Linguaggi Formali, Automi e Logiche

Angelo Montanari
Dipartimento di Matematica e Informatica
Università di Udine, Italy
montana@dimi.uniud.it t

Struttura dell'intervento

0) cos'è un linguaggio formale? macchine, automi e logiche

1) la macchina di Turing

2) computabilità: decidibilità e indecidibilità

3) complessità: trattabilità e intrattabilità

4) gerarchie di linguaggi

5) linguaggi e automi/macchine

6) linguaggi, automi e logiche

Cos'è un linguaggio formale? - 1

Sia data la nozione primitiva di simbolo.

Una *parola* (o stringa) finita è una sequenza finita di simboli giustapposti (esempio: *acba*).

La *concatenazione* di due parole *u* e *v*, notazione *uv*, è (definita come) la loro giustapposizione.

Siano dati due insiemi A e B di parole. L'insieme concatenazione di A e B è $A \cdot B = \{uv : \ u \in A, \ v \in B\}$.

Cos'è un linguaggio formale? - 2

Per ogni $n \geq 0$, definiamo ricorsivamente l'insieme A^n nel seguente modo: $A^0 = \{\epsilon\}$, dove ϵ è la parola vuota, $A^{n+1} = A \cdot A^n$. L'insieme chiusura (di Kleene) di A è $A^* = \bigcup_{n \geq 0} A^n$.

Dato un insieme finito di simboli A, detto *alfabeto*, una parola su A è un elemento di A^*.

Un **linguaggio** (di parole finite) su A è un sottoinsieme di A^*.

Problemi fondamentali

- problema dell'appartenenza: dati L e $x \in A^*$, $x \in L$?
- problema del vuoto: $L = \emptyset$?
- problema dell'universalità: $L = A^*$?
- problema dell'inclusione: $L_1 \subseteq L_2$?
- problema dell'equivalenza: $L_1 = L_2$?

Come caratterizzare un linguaggio?

Esploreremo tre punti di vista alternativi:

- linguaggi **accettati** da una macchina/automa (linguaggio = insieme delle stringhe accettate dall'automa)
- linguaggi **generati** da una grammatica (linguaggio = insieme delle parole generate dalla grammatica)
- linguaggi **definiti** da una formula (linguaggio = insieme dei modelli della formula)

La nozione di automa/macchina

Un automa/macchina è un modello formale per la descrizione di procedure effettive (**algoritmi**).

Ogni automa/macchina deve possedere le seguenti proprietà:

(i) ogni procedura deve essere finitamente descrivibile;

(ii) ogni procedura deve consistere di un insieme di passi discreti, ciascuno dei quali deve essere eseguibile automaticamente.

Modello classico: la **macchina di Turing**

La macchina di Turing: struttura

Una macchina di Turing è costituita da un controllo finito, un nastro di ingresso suddiviso in celle e una testina di lettura che scandisce una cella del nastro alla volta. Il nastro è limitato a sinistra (esiste una cella più a sinistra di tutte), ma è infinito a destra. Ogni cella del nastro può contenere esattamente un simbolo appartenente ad un insieme finito Γ di simboli (simboli di nastro). Inizialmente, le n celle più a sinistra del nastro, per un qualche $n \geq 0$, contengono l'input, costituito da una stringa di simboli scelti da un opportuno sottoinsieme Σ dei simboli di nastro (simboli di input). Le restanti (infinite) celle del nastro contengono un simbolo di nastro speciale (blank), non appartenente ai simboli di input.

La macchina di Turing: mosse

In una mossa, la macchina di Turing, in base al simbolo presente sulla cella corrente (letto dalla testina) e allo stato corrente del controllo finito,

1. cambia stato;
2. stampa un simbolo sulla cella corrente, sostituendo quello ivi presente;
3. muove la testina nella cella immediatamente alla destra o alla sinistra della cella corrente.

