Introduzione ai Linguaggi formali

Cosa faremo

STUDIEREMO DA UN PUNTO DI VISTA MATEMATICO I LINGUAGGI E LE GRAMMATICHE

Perché studiare le grammatiche?

I LINGUAGGI DI PROGRAMMAZIONE SONO LINGUAGGI ARTIFICIALI E I PROGRAMMI PRIMA DELL'ESECUZIONE SONO TRADOTTI IN LINGUAGGIO MACCHINA (CHE E' A SUA VOLTA UN LINGUAGGIO ARTIFICIALE)

LA TRADUZIONE È FATTA DA UN PROGRAMMA

I LINGUAGGI DI PROGRAMMAZIONE SONO LINGUAGGI ARTIFICIALI

COME POSSIAMO FORMALIZZARE LE DEFINIZIONI DI UNA LINGUA ARTIFICIALE UTILE IN AMBITO INFORMATICO?

POSSIAMO TROVARE RAPPRESENTAZIONI CHE SIANO INTERPRETABILI MATEMATICAMENTE IN MODO DA PERMETTERCI

1. *DI RAGIONARE FORMALMENTE SU DI ESSE E*
2. *PROGETTARE PROGRAMMI TRADUTTORI DA UN LINGUAGGIO ARTIFICIALE NEL LINGUAGGIO MACCHINA DI UN CALCOLATORE ELETTRONICO?*

Se usiamo linguaggi artificiali dobbiamo anche progettare traduttori

Dobbiamo tradurre programmi scritti in un linguaggio di programmazione ad alto livello in un linguaggio direttamente eseguibile da un calcolatore.

Questo richiede algoritmi che siano in grado di tradurre in modo automatico ed efficiente.

Due questioni rilevanti:

- Quale formalismo per definire linguaggi è abbastanza espressivo e risulta abbastanza semplice per realizzare un programma di traduzione automatico?
- La traduzione automatica di un programma da parte di un elaboratore richiede che la definizione del linguaggio sia non ambigua. Come possiamo essere sicuri che il formalismo che utilizziamo non produca mai ambiguità?

Sintassi e semantica

La Sintassi specifica le regole secondo le quali una frase è corretta o meno nella lingua: una frase è sintatticamente corretta se è ottenuta applicando delle specifiche regole

Esempio, in Italiano abbiamo

- IL TOPO MANGIA IL FORMAGGIO : OK
- IL TOPO IL GATTO IL FORMAGGIO : Scorretta

Sintassi e semantica

Semantica: la sintassi non dice nulla circa il significato di una frase; questo è il compito della semantica.

Esistono frasi sintatticamente corrette che non hanno alcun significato. Esempio

Idee verdi incolori dormono furiosamente

Nota:

- Quando parliamo usiamo spesso frasi scorrette, e l’ascoltatore di solito riesce a ‘determinare’ la sintassi e la semantica sottostanti e capire il significato
- Questo è particolarmente evidente nei discorsi di un bambino o in alcune poesie.

Sintassi e semantica

Due questioni rilevanti

- Quale formalismo per definire linguaggi è sufficientemente espressivo e allo stesso tempo sia semplice e permetta di realizzare un programma di traduzione automatico veloce?

- La traduzione automatica di un programma da parte di un elaboratore richiede che la definizione del linguaggio sia non ambigua. Come possiamo essere sicuri che il formalismo che utilizziamo non produca mai ambiguità?

Cosa faremo (in questa parte del corso)

Ci concentriamo sulla Sintassi dei linguaggi con particolare attenzione ai linguaggi di programmazione

Vedremo

1. UNA DEFINIZIONE FORMALE (MATEMATICA) DI UN LINGUAGGIO USANDO

2. DIVERSI APPROCCI PER DEFINIRE LINGUAGGI

Nota: fortunatamente i linguaggi di programmazione sono linguaggi molto più semplici dei linguaggi naturali (es. italiano, inglese)

Cosa faremo (in questa parte del corso)

2. DIVERSI APPROCCI PER DEFINIRE LINGUAGGI

- Approccio algebrico che utilizza opportune operazioni su stringhe di caratteri (visto a Fondamenti I con espressioni regolari)
- Definizione tramite opportuna formalizzazione del concetto di grammatica

3. DEFINIZIONE DI ALGORITMI CHE DECIDONO SE UNA STRINGA APPARTIENE O MENO AL LINGUAGGIO

- Questa definizione è un passo preliminare per la traduzione in linguaggio macchina)

