Note del corso di Calcolabilità e Linguaggi Formali - Lezione 1

Alberto Carraro
5 ottobre 2011

DAIS, Università Ca' Foscari Venezia `http://www.dsi.unive.it/~acarraro`

1 Alfabeti, stringhe linguaggi

Definition 1 (Alfabeto). *Un* alfabeto *è un insieme finito non vuoto.*

Per indicare un alfabeto si usano convenzionalmente le lettere greche maiuscole $\Sigma, \Sigma', \ldots$. Per esempio un noto alfabeto è l'insieme $\{0, 1\}$, detto *alfabeto binario*. Per convenzione gli elementi di una alfabeto si chiamano *simboli* e si usano le prime lettere minuscole dell'alfabeto $a, b, c, \ldots$ per indicare generici simboli di un alfabeto.

Definition 2 (Stringa). *Si Σ un alfabeto. Una* stringa *(o* parola*) su Σ è una sequenza finita di simboli scelti da Σ.*

Ad esempio 01001 è una stringa sull'alfabeto binario. Per indicare le stringhe si usano di solito le ultime lettere minuscole dell'alfabeto inglese come $x, y, u, w, z, \ldots$, con eventuali apici/pedici. Una particolare stringa è quella vuota, composta da zero simboli. Questa stringa, indicata con ϵ, è la sola componibile con qualunque alfabeto. La *lunghezza* di una stringa w, indicata con $|w|$, è la quantità di occorrenze di simboli in essa. Ad esempio $|01001| = 5$ e $|\epsilon| = 0$.

Un'operazione molto utilizzata sulle stringhe è la *concatenazione*. Il risultato della concatenazione di x ad y è la stringa xy di lunghezza $|x| + |y|$ ottenuta giustapponendo i simboli di y, nel loro ordine, all'ultimo simbolo di x. Ad esempio 01001 è la concatenazione di 01 a 010. La concatenazione è un'operazione associativa, cioè $(xy)z = x(yz)$; ciò rende superfluo l'uso delle parentesi, permettendo di scrivere semplicemente xyz.

Diciamo che x è *sottostringa* di y se esistono due stringhe w, w' tali che $wxw' = y$. Due particolari casi di sottostringa sono il *suffisso* ed il *prefisso*. Una stringa x è suffisso di y se esiste una stringa w tale che $wx = y$ ed è prefisso di y se esiste una stringa w tale che $xw = y$. Ad esempio 01 e 11 sono, rispettivamente, un prefisso ed un suffisso della stringa 0101011. Diciamo che un prefisso (resp. suffisso) x di w è *proprio* se $|x| < w$ e *non banale* se $0 < |x| < w$.

La *potenza* n-esima di una stringa x, indicata con x^n, è definita per induzione da $x^0 = \epsilon$ e $x^{n+1} = x^n x$. Ad esempio $(01)^2 = 0101$.

La stringa *rovesciata* di $w = a_1 \cdots a_n$, indicata con w^R, è definita come $w^R = a_n \cdots a_1$.

Definition 3 (Linguaggio). *Un* linguaggio *su un alfabeto Σ è un insieme di stringhe su Σ.*

Per indicare un linguaggio si usano convenzionalmente le lettere maiuscole $L, L', \ldots$. Si noti che ogni alfabeto è un caso particolare di linguaggio su se stesso.

Le operazioni applicabili alle stringhe si possono di norma estendere ai linguaggi. Ad esempio abbiamo

- Rovesciamento: $L^R = \{w^R : w \in L\}$
- Concatenazione: $LL' = \{xy : x \in L,\ y \in L'\}$
- Potenza: $L^0 = \{\epsilon\}, \qquad L^{n+1} = L^n L$

Notate che $\{x^n : x \in L\} \subseteq L^n$ ma i due linguaggi non sono in generale uguali.

Sui linguaggi si possono inoltre applicare le operazioni proprie degli insiemi, come l'unione, l'intersezione, ecc.

Combinando unione e potenza si ottiene un'importante operazione, chiamata *stella (o chiusura) di Kleene*, così definita: $L^* = \bigcup_{n \geq 0} L^n$. Un'operazione da essa derivata è $L^+ = \bigcup_{n \geq 1} L^n = L^* - \{\epsilon\}$.

Risulta dunque che Σ^* è il più grande (al senso dell'inclusione insiemistica) linguaggio su Σ. È per tale motivo che spesso si può genericamente definire un linguaggio come un sottoinsieme di Σ^*.