Il **linguaggio accettato da una macchina di Turing** M è l'insieme di tutte le parole in Σ^* tali che, quando inserite sul nastro allineate a sinistra, con il controllo posto nello stato iniziale e la testina posizionata sulla cella più a sinistra, danno origine ad una computazione di M che termina in uno stato finale.

La macchina di Turing: un esempio

Vediamo il **funzionamento di una macchina di Turing** M che accetta il linguaggio $L = \{0^n 1^n . n \geq 1\}$. Inizialmente, il nastro di M contiene la parola $0^n 1^n$, seguita da un'infinità di blank. Ripetutamente, M rimpiazza l'occorrenza più a sinistra di 0 con X, si muove verso destra fino ad incontrare il primo 1 e lo sostituisce con Y, si muove verso sinistra fino ad incontrare la prima X, si muove di una cella a destra per posizionarsi sull'occorrenza più a sinistra di 0 (se esiste) e ripete il ciclo. Se, ad una data iterazione, M trova blank anziché 1, la computazione termina con fallimento (la parola non è accettata). Se, dopo aver sostituito 1 con Y, M non trova più 0, controlla che non vi siano altri 1 e, in tal caso, accetta la parola; altrimenti, termina con fallimento.

Linguaggi e funzioni computabili

La macchina di Turing può essere caratterizzata in termini di (caratterizzazioni alternative, ma equivalenti):

(i) la **classe di linguaggi** che definisce (linguaggi/insiemi ricorsivamente enumerabili)

(ii) la **classe di funzioni intere** che computa (funzioni ricorsive parziali)

Osservazione: ogni problema si può formulare come un problema di riconoscimento (accettazione) di un linguaggio.

Linguaggi ricorsivi e ricorsivamente enumerabili

Un linguaggio è detto **ricorsivo** se è accettato da almeno una macchina di Turing che termina (con successo o fallimento) su tutti i possibili input.

La classe dei linguaggi **ricorsivamente enumerabili** include, invece, dei linguaggi per i quali non è possibile determinare automaticamente l'appartenza. Più precisamente, s L è uno di tali linguaggi, ogni macchina di Turing che lo riconosce non termina su qualcuno degli input che non appartengono a L. Al contrario, se l'input appartiene a L, la computazione della macchina di Turing termina.

Problema (linguaggi ricorsivamente enumerabili, ma non ricorsivi): durante l'elaborazione di un dato input da parte di una macchina di Turing non è possibile sapere se la computazione è destinata a terminare con successo (accettazione dell'input), o con fallimento (rufiuto dell'input), o a proseguire all'infinito.

Funzioni ricorsive totali e parziali

Le funzione **ricorsive totali** in k argomenti sono definite per tutte le k tuple di interi (controparte degli insiemi ricorsivi).

Le funzioni intere in k argomenti calcolate da una macchina di Turing sono dette funzioni **ricorsive parziali** (controparte degli insiemi ricorsivamente enumerabili). Esse sono calcolate da macchine di Turing che possono o meno fermarsi su un dato input.

La tesi di Church

L'assunzione che la nozione intuitiva di "funzione computabile" possa essere identificata con la classe delle funzioni ricorsive parziali (e quindi con la nozione di macchina di Turing) è nota con il nome di **tesi di Church**.

Un'ampia gamma di modelli di computazione alternativi alla macchina di Turing è stata proposta in letteratura e mostrata equivalente alla macchina di Turing dal punto di vista del potere computazionale.

La macchina di Turing universale

La **macchina di Turing universale** è una macchina di Turing che accetta in ingresso la descrizione di una qualsiasi macchina di Turing M e un qualunque dato di input per M e fornisce in uscita il risultato della computazione di M in corrispondenza del dato z.

Tale macchina di Turing consente di computatare tutte le funzioni computabili da una macchina di Turing.

La macchina di Turing universale è un buon modello del concetto di **calcolatore a programma memorizzato**.