Cosa faremo (in questa parte del corso)

Trattare tutti i punti precedenti in modo approfondito richiede un corso intero (almeno)

Nel seguito ci concentriamo su

- Definizione di grammatica formale
- Ambiguità nei linguaggi
- Algoritmi per riconoscere opportune classi di linguaggi (di interesse in programmazione)

Indice (oggi)

1. Introduzione ai linguaggi formali e alle stringhe
2. Problemi di decisione sui linguaggi
3. Classificazione di Chomsky delle grammatiche

Insiemi finiti e infiniti

- Gli insiemi possono essere finiti o infiniti
  - Infinito numerabile : può essere messo in uno-a –uno con i numeri naturali (ad esempio , numeri razionali , numeri interi)
  - Infinito non numerabile: non può essere messo in uno-a- uno corrispondenza con i numeri naturali (un esempio di infinito non numerabile: i numeri reali)

Tipi di dati e stringhe

- Nella vita reale , usiamo diversi tipi di dati : numeri interi , reali , vettori , numeri complessi, grafici , programmi
- Tutti i possibili dati possono essere codificati come stringhe
- Così possiamo restringerci ad un solo tipo di dati : le stringhe

Stringhe

- Un alfabeto è un insieme finito di simboli distinti
  - $\{0 , 1\}$, $\{0 , 1,2, . . . , 9\}$, $\{a, b , c\}$
  - Indichiamo un alfabeto generico con Σ
- Una stringa è una sequenza di lunghezza finita di elementi di Σ
  - esempio , se $\Sigma = \{a , b\}$ allora aba , aaaa , ... , ababababab sono alcune stringhe sull'alfabeto Σ

Stringhe

- La lunghezza di una stringa S è il numero di simboli nella stringa che indichiamo con $|S|$
  - ad esempio se $S = aba$, allora $|S| = 3$
- Il simbolo ε indica la stringa vuota di lunghezza 0
- La concatenazione di stringhe
  - Se $S = a_1a_2a_3 \ldots a_n$ e $T = b_1b_2b \ldots b_m$ allora $ST = a_1a_2a_3 \ldots a_n b_1b_2b \ldots b_m$
  - La concatenazione è associativa con ε come elemento di identità (infatti $S\varepsilon = \varepsilon S = S$)
- a^n indica una stringa di n a concatenate
  - Esempio se $\Sigma = \{0, 1\}$, allora $0^5 = 00000$
    inoltre per definizione abbiamo : $a^0 = \varepsilon$ e $a^{n+1} = a^n a$

Stringhe

- L' inversione di una stringa S è indicata con S^R
  - Se $S = a_1 a_2 \dots a_n$ allora $S^R = a_n a_{n-1} \dots a_1$
- Una sottostringa di una stringa S è una stringa y t.c. $S= xyz$ con $|x|,|y|,|z| \geq 0$ e $|x|+|y|+|z|= |S|$
- Se $S= xy$ con $|x|, |y| \geq 0$ e $|x| + |y| =|S|$, allora x è prefisso di S e y è un suffisso di S
  - Dato S= abaab ,
    - ε, a, aba , e abaab sono alcuni prefissi
    - ε, abaab , aab , e baab sono alcuni suffissi
- Se x è una stringa , scriviamo x^n per la stringa ottenuta concatenando n copie di x .

  $(aab)^3 = aabaabaab$

Stringhe

- L' insieme di tutte le possibili stringhe sull' alfabeto Σ è denotato con Σ^*
- Definiamo $\Sigma^0 = \{\varepsilon\}$ e $\Sigma^n = \Sigma^{n-1}\Sigma$

  - simbolo ε indica la stringa vuota di lunghezza 0

  - con qualche abuso della notazione la concatenazione è applicata insiemi di stringhe
- Σ^n rappresenta le stringhe di n caratteri
  - $\Sigma^n = \{ S \mid S = xy \text{ e } x \in \Sigma^{n-1} \text{ e } y \in \Sigma \}$

