni, principi e leggi generali che consente di descrivere, interpretare, classificare, spiegare, aspetti della realtà e talvolta prevedere la loro evoluzione futura

(la computazione)

Elementi:

Per fortuna sarà solo un'introduzione base ;)

Prerequisiti

Non vi sono propedeuticità formali, ma dei requisiti necessari per affrontare il corso.

- MATEMATICA DISCRETA: INSIEMI, OPERAZIONI SU INSIEMI, CARDINALITÀ, PRODOTTO CARTESIANO, FUNZIONI
- LOGICA: PREDICATI E QUANTIFICATORI. METODI E STRATEGIE DI DIMOSTRAZIONE
- ALGORITMI: PROGETTAZIONE E ANALISI

Questo è un Corso di Teoria del terzo anno.

Ci si aspetta che abbiate una certa familiarità con queste nozioni.

Invertendo l'ordine dei corsi, potreste avere difficoltà.

Alcuni concetti base verranno ripresi (prossima lezione).

Informatica teorica

- L'informatica teorica presenta dei dettagli che potrebbero renderla faticosa, ma anche molte idee **affascinanti**.
- Cercate di non perdere di vista le **idee** principali e non perdetevi nei dettagli (necessari per una corretta comprensione).
- Il libro di Sipser, per le dimostrazioni più complesse, fornisce prima IDEA, poi DIMOSTRAZIONE.
- Una preparazione tecnica specifica, oggi utile, può diventare obsoleta in pochi anni. Questo corso mira invece a favorire la capacità di pensare, di esprimersi in modo chiaro e preciso per la risoluzione dei problemi.
- Queste abilità hanno un valore durevole. Studiare la teoria vi forma in questo ambito.

Contenuto del corso (in breve)

Il corso è un'introduzione alle tre aree centrali della teoria della computazione:

- Teoria degli Automi (Linguaggi formali e modelli di calcolo)
- Teoria della Calcolabilità /Computabilità
- Teoria della Complessità

Zaccagnino (da giovedì per 12 lezioni)

Anselmo (a seguire)

Le tre aree sono legate dalla domanda:

Quali sono le capacità e i limiti dei computer?

Coloro che hanno scelto di dedicare alcuni anni della loro formazione allo studio dei computer non possono non essere curiosi circa le loro capacità e limiti.

Contenuto del corso

- MODELLI DI COMPUTAZIONE:

  AUTOMI FINITI DETERMINISTICI E NON DETERMINISTICI.
  ESPRESSIONI REGOLARI. PROPRIETÀ DI CHIUSURA DEI LINGUAGGI REGOLARI. TEOREMA DI KLEENE. PUMPING LEMMA PER I LINGUAGGI REGOLARI.

  MACCHINA DI TURING DETERMINISTICA A NASTRO SINGOLO. IL LINGUAGGIO RICONOSCIUTO DA UNA MACCHINA DI TURING. VARIANTI DI MACCHINE DI TURING E LORO EQUIVALENZA.

- IL CONCETTO DI COMPUTABILITÀ: FUNZIONI CALCOLABILI, LINGUAGGI DECIDIBILI E LINGUAGGI TURING RICONOSCIBILI. LINGUAGGI DECIDIBILI E LINGUAGGI INDECIDIBILI. IL PROBLEMA DELLA FERMATA. RIDUZIONI. TEOREMA DI RICE.

- IL CONCETTO DI COMPLESSITÀ: MISURE DI COMPLESSITÀ: COMPLESSITÀ IN TEMPO DETERMINISTICO E NON DETERMINISTICO. RELAZIONI DI COMPLESSITÀ TRA VARIANTI DI MACCHINE DI TURING. LA CLASSE P. LA CLASSE NP. RIDUCIBILITÀ IN TEMPO POLINOMIALE. DEFINIZIONE DI NP-COMPLETEZZA. RIDUZIONI POLINOMIALI. ESEMPI DI LINGUAGGI NP-COMPLETI.