Computabilità e non computabilità

È facile mostrare che **esistono funzioni non computabili** (ci sono più funzioni che nomi per designare funzioni o algoritmi per computarle)

Intuizione (l'argomento può essere formalizzato)

Si assuma che per ogni funzione computabile esista un algoritmo/programma che la computa. Poiché ogni algoritmo è finitamente specificabile (un algoritmo non è altro che una stringa di lunghezza finita su un qualche alfabeto finito), l'insieme di tutti gli algoritmi è numerabile. Si considerino ora tutte le funzioni che mappano gli interi in 0 e 1. Si supponga, per assurdo, che l'insieme di tali funzioni sia numerabile e che i suoi elementi (le funzioni) siano stati messi in corrispondenza con l'insieme dei numeri interi. Sia f_i la funzione che corrisponde all'intero i-esimo. La funzione che assegna il valore 0 all'i-esimo intero se f_i assegna ad esso il valore 1, e il valore 1 altrimenti, non corrisponde ad alcun intero (contraddizione).

Terminazione della macchina di Turing

Vi sono diversi problemi di rilevanza pratica che risultano essere non computabili.

Esempio: Esiste un algoritmo in grado di decidere, per una qualunque macchina di Turing M e un qualunque dato di ingresso x, se la computazione di M su x termina?

È possibile dimostrare che un tale algoritmo non esiste (enumerazione delle macchine di Turing e argomento diagonale).

Osservazioni

Osservazione 1: l'essere una funzione non computabile non implica che per nessun argomento è possibile conoscere il corrispondente valore, ma solo che non esiste un algoritmo che permetta di calcolarla per ogni argomento.

Osservazione 2: l'essere computabili o meno è una proprietà intrinseca delle funzioni e non del particolare modello formale usato per descriverle.

Decidibilità e indecidibilità

Dato un problema, definiamo **istanza** del problema una lista di argomenti, uno per ogni parametro del problema.

Se restringiamo la nostra attenzione a problemi con risposta sì/no (tale restrizione non è limitativa) e codifichiamo le istanze del problema come stringhe su un qualche alfabeto finito, possiamo trasformare la questione circa l'esistenza o meno di un algoritmo che risolve il problema in una questione relativa alla natura ricorsiva o meno di un particolare linguaggio.

Un problema il cui linguaggio è ricorsivo è detto **decidibile**; altrimenti, il problema è detto **indecidibile**.

In altri termini, un problema è indecidibile se non esiste alcun algoritmo che riceva in input un'istanza del problema e determini se la risposta per tale istanza è sì o no.

La complessità computazionale

Stabilire se un problema è o meno decidibile non è in generale sufficiente. Sapere che un problema può essere risolto, infatti, non garantisce l'esistenza di un algoritmo realmente utilizzabile che lo risolve, in quanto la quantità di risorse (tempo e/o di spazio) richiesta da un tale algoritmo può essere esorbitante

La classificazione dei linguaggi sulla base dell'ammontare di risorse (tempo e/o spazio) richiesto per il loro riconoscimento da parte di un qualche strumento di calcolo (ad esempio, una macchina di Turing) prende il nome di **complessità computazionale**.

Trattabilità e intrattabilità

Fra i problemi decidibili, ve ne sono alcuni così difficili (in termini di complessità computazionale) che, per tutti i possibili scopi pratici, non possono essere risolti nella loro piena generalità da un calcolatore. Tali problemi sono detti **problemi intrattabili**.

Per alcuni di tali problemi, si è dimostrato che determinare la soluzione richiede un tempo esponenziale. Per altri, è "molto probabile" che la loro risoluzione richieda un tempo esponenziale (se fosse possibile trovare una soluzione in modo più veloce, una grande quantità di importanti problemi in matematica, in informatica e in altre discipline, problemi per i quali buone soluzioni sono state cercate invano per anni, potrebbe essere risolta con strumenti assai migliori rispetto a quelli noti).

$\mathcal{P} = \mathcal{NP}$?

Gerarchie di classi di complessità.