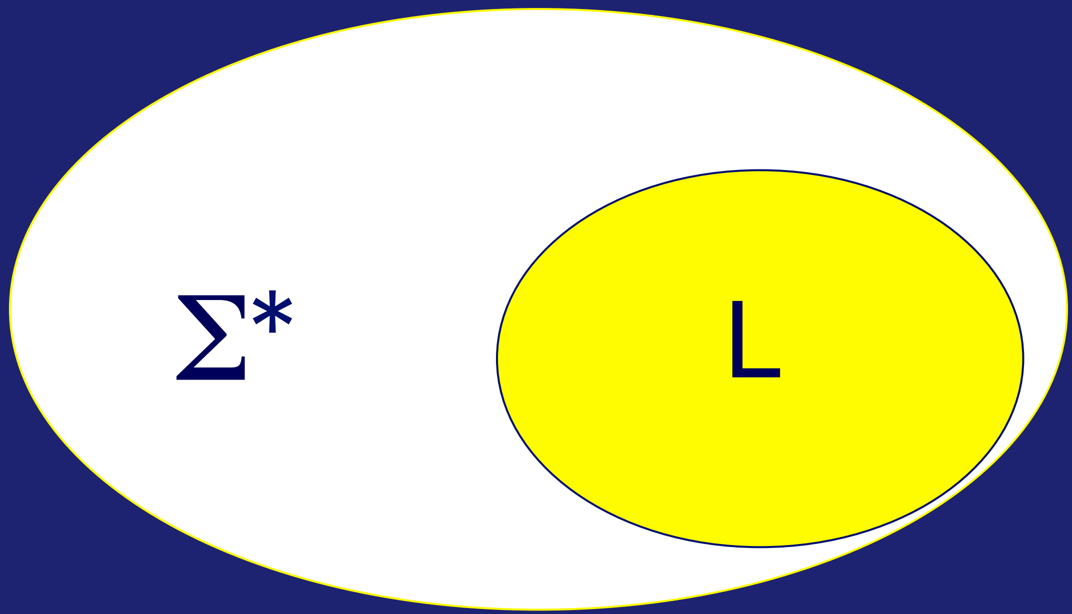
Cosa è un linguaggio?

Dato un alfabeto Σ consideriamo Σ^*

- Un linguaggio L è un sottoinsieme di Σ^* (alcune stringhe Σ^* di appartengono al linguaggio altre no)
- Definire un linguaggio richiede di specificare le regole per manipolare gli elementi dell'alfabeto e generare le sequenze di caratteri dell'alfabeto che appartengono al linguaggio

Sia dato un alfabeto Σ (finito)
- Σ* (infinito) denota tutte le stringhe ottenibili da Σ (inclusa la stringa vuota)

- dato Σ un linguaggio L su Σ e' un sottoinsieme di Σ*



Es.: I programmi python sono un sottoinsieme delle stringhe che si possono scrivere

Esempi di linguaggi

- $L = \Sigma^*$ - La madre di tutte i linguaggi (comprende tutte le stringhe)!
- $L = \{a, ab, aab\}$ - Un linguaggio finito (Descrizione per enumerazione)
- $L = \{a^n b^n : n \geq 0\} = \{\varepsilon , ab , aabb , aaabbb ....\}$
- $L = \{ S \mid S = S^R\}$ - Tutte le stringhe palindromi (che sono uguali alla loro inversione)
- $L = \{ S \mid S = xx\}$ - Tutte le stringhe formate da duplicare qualche stringa una volta
- L = {S | S è un programma Python sintatticamente corretto}

Sintassi dei linguaggi di programmazione

- Per i linguaggi di programmazione Σ = ASCII dove ASCII è l'insieme dei caratteri alfanumerici cioè tutti i caratteri presenti sulla tastiera dei computer (standard internazionale)
- Quindi ogni linguaggio di programmazione L abbiamo che $L \subseteq ASCII^*$
- Un programma è una frase di L quindi un programma è una sequenza di caratteri alfanumerici

Sintassi dei linguaggi di programmazione

Nota La definizione di linguaggio come sottoinsieme è una definizione sintattica

Esempi:

- I programmi python (C, Java…) sintatticamente corretti
- Le frasi italiane sintatticamente corrette

Pertanto permette di definire se un programma ècorretto sintatticamentema non aiuta nel sapere se è corretto. In particolare non aiuta a stabilire se il programma cicla o se presenta errori rilevabili a tempo di esecuzione

•

Sintassi e Semantica dei linguaggi

Un linguaggio formale è l'insieme delle stringhe che si possono scrivere utilizzando le regole del linguaggio

- Analisi sintattica: verificare se una stringa di simboli verifica le regole
  - es. "Oggi piove", è sintatticamente corretta; "noi vado a casa", non è sintatticamente corretta
- Analisi semantica: verificare se una stringa sintatticamente corretta ha un significato
  - es. "oggi piove" è semanticamente corretta,"io peso meno di 50 chili e più di 80 chili" è sintatticamente corretta ma non ha una semantica corretta.