Calcolabilità

Negli anni ‘30 (del secolo scorso), quando ancora l’informatica e i computer NON esistevano, quindi indipendentemente da essi, alcuni logici si proposero questo progetto:

> Formalizzare in modo esatto la nozione intuitiva di problema e di procedura effettiva di calcolo, così da definire quando un problema è risolvibile o no.

Questo progetto fu realizzato indipendentemente e con una diversa risposta da

- Alan Turing (1936) con il suo modello noto come Macchina di Turing
- Alonso Church (1936) con il λ-calcolo.

Tesi di Church - Turing

Fu dimostrato che le due risposte erano equivalenti, cioè definivano la stessa classe di problemi risolubili. Tutto ciò che poteva essere calcolato con una Macchina di Turing poteva essere calcolato (simulando la MdT) col λ-calcolo, e viceversa.

In seguito, ogni altra ‘’ragionevole" formalizzazione della nozione di procedura effettiva ha condotto agli stessi risultati.

Per questo è possibile parlare di un concetto di "calcolabilità", indipendente dal particolare formalismo.

Tesi di Church-Turing:

Tutto ciò che può essere intuitivamente calcolato tramite una procedura effettiva può essere calcolato con una Macchina di Turing.

Tesi di Church - Turing

La Tesi di Church – Turing non è un teorema, non dovrete studiarne la dimostrazione!

Anche perché non è proprio dimostrabile.

Può essere vista come definizione di procedura effettiva / algoritmo.

Questo significa anche che qualsiasi modello di «calcolatore» si possa mai progettare oggi o in futuro, con tutte le risorse e le tecnologie possibili, non potrà che risolvere i problemi risolti con una Macchina di Turing!

Per esempio, anche i ‘’quantum computer’’ obbediscono alla tesi, quindi non possono fornire vantaggi sui computer classici in termini di computabilità. Possono invece fornirli in termini di complessità di tempo.

Per esempio, per la fattorizzazione di interi in numeri primi come dimostrato da Shor nel 1994.

Algoritmo $\approx$ Macchina di Turing

Teoria della calcolabilità

La teoria della calcolabilità nasce nella prima metà del

ventesimo secolo quando alcuni logici, tra cui Goedel, Turing e Church, scoprirono l'esistenza di problemi che non possono essere risolti!

Saranno problemi strani, astratti o che non ci riguardano?

Un esempio

Progettare un algoritmo per decidere se una data affermazione matematica è Vera o Falsa.

Sarebbe bello avere un programma che faccia questo, ma non è possible: non esiste un algoritmo che possa risolvere questo problema

Esempio (terminazione)

Domanda: Dato un qualsiasi programma in C, possiamo stabilire se termina su ogni input (oppure va in loop)?

```
input n;
while (n !=1)
{
 if (n è pari)
     n := n/2;
 else
    n := 3*n+1;
}
```

Termina per ogni n>1?

Proviamo per n=17. Allora, n sarà aggiornato in 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. ok, termina!

Risposta: in generale NO!

Esempi in ambito informatico

Problemi riguardanti programmi (es. in C)

- Dato un programma in C, si arresta su ogni input, ovvero esiste un input che lo manda in loop?
- Data una sequenza di caratteri ASCII, rispetta la sintassi del C?
- Stabilire l'equivalenza di programmi: Dati due programmi essi forniscono lo stesso output?
- Individuazione dei virus: Questo programma è un virus?
- ….

Teorema di Rice

stabilisce il risultato negativo che, salvo i casi banali, non è algoritmicamente possibile decidere in generale se, dato un programma, la funzione computata verifica o meno una proprietà preassegnata (terminazione, equivalenza, …)!

Il X problema di Hilbert

Nel **1900**, D. Hilbert presentò al Secondo Congresso Internazionale di Matematica a Parigi, una lista di 23 problemi come sfida per il secolo nascente.