A questo punto possiamo anche definire formalmente il complemento di un linguaggio L come $L^c = \{w \in \Sigma^* : w \notin L\}$.

1.1 Linguaggi e problemi - una visione d'insieme sul corso

Un tema fondamentale della teoria dei linguaggi formali è la loro classificazione. Il criterio di maggior interesse adottato per tale classificare un linguaggio L è la difficoltà di decidere se, data una stringa w, essa appartiene ad L. La procedura di decisione a cui si fa implicitamente riferimento qui si intende essere implementata da un dispositivo automatico che può essere descritto matematicamente ed il concetto di difficoltà può corrispondere sia alla complicazione/potenza del dispositivo automatico sopraccitato, che al tempo di calcolo richiesto dal dispositivo per dare la risposta.

Il problema di decisione, la cui forma generale è

(∗) "data w, è vero che $w \in L$?"

è di fondamentale importanza perché, tramite opportuna codifica, esso potrebbe corrispondere a domande come "dato un grafo G, è vero che G è bipartito?".

I principali dispositivi automatici, detti *automi*, che incontreremo sono:

- gli Automi Finiti (FA),
- gli Automi a Pila (PDA),
- le Macchine di Turing.

La potenza di una classe di dispositivi è data dalla classe di problemi del tipo (∗) che essa può risolvere, o equivalentemente dalla classe di linguaggi che essa può riconoscere; chiaramente daremo un significato matematico a questa frase. L'elenco rispecchia la potenza di calcolo dei dispositivi citati (crescente dall'alto verso il basso).

La grande differenza delle Macchine di Turing rispetto agli altri tipi di automi è il fatto che esista una Macchina di Turing "general purpose"; al contrario gli Automi Finiti ed a Pila sono "single purpose". Avremo modo di capire cosa significano queste cose.

Ogni tipo di automa corrisponde a un modello astratto di computazione: per ogni tale modello vedremo studieremo delle sue variazioni che possono o meno mutarne la potenza, ovvero la classe dei linguaggi che quel tipo di automa può riconoscere.

Le classi di linguaggi di cui parliamo sono le seguenti:

- i linguaggi regolari,
- i linguaggi liberi dal contesto,
- i linguaggi ricorsivamente enumerabili.

L'ordine di questo elenco riflette il fatto che ognuna di queste classi è definita come la classe di linguaggi riconosciuti dal tipo di automi nella posizione corrispondente dell'elenco precedente. Un tema centrale della teoria dei linguaggi formali è dare caratterizzazioni alternative per le classi di linguaggi sopraccitate. Per definire un linguaggio possiamo utilizzare diversi approcci:

- un approccio *riconoscitivo*, fornando cioè una 'macchina' che ricevendo una stringa in input dice se essa appartiene o no al linguaggio;
- un approccio *algebrico*, mostrando cioè come il linguaggio è costruito a partire da linguaggi più elementari utilizzando operazioni su linguaggi;
- un approccio *generativo*, definendo cioè mediante una grammatica le regole strutturali che devono essere soddisfatte dalle sue stringhe;
- un approccio *descrittivo*, che da delle condizioni sulla forma delle stringhe che appartengono ad un linguaggio. Questo ultimo approccio si applica bene per le classi di linguaggi più ristrette ed è soprattutto utile per dimostrare quando un dato linguaggio *non* appartiene ad una data classe (*Pumping Lemmas*).

Concludiamo questa sezione con una tabella che riassume le corrispondenze tra i temi principali della teoria dei Linguaggi Formali, che occupa la prima parte del corso.

Grammatiche (Chomsky)	**Automi**	**Linguaggi**
Tipo 3	Automi finiti	Regolari
Tipo 2	Automi a pila	Liberi dal contesto
Tipo 1	Automi linear-bounded	Sensibili al contesto
Tipo 0	Decisori Macchine di Turing	Ricorsivi Ricorsivamente enumerabili

La seconda parte del corso invece è dedicata alla teoria della Calcolabilità, che si sviluppa uno studio approfondito dei linguaggi citati nell'ultima riga della tabella qui sopra. In particolare la teoria della Calcolabilità si sviluppa su un modello matematico di computazione alternativo alle macchine di Turing (ma equivalente) basato su insiemi di numeri naturali e funzioni su di essi.