Operazioni su linguaggi

- Poiché i linguaggi sono insiemi, tutte le operazioni insiemistiche abituali come intersezione e unione, ecc. sono definite
- Complementazione è definita rispetto all' universo

  Dato L definito su alfabeto Σ : $L^C = \Sigma^* - L$
- Esempi: Se L , L1 e L2 sono linguaggi :
  - $L1 \cup L2$ e $L1 \cap L2$ (unione e intersezione)
  - $L1 \cdot L2$ = {xy | x appartiene a L1 e y appartiene a L2}

    - concatenazione di linguaggi
  - Rappresentiamo $L \cdot L$ come L^2 e $L^{i+1} = L \cdot L^i$
  - Iterazione $L^* = \{L^i \mid i \geq 0\}$

Descrizione di linguaggi

- I Linguaggi interessanti sono infiniti
- Abbiamo bisogno di descrizioni finite per definire insiemi infiniti
  - esempio $L = \{a^n b^n : n \leq 0\}$, decrive il linguaggio infinito di a seguite da un pari numero di b
- Dobbiamo essere in grado di utilizzare queste descrizioni in procedure meccanizzabile (programmi)

Problemi di decisione

Un problema di decisione è una funzione il cui valore è SI / NO

Abbiamo bisogno di specificare

- l'insieme A di possibili ingressi (di solito A è infinito)
- il sottoinsieme B di A delle istanze SI (solitamente B è anche infinito)
- Il sottoinsieme B dovrebbe avere una descrizione finita

Esempi, A: numeri interi

- x è pari?
- X è un numero primo?

Problemi su linguaggi

I problemi di interesse sui linguaggi sono problemi di decisione

Esempi

- Data una stringa x, x rappresenta un programma Python corretto?
- Data una stringa x di caratteri alfabetici è una parola palindroma?
- Data una stringa formata da caratteri “a” e “b” il numero di “a” e “b” sono uguali?

Una MdT *riconosce* un linguaggio se e solo se **accetta tutte** le stringhe che appartengono al linguaggio.

Un linguaggio e' **semi-decidibile** se e solo se esiste una MdT che con input x **riconosce** se x **appartiene** a L.

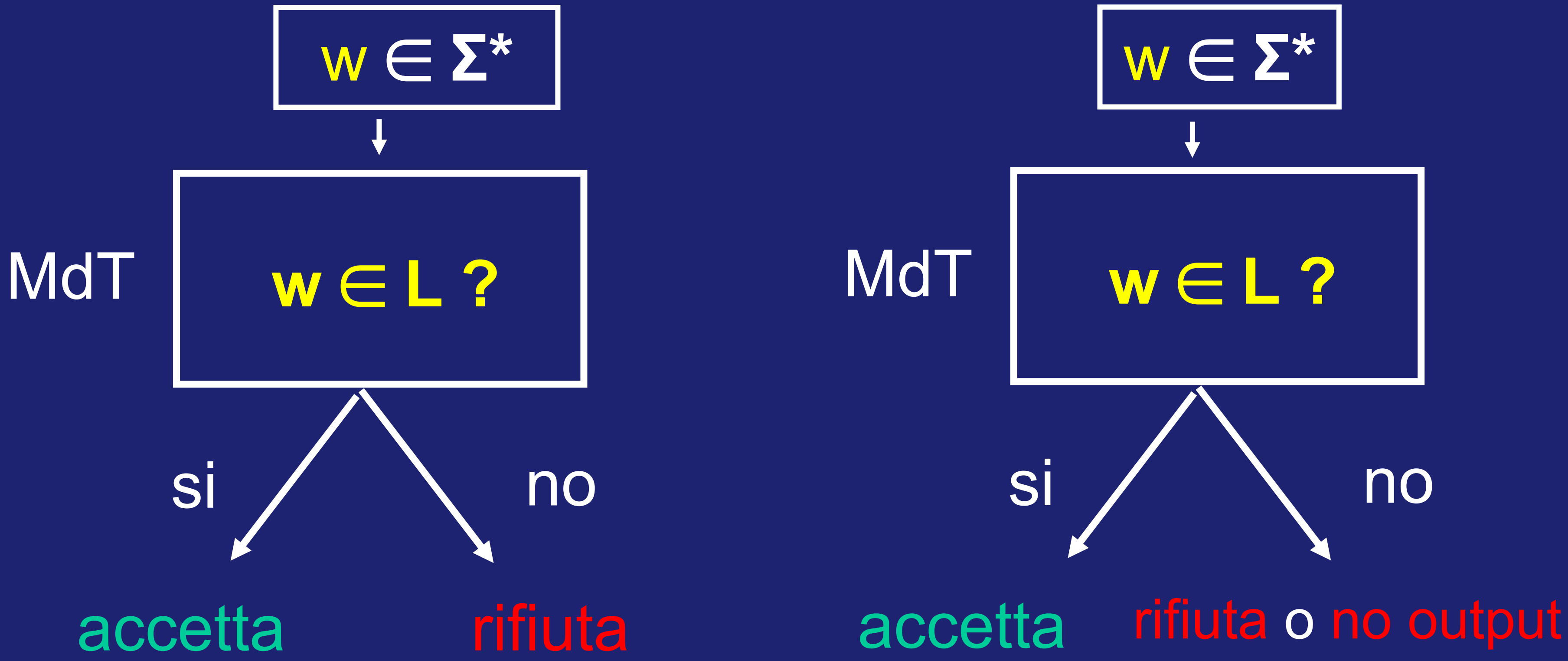
Una MdT *decide* un linguaggio L se e solo se i) **accetta** tutte le stringhe in L **E** ii) **rifiuta** tutte le stringhe non in L.