Il decimo problema di Hilbert era il seguente:

'' Progettare un processo in base al quale può essere determinato in un numero finito di operazioni se un polinomio a coefficienti interi ha radici intere''

Es.: $5n^2+4n-30 = 0$

Questo problema fu risolto nel 1970 da Matijasevic che - grazie a risultati precedenti di Davis, Putnam e Robinson – provò che non esiste un algoritmo che risolve tale problema.

Problemi o linguaggi?

Non parleremo di problemi e algoritmi (se non come sinonimo di Macchina di Turing) ma di linguaggi (insiemi di stringhe, le stringhe sono sequenze di simboli) e Macchine di Turing.

Esempio: il problema

‘’dato un numero x, x è primo?"

si può formulare come

‘’dato un numero x, $x \in \{ y \text{ in } N \mid y \text{ è primo} \}$?’’

Tipicamente i problemi considerati nella teoria della calcolabilità e complessità sono problemi di decisione, cioè problemi che hanno come soluzione una risposta sì o no.

Problema $\approx$ Linguaggio

risolvibile $\approx$ decidibile

Teoria della calcolabilità: argomenti trattati

- Noi introdurremo come modello di calcolo, la Macchina di Turing e alcune sue varianti.
- Prima di introdurre la Macchina di Turing, verrà introdotto un modello più semplice, l'automa a stati finiti.
- Dimostreremo l'indecidibilità di alcuni problemi, tra cui quello della fermata per le macchine di Turing.
- Per alcuni di essi, ne dimostreremo l'indecidibilità utilizzando il metodo della riducibilità mediante funzione.

Complessità

Vi sono moltissimi esempi di problemi che sono risolvibili mediante algoritmi.

Per esempio: ordinare n numeri, MST e altri studiati in PA

Però…..

.. alcuni problemi risolvibili sono (più) ‘’facili" e altri (più) ‘’difficili", a seconda che gli algoritmi finora noti per risolverli siano utili o meno in pratica.

Nel primo caso si parla di algoritmi efficienti, nel secondo caso

di algoritmi inefficienti.

Ad esempio, ordinare n numeri è facile: O(n lg n)

Calcolabilità e Complessità

Calcolabilità: si occupa di problemi risolvibili algoritmicamente in linea di principio

Domande che affronta:

Quali problemi sono risolvibili?

Cosa significa procedura effettiva di calcolo?

Complessità: si occupa di problemi risolvibili algoritmicamente in pratica?

La teoria della Complessità analizza problemi risolvibili.

Domande che affronta:

Quali sono le risorse minime necessarie (es. tempo di calcolo e memoria) per la risoluzione di un problema?

Come si misura il consumo delle risorse?

Complessità

Sono considerati efficienti gli algoritmi di complessità polinomiale.

La complessità esponenziale invece è spesso proibitiva, per dimensioni di istanze anche ragionevolmente piccole, tanto da rendere intrattabili i problemi di complessità esponenziale, come se l'algoritmo che li risolve non ci fosse.

La teoria della complessità cerca di capire le ragioni di questa complessità.

Raggruppiamo in una classe tutti i problemi che possono essere risolti con algoritmi della stessa complessità (di tempo o di spazio) per poi studiarli.

Problemi trattabili

Problema trattabile se può essere risolto da un algoritmo in tempo polinomiale.

Alla domanda:

Questo tale problema è trattabile?

1. Sì, ecco un algoritmo efficiente che lo risolve! 
2. No, ed è anche possibile dimostrarlo 
3. Nessuna delle due! 

Esempio TSP

Problema del commesso viaggiatore

(Traveling Salesman Problem, TSP)

Un commesso viaggiatore deve recarsi in alcune di città per il suo lavoro e vorrebbe visitare tutte le città una e una sola volta per poi tornare alla città di partenza minimizzando la lunghezza del percorso.

Un innocuo problema su grafi come altri studiati?

