Linguaggi e Computabilità

DaveRhapsody

2 Ottobre 2019

Indice

L’esame

Avremo due compitini, uno a novembre ed uno a Gennaio, in un anno sono disponibili 5 appelli, se uno è del terzo anno, può fare i compitini, basta che ci sia spazio nelle aule, la precedenza va a coloro che sono del secondo anno.

Al secondo appello (Quello di Febbraio) puoi recuperare il voto negativo di uno dei due compitini. Non presentarsi è esattamente come provarci e non passare, quindi rischiate, conviene.

L’orale va sostenuto nello stesso appello dello scritto, cioè io faccio lo scritto, lo passo, l’orale lo devo fare in quella sessione. Per chi fa i compitini ed ha consegnato anche gli esercizi di lab. può fare un orale prima del 5 Febbraio OPPURE si può fare assieme a coloro che hanno fatto l’esame il 5.

Gli esercizi valgono dal momento che li invii fino a fine anno, quindi ha senso farli subito tutti

---

Capitolo 1

Linguaggi formali

Nascono per essere in grado di creare i linguaggi di programmazione, o meglio servono per gestire i protocolli di comunicazione e la possibilità di comunicare una determinata operazione al calcolatore.

1.1 Backus-naur form (Backus Normal Form)

Definizione Da Wikipedia: è una metasintassi, ovvero un formalismo attraverso cui è possibile descrivere la sintassi di linguaggi formali (il prefisso meta ha proprio a che vedere con la natura circolare di questa definizione). Si tratta di uno strumento molto usato per descrivere in modo preciso e non ambiguo la sintassi dei linguaggi di programmazione, dei protocolli di rete e così via, benché non manchino in letteratura esempi di sue applicazioni a contesti anche non informatici e addirittura non tecnologici. La BNF viene usata nella maggior parte dei testi sulla teoria dei linguaggi di programmazione e in molti testi introduttivi su specifici linguaggi.

1.2 Model checking

Usato per protocolli di comunicazione, per esempio per protocolli di pagamento, in realtà di qualsiasi tipo, chiaramente per la sicurezza questo è l'ideale, perchè si descrive lo stato di sistema, e si specifica se ogni stato è sicuro (Sicuro sia dal punto di vista dei risultati corretti che sicuri)

E' usato anche per il software, cioè in maniera automatica deduce in base alle condizioni di ingresso, se son corrette. Ce la fa? Si per programmi piccini, ma alla fine, ma ingenerale, non esiste una tecnica che preso un software ti dimostra che esso sia corretto in ogni caso. Non esiste nessuna procedura generale, se esistesse ci sarebbero contraddizioni logiche.

Cos'è una contraddizione logica? E' un paradosso, ma a livello un po' più infame, pensate alla frase "Questa frase è vera", se ci scavate a fondo, dopo un po' diventa una contraddizione.

1.3 Automi a stati finiti

Sono insiemi di stati ai quali arrivan dall'esterno dei dati, ed a seconda dello stato in cui si trovano, e del dato che arriva, allora potrebbero verificarsi le famose "Transizioni" che consistono nel cambiare stato.

La memoria del Latch SR, ad esempio, funziona come un automa, nel senso, varia a seconda dello stato interno, e del valore di ingresso.

Linguaggio Per E' uno dei primi linguaggi di scripting, anche se ce n'era qualcun altro prima, e contiene istruzioni per gestire espressioni regolari che possono essere applicate su testi lunghi per fare ricerche.
In pratia prendevano delle sequenze di DNA (tera di dati), e venivano analizzati (con espressioni regolari) da questo linguaggio.

Capitolo 2

Alfabeto

E' un insieme finito e non vuoto di simboli, ad esempio: A, B, C, D, ... , Z , 1, 2, 3, 4, ..., 9.
Per gli alfabeti useremo lettere greche tipo: Σ, Λ, Γ, vediamo alcune definizioni ora:

Stringa La stringa è una sequenza di simboli, se è vuota si definisce vuota, può esistere. Data una stringa w, si indica la sua lunghezza con |w|. Per esempio: |acdas234| = 8, mentre se ho $|\epsilon| = 0$, poichè si indica che una stringa è vuota dicendo che essa abbia solo una lettera greca dentro

Concatenazione tra stringhe La concatenazione fa in modo che date due stringhe w, x l'ultimo carattere di x sarà il successivo dell'ultimo di w. pertanto, w, x $\rightarrow$ w $\circ$ x = wx
Per esempio se ho una stringa vuota, e la concateno ad una stringa, otterrò la stringa (3+0 fa 0, no? :)) $\rightarrow \epsilon \circ$ w = w

Chiaramente si vanno a sommare le lunghezze delle due stringhe in ogni caso.

NON Commutatività di una stringa Concatenare due stringhe non è sempre possibile, a meno che siano perfettamente identiche

Potenze di un alfabeto Prendiamo un alfabeto Σ e per un k intero >= 0 $\Sigma^k = \Sigma x, \Sigma x, \Sigma x, \Sigma x$, ottengo una permutazione di k volte Σ, tutte appartenenti a Σ^k

Come sarà la sua cardinalità? $|\Sigma|$ = q $\rightarrow |\Sigma^k| - q^k$.