Un linguaggio L e' **decidibile** (o **ricorsivo**) **se e solo se** esiste una MdT che con input x **rico-nosce** se x **appartiene o non appartiene** a L

Poiché assumiamo che la tesi di Church-Turing sia vera le seguenti definizioni sono equivalenti alle precedenti

Un linguaggio e' **semi-decidibile** se e solo se esiste una ~~MdT~~ **un programma (un algoritmo)** che con input x **riconosce** se x **appartiene** a L.

Un linguaggio L e' **decidibile** (o **ricorsivo**) **se e solo se** esiste ~~una MdT~~ **un programma (un algoritmo)** che con input x **riconosce** se x **appartiene o non appartiene** a L



L e' decidibile

L e' semi-decidibile
(Turing-riconoscibile)

Teorema: L è decidibile se sia L e ¬L (complemento di L) sono semi-decidibili

$\neg$L denota il linguaggio complemento di L

Teorema: Se L e $\neg$L sono semidecidibili allora L è decidibile

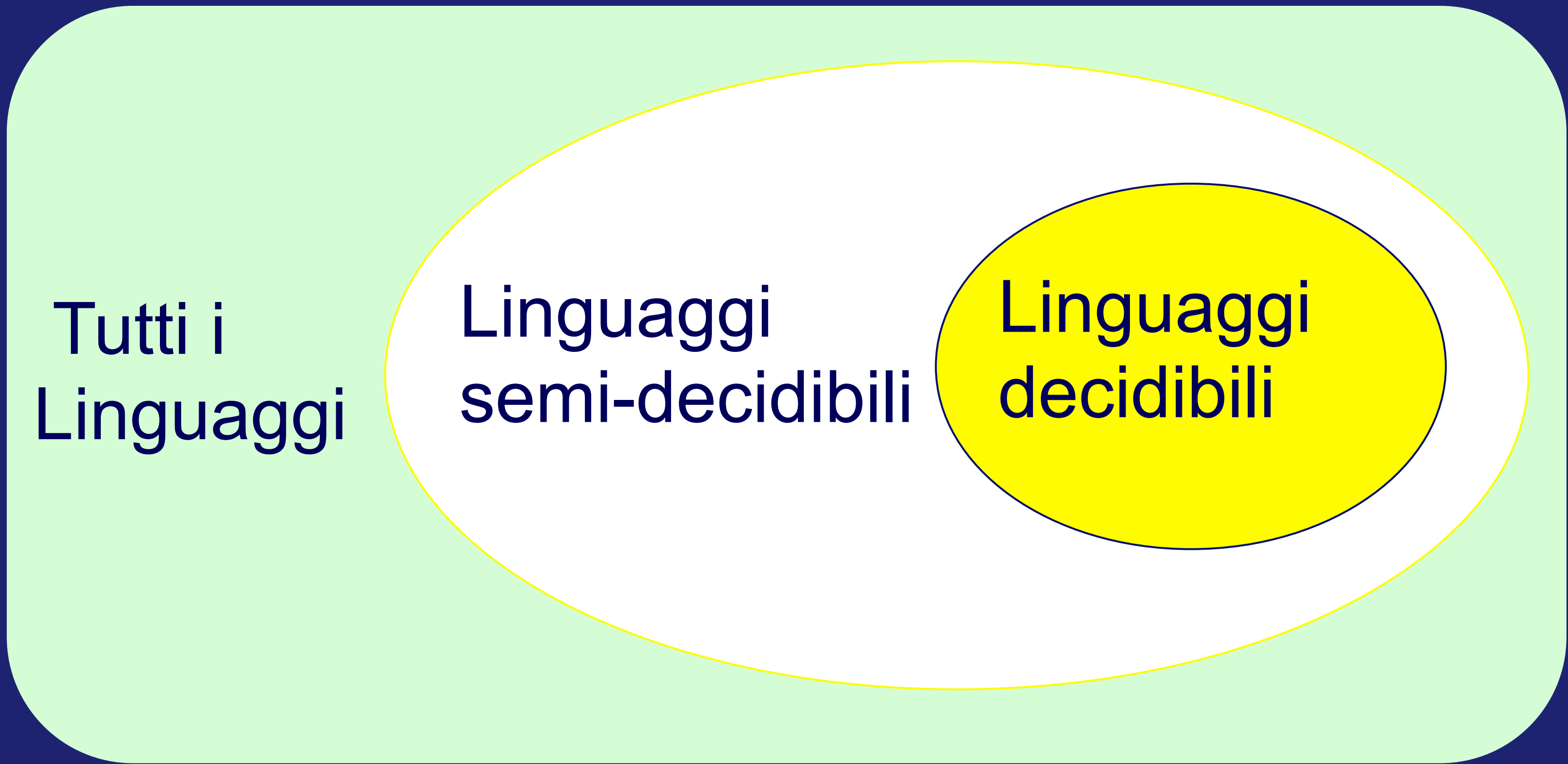
Siano TM_A e TM_R due MdT che semidecidono L e $\neg$L rispettivamente

Costruiamo una nuova MdT che decide L

- Esegui TM_A e TM_R in parallelo (un passo ciascuna)
  Una fra le due eventualemnte riconosce la stringa
- Se TM_A riconosce allora accetta
- Se TM_R riconosce allora rifiuta

Un linguaggio e' semi-decidibile se e solo se esiste una MdT che riconosce L.