Problemi 

Ovvero Problemi NP-completi:

- Traveling Salesman Problem
- Problema del ciclo Hamiltoniano
- Vertex - Cover
- 3-coloring di grafi
- SAT, soddisfacibilità di formule booleane
- Scheduling Multiprocessore
- Folding Proteine,
- Programmazione lineare intera
- …..

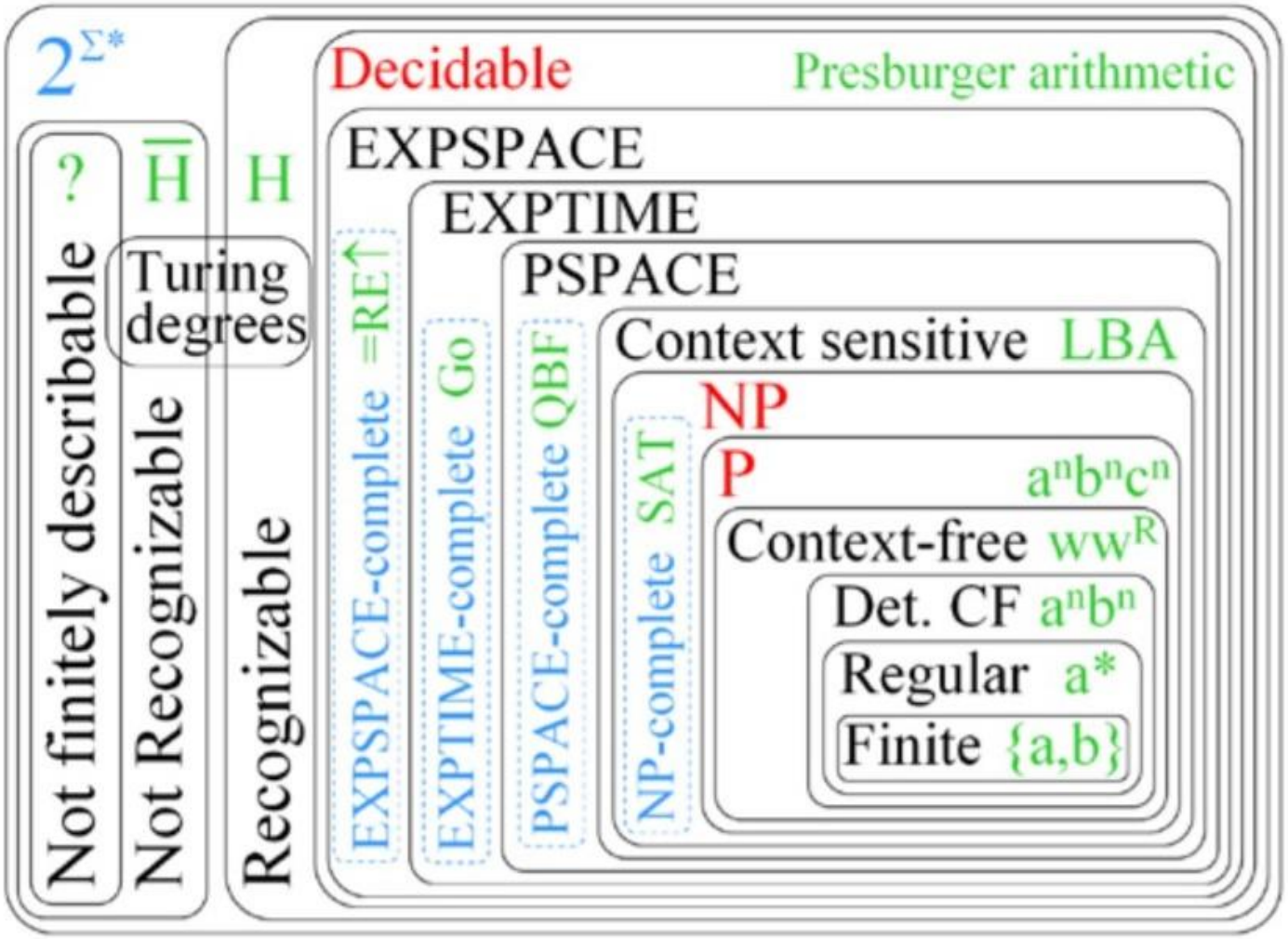
Tutti risolvibili efficientemente o nessuno!