Per k = 1 avrei Σ^1 w = qualsiasi elemento di Σ(un solo elemento)

Se ho Σ = 0, 1
Σ^2 = Tutte le permutazioni che posso fare con 0, 1 **i lunghezza 2** (I valori di Σ)

Per definizione $\Sigma^0 = \epsilon$,

Attenzione Quello che è contenuto in Σ è un insieme di STRINGHE non caratteri o simboli.

Chiusura di Kleene $\Sigma^* =$ U per k ≥ 0 $\Sigma^* =$ U per k $\in 0$ ad ∞ di Σ^k

$\Sigma^+ = \Sigma^* - \Sigma^0$, invece Σ^* è considerabile come $\Sigma^+ \cup \Sigma^0$

ATTENZIONE La L che userò nei prossimi passaggi ($\rightarrow L$) è una MACCHINA AUTONOMA che verifica la stringa in questione

Linguaggio L su Σ E' un sotto insieme di Σ^*, o meglio L $\subseteq \Sigma^*$ Ad esempio:

$\Sigma = a, b, c \rightarrow L_1 = aa, cbc \subseteq \Sigma^* L_2 = w \in a, b, c^* t.c.\ W\ contiene\ stesso\ numero\ di\ a\ e\ c$

In pratica

$$L_2 = ac, ca, acb, abc, cab, cba, ...abc \in L_2??Yesccbb \in L_2??Nope$$

Detto meglio (Problema di Membership) $W \in \Sigma^* \rightarrow L \begin{cases} SI,\ Se\ w \in L \\ NO,\ Se\ W \in \Sigma^* senza\ L(Complemento\ di \end{cases}$

Attenzione, il linguaggio è un insieme, contiene quindi ELEMENTI, e di conseguenza può contenere anche l'insieme vuoto!

Osservazione: w può essere appartenente a Σ^* MA non all'insieme vuoto. Occhio a non confondersi

In generale un linguaggio formale va studiato secondo due punti di vista almeno.

Avendo un linguaggio L $\subseteq \Sigma^* posso \begin{cases} generarlo\ (grammatica) \\ riconoscerlo\ (macchina\ autonoma) \end{cases}$

Grammatica Insieme di regole che specificano come va fatta una stringa Una grammatica G è una quadrupla $\rightarrow$ G = (V, T, P, S) in cui

- V: variabili
- T: Simboli terminali
- P: insieme delle produzioni
- $S \in V$: simbolo di start

Problema di decisione E' un problema la cui risposta possibile è sì o no (Cioè alla fine true o false). Risolvere un problema di decisione non pè altro che risolvere un problema di membership.

Il complemento di un linguaggio può essere sia infinito che finito (Nel senso posso escludere elementi oppure posso considerare solo quelli!)

2.1 Linguaggio context-free (CFL) legati a grammatiche Context Free (CFG)

In questo caso si utilizza una forma ricorsiva per definire questi linguaggi,

Ricordiamoci del fatto che io posso mettere due linguaggi in serie, posso includerne uno in un altro MA non posso accavallarli. Nel senso, o tutto di entrambi, o niente. Ma torniamo ai **Context free**

La stringa palindroma Sono stringhe la cui lettura è identica in qualsiasi verso si leggano. Supponiamo $\Sigma = 0, 1$ es. $L_{pal} \subseteq \Sigma^* \rightarrow$ "0110", "11011", ϵ, perchè la stringa vuota è considerata palindroma.
Più in modo formale si può dore che w è **palindroma** quando $w = w^R$ Definizione induttiva:

$$\begin{cases} base: \ \epsilon, 0, 1 \in L_{pal} \\ passo\ induttivo: \ se\ w \in \ L_{pal}\ allora\ OwO,\ 1w1 \ \in \ L_{pal} \end{cases}$$

$$S \in \epsilon S \ \in \ 0S \ \in \ 1S \ \in \ 0S0S \ \in \ 1S1$$

Con S che è una variabile (categoria sintattica), e 0, 1 che sono i simboli terminali con cui si scrivono le stringhe del linguaggio.

Queste si chiamano regole di produzione in cui la testa è occupata in questo caso dalla freccia, mentre i vari 0 1 0S0 e 1S1 sono il corpo. S PUO' diventare il corpo

Detto questo possiamo dire che

$$G_{pal} = (V, T, P, S), incui$$

- V = S
- T = 0, 1
- P = $S \in \epsilon, S \in 0, S \in 1, S \in 0S0, S \in 1S1$

  Più in generale

$$G_{pal} = (S, 0, 1, P, S) dove P = \ S \in \epsilon, ...$$

Derivazione $S \implies 1S1 \implies 10S01$, dove 1S1 è una forma sentenziale e la S cambia in funzione delle regole che ho deciso sopra (per generare la stringa ovviamente.)

In modo compatto:
$S \in \epsilon|0|1|0S0|1S1$

C'è una precisazione da fare, se per esempio avessi la regola che le mie stringhe debbano iniziare per 0, quando andrò a fare 0S0, allora quell'S volendo può essere sostituita con una **nuova** variabile che chiamiamo **X**.

X non è altro che una variabile che rappresenta l'insieme di tutte le palindrome. Perchè cambiare variabile? Perchè se io voglio ad esempio le palindrome che iniziano per 0, devo avere, dato che non posso forzare l'ordine con cui vengono applicate le mie regolo, devo avere un "permesso" speciale che consenta di generare 0 all'inizio alla fine. Cioè, dentro ci può essere un pandemonio, ma fuori ci deve essere la regola che stabilisce l'esistenza di 0.