Un linguaggio L e' decidibile se e solo se esiste una MdT che decide L.



Ci sono linguaggi definiti su {0,1} che non sono decidibili

Assumendo la tesi di Church-Turing allora ci sono problemi che i linguaggi non sono decidibili

Abbiamo visto (video) la prova usando un argomento di cardinalità (conteggio)
Abbiamo visto (video) la prova per il problema della fermata

Ci sono linguaggi definiti su {0,1} che non sono decidibili

Abbiamo visto (video) la prova usando un argomento di cardinalità (conteggio)

Infatti (vedi anche dispense) abbiamo visto la dimostrazione per i problemi di decisione
La prova (Teorema 8- dispense) considera problemi del tipo "*decidere se per ogni insieme S di numeri naturali esiste un programma P_S che decide S*"

Input di P_S *è un intero* x *e con input* x *risponde sì o no a seconda che* x *appartenga o meno a* P_S

Ci sono linguaggi definiti su {0,1} che non sono decidibili

La prova (Teorema 8- dispense) considera problemi del tipo "*decidere se per ogni insieme S di numeri naturali esiste un programma P_S che decide S*"

Input di P_S *è un intero* x *e con input* x *risponde sì o no a seconda che* x *appartenga o meno a* P_S

Esempi

- se S è insieme dei numeri pari : facile
- se S è insieme dei numeri primi: facile