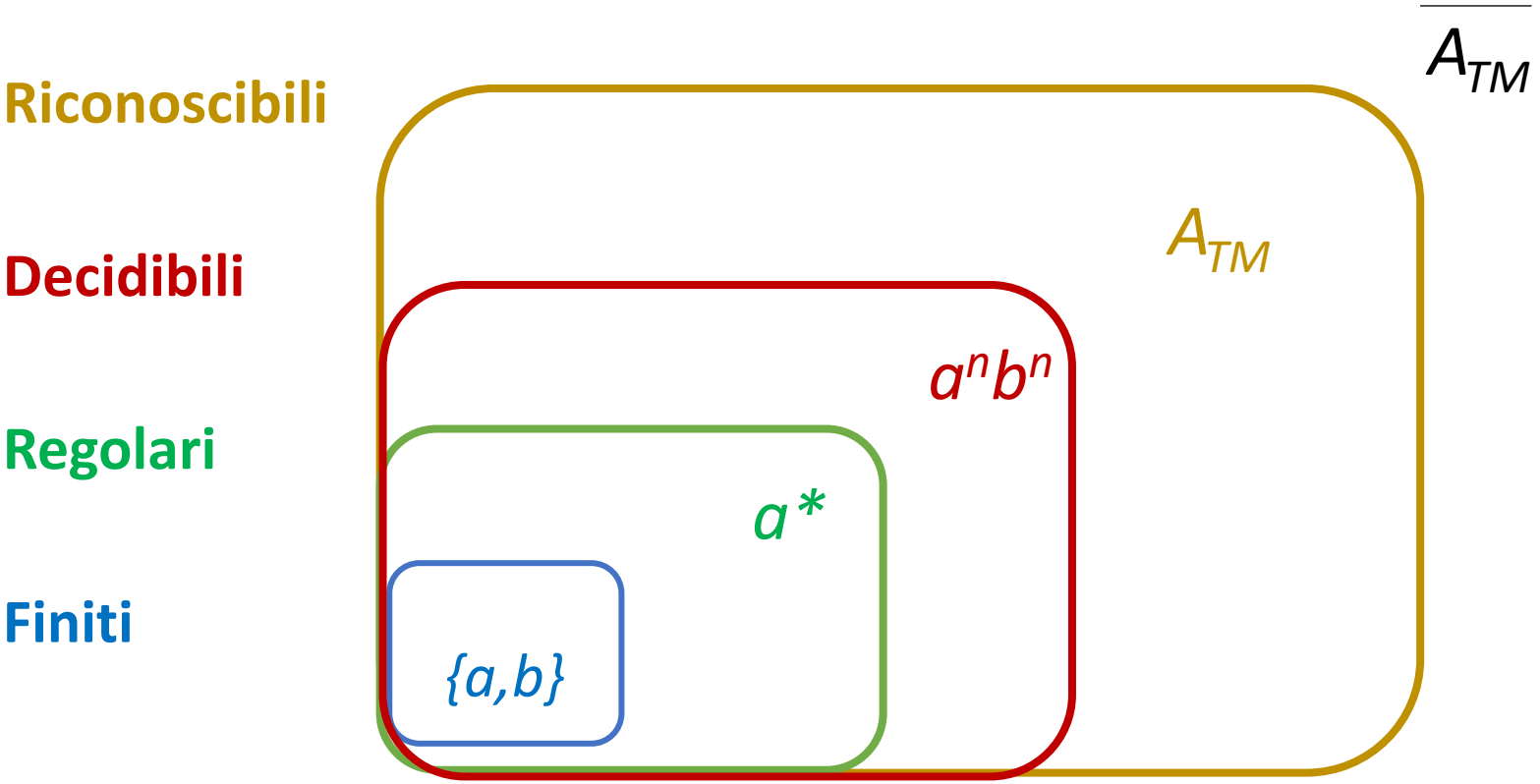
Complessità

- Studieremo le classi di complessità più note.
- La classe P, che corrisponde alla classe dei problemi risolubili con un algoritmo di complessità di tempo polinomiale e la classe NP.
- Introdurremo il concetto di riduzione polinomiale tra linguaggi/problemi come strumento per dimostrare l'appartenenza o meno a una classe di complessità.
- Esporremo uno dei più grandi problemi aperti dell'informatica teorica: il limite fra problemi trattabili e intrattabili non è chiaro

$$P = NP?$$

The Extended Chomsky Hierarchy




Riconoscibili
Decidibili
Regolari
Finiti
$\overline{A_{TM}}$
A_{TM}
$a^n b^n$
$a*$
$\{a,b\}$

Algoritmo $\approx$ Macchina di Turing

Problema $\approx$ Linguaggio

risolvibile $\approx$ decidibile

Calcolabilità e Complessità

Calcolabilità: quali problemi è possibile risolvere algoritmicamente in linea di principio?

La Macchina di Turing e sue potenzialità.

Complessità: quali problemi è possibile risolvere algoritmicamente in pratica?

La teoria della Complessità analizza problemi risolvibili e li classifica secondo le risorse necessari per risolverli.

E la Teoria degli Automi?

Teoria degli Automi

Studierete gli automi finiti come primo semplice modello di computazione.

Verranno poi arricchiti di altre capacità e risorse fino ad arrivare alle Macchine di Turing.

La teoria si sviluppa negli anni 50-60 e alcuni risultati teorici hanno avuto un grandissimo impatto sullo sviluppo software.