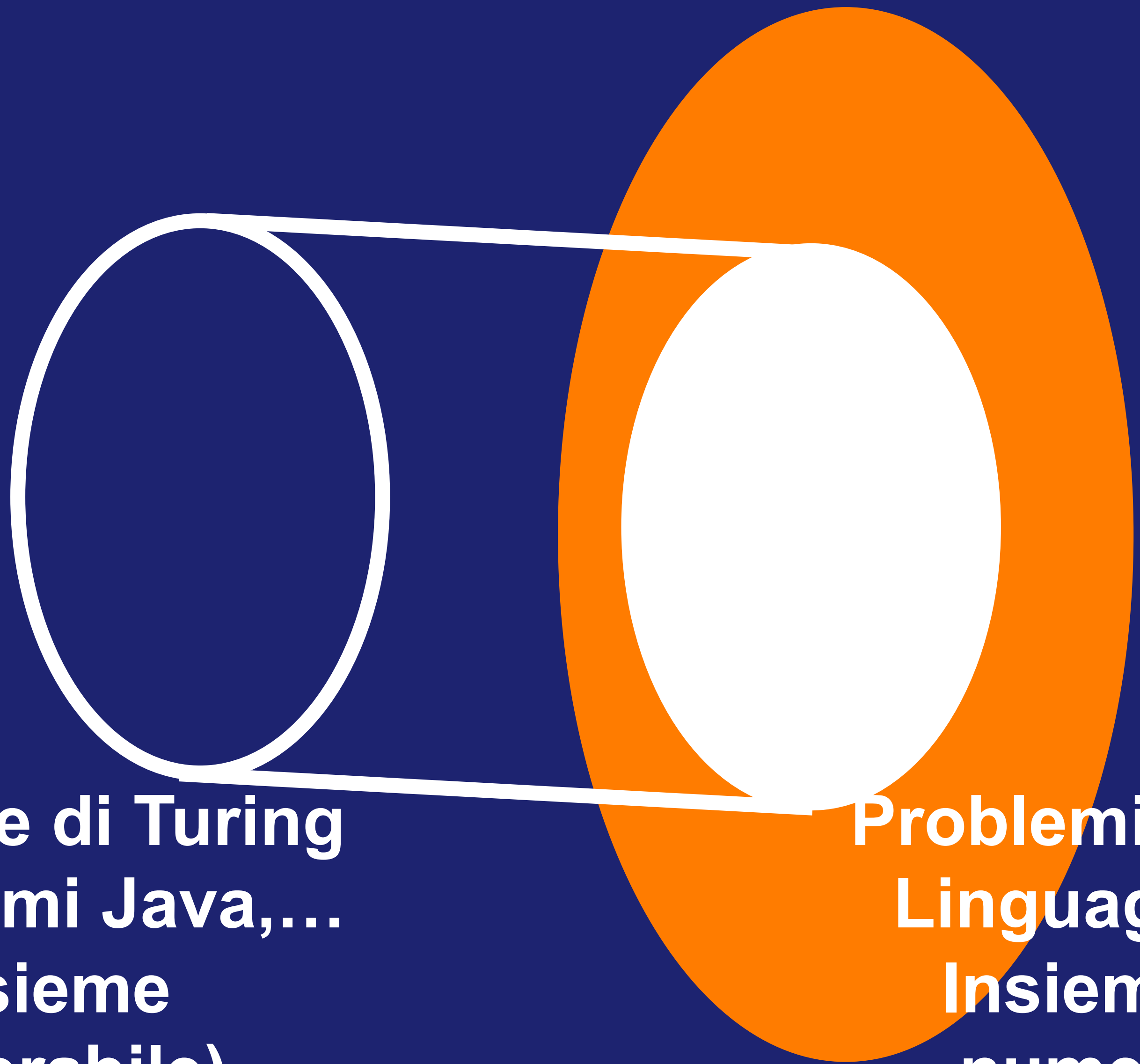
La domanda che ci poniamo è se un tale programma esiste per ogni S

Ci sono linguaggi definiti su {0,1} che non sono decidibili

La prova (Teorema 8- dispense) considera problemi del tipo "*decidere se per ogni insieme S di numeri naturali esiste un programma P_S che decide S*"

Il problema è equivalente al seguente problema su linguaggi: considerato l'alfabeto delle cifre 0-9 sia L_S il linguaggio composto dalle sequenze di cifre (stringhe) che se interpretate come numeri appartengono a S
Esempio;
Σ={0,1,2,3,4,5,6,7,8,9},
Σ^*=(tutte le possibili sequenze di cifre}
All'insieme dei numeri pari corrisponde il linguaggio che contiene le sequenze di cifre L_S={2,4,6,..,10,12,…}

Prova basata sulla cardinalità



Macchine di Turing
Programmi Java,…
(Insieme numerabile)

Problemi su interi
Linguaggi {0,1}
Insieme non numerabile

Teorema di Cantor

Sia L un insieme e 2^L l'insieme delle parti di L

Teorema: Non esiste una corrispondenza biunivoca fra L e 2^L

Prova: Assumi per contraddizione che ci sia una corrispondenza biunivoca $f : L \rightarrow 2^L$

$$\textbf{Sia } S = \{ x \in L \mid x \notin f(x) \}$$

Abbiamo costruito *S* in modo tale che, per ogni *x* in *L*, l'insieme *S* differisce da f(x):

$S \neq f(x)$ perché $x \in S$ se e solo se $x \notin f(x)$

Abbiamo dimostrato che

Per ogni insieme L (finito o infinito), l'insieme delle parti 2^L *ha sempre cardinalità maggiore di L*

Il teorema precedente implica che

L'insieme di tutti i linguaggi su un alfabeto Σ ha cardinalità superiore a quella dell'insieme Σ^*

Concludiamo

Esistono linguaggi non decidibili

Non tutti i linguaggi su {0,1} sono decidibili, infatti: non tutti i linguaggi su {0,1} sono semi-decidibili

{Macchine di Turing}

{Stringhe di 0 e1}

I due insiemi hanno
la stessa cardinalità
Infatti posso codificare
una MdT con una
stringa binaria

Insieme L

Insiemi numerabili

{Tutti i Linguaggi su {0,1}}

{Insieme delle stringhe
su 0,1}

I due insiemi hanno
la stessa cardinalità
Infatti un linguaggio su
{0,1} è un insieme di
stringhe su {0,1}

Insieme delle parti: 2^L

Insiemi non numerabili

Il PROBLEMA DELLA FERMATA

Halting Problem

$HALT_{TM}$ ={(M,w) | M è MdT che si ferma con input w }

$HALT_{TM}$ è un problema di decisione

Definiamo il linguaggio L-HALT che rappresenta il problema della fermata

Usiamo come alfabeto $\Sigma = \{0,1,\#\}$

limitiamo la nostra attenzione a MdT con alfabeto {0,1}

1. possiamo codificare le macchine di Turing con una stringa binaria
2. L-HALT={stringhe del tipo x#y | x è la codifica di una MdT M(x), y codifica un input a M(x) e M(x) si ferma con input y}

Il PROBLEMA DELLA FERMATA

Halting Problem

$HALT_{TM}$ ={(M,w) | M è MdT che si ferma con input w }

Teorema: $HALT_{TM}$ non è decidibile

Prova: Assumi per contraddizione che esista MdT H che decide $HALT_{TM}$

Usiamo H per costruire una MdT D che con input (M,w) esegue H con input (M,w)

Se H rifiuta allora termina e rifiuta

Se H accetta , allora cicla (non termina)

Ma allora H(D,w) che fa?

- Se H(D,w) termina allora D non termina
- Se H(D,w) non termina allora D termina

ESERCIZIO

Abbiamo visto che

- L-HALT non è decidibile (non sappiamo decidere se x#y appartiene a L-HALT)
- L-HALT è semi-decidibile: se eseguiamo la MdT x con input y e MdT x si ferma allora x#y appartiene a L-HALT; (Si noti che il problema è che non sappiamo decidere quando x cicla; dopo 1 giorno, un anno, un secolo?)

Consideriamo il linguaggio ¬L-HALT (il linguaggio complemento di L-HALT)

Le stringhe in ¬L-HALT sono le stringhe x#y per cui la MdT x cicla con input y

Domanda: ¬L-HALT è decidibile? è semidecidibile?

ESERCIZIO

Le stringhe in ¬L-HALT sono le stringhe x#y per cui la MdT x cicla con input y

Teorema ¬L-HALT NON è semidecidibile

Prova Sappiamo che

1) HALT è semidecidibile
2) HALT NON è decidibile

Abbiamo visto (in un lucido precedente) che

3) Se L e ¬L sono semidecidibili allora L è decidibile

Quindi se ¬L-HALT fosse semidecidibile 1) e 3) implicano che HALT è decidibile contraddicendo 2

Quindi deduciamo che

¬L-HALT NON è semidecidibile