Gli automi finiti sono dei modelli di calcolo a memoria finita; hanno cioè un certo numero finito di stati di memoria in cui si possono trovare. Processano il dato in ingresso sequenzialmente in base allo stato in cui si trovano, fino ad arrivare agli stati finali, di accettazione o rifiuto.

Lo studio di questo semplice modello aiuterà a comprendere l'altro modello più complesso che studieremo, la Macchina di Turing.

Comincerete da giovedì col prof. Zaccagnino

Prime lezioni

- Oggi: Introduzione al corso
- Giovedì 2 : Ripasso concetti e notazioni di matematica. Automi finiti.

نظری‌ه زبان ها و ماشی‌ن ها

فصل صفر- مقدمه

دانشگ‌اه صنعتی شری‌ف

نی‌مسال دم سال تحصی‌لی 1386

مراجع درس

- مرجع اصلی:

M. Sipser, "Introduction to the Theory of Computation," 2nd Ed., Thompson Learning Inc., 2006.

- مراجع کمکی:

P. Linz, "An Introduction to Formal Languages and Automata," 3rd Ed., Jones and Barlett Publishers, Inc., 2001.

J.E. Hopcroft, R. Motwani and J.D. Ullman, "Introduction to Automata Theory, Languages, and Computation," 2nd Ed., Addison-Wesley, 2001.

P.J. Denning, J.B. Dennnis, and J.E. Qualitz, "Machines, Languages, and Computation," Prentice-Hall, Inc., 1978.

ادامه

- اثبات : دنباله اي است از گـزاره هاي $p_1, p_2, \ldots, p_k$ به گـونه اي كـه p_1 بايد يكـي از اصول و p_2 بايد يكـي از اصول باشد يا با استفاده از p_1 و يكـي از قواعد استنتاج اثبات مي شود و p_3 هم ... كـه به اين مراحل يک اثبات براي p_k گـفته مي شود.

- قضيه (theorem) : گـزاره اي است كـه در منطق، براي آن اثباتـي وجود داشته باشد.

- منطق غير كـامل : فرمول هايي وجود دارد كـه نمي توان آنها را اثبات كـرد.