In particolare, $\mathcal{P} \subseteq \mathcal{NP}$ dove

- $\mathcal{P}$ (trattabile) identifica la classe dei problemi la cui soluzione può essere trovata in un tempo polinomiale.
- $\mathcal{NP}$ (intrattabile) identifica la classe dei problemi in cui è possibile verificare in tempo polinomiale una possibile soluzione (" È più facile criticare che fare" M. Vardi).

Un problema aperto: $\mathcal{P} = \mathcal{NP}$?

Linguggi formali: la gerarchia di Chomski

Definiamo **grammatica** una quadrupla di elementi $G = (V, T, P, S)$, dove V è un insieme finito di variabili, T è un insieme finito di simboli terminali, P è un insieme finito di regole di produzione e S è una variabile speciale, detta *simbolo iniziale*.

La **gerarchia di Chomski**:

1. linguaggi regolari
2. linguaggi liberi dal contesto
3. linguaggi sensibili al contesto
4. linguaggi ricorsivamente enumerabili

Linguaggi regolari

I linguaggi regolari sono i linguaggi generati da una grammatica regolare, le cui regole di produzione sono del tipo:

- $A \rightarrow wB$, oppure
- $A \rightarrow w$,

dove $A, B \in V$ e w è una stringa, eventualmente vuota, appartenente a T^* (grammatica regolare destra) o, equivalentemente, del tipo (grammatica regolare sinistra):

- $A \rightarrow Bw$, oppure
- $A \rightarrow w$.

Linguaggi liberi dal contesto

I linguaggi liberi dal contesto sono i linguaggi generati da una grammatica libera dal contesto, le cui regole di produzione sono del tipo:

- $A \rightarrow \alpha$,

dove $A \in V$ e α è una stringa di simboli appartenente a $(V \cup T)^*$.

Linguaggi sensibili al contesto

I linguaggi sensibili al contesto sono i linguaggi generati da una grammatica sensibile al contesto, le cui regole di produzione sono del tipo:

- $\alpha \rightarrow \beta$,

dove $\alpha(\neq \epsilon)$ e $beta$ sono stringhe arbitrarie di simboli grammaticali tali che la lunghezza β è maggiore o uguale alla lunghezza di α.

Perchè il nome linguaggi sensibili al contesto? Esiste una forma normale per tali grammatiche in cui ogni produzione è del tipo:

- $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$,

con $\beta \neq \epsilon$.

Linguaggi ricorsivamente enumerabili

I linguaggi ricorsivamente enumerabili sono i linguaggi generati da una grammatica a struttura di frase, le cui regole di produzione sono del tipo:

- $\alpha \rightarrow \beta$,

dove $\alpha(\neq \epsilon)$ e $beta$ sono stringhe arbitrarie di simboli grammaticali.

Le relazioni tra i linguaggi della gerarchia

È possibile dimostrare che:

- i linguaggi regolari sono propriamente contenuti nei linguaggi liberi dal contesto;
- i linguaggi liberi dal contesto, che non contengono la stringa vuota, sono propriamente contenuti nei linguaggi sensibili al contesto;
- i linguaggi sensibili al contesto sono propriamente contenuti nei linguaggi ricorsivamente enumerabili.

I linguaggi sensibili al contesto sono **ricorsivi** (e quindi tali sono anche i linguaggi regolari e i linguaggi liberi dal contesto), ma esistono linguaggi ricorsivi che non sono linguaggi sensibili al contesto (i linguaggi sensibili al contesto non esauriscono i linguaggi ricorsivi).

Proprietà di chiusura dei linguaggi

Dati L, L_1, L_2 linguaggi regolari,

- $L_1 \cup L_2$ è un linguaggio regolare (chiusura rispetto all'operazione di unione)?
- $L_1 \cap L_2$ è un linguaggio regolare (chiusura rispetto all'operazione di intersezione)?
- $L_1 \cdot L_2$ è un linguaggio regolare (chiusura rispetto all'operazione di concatenazione)?
- L^* è un linguaggio regolare (chiusura rispetto all'operatore $*$)?
- $A^* - L$ è un linguaggio regolare (chiusura rispetto alla complementazione)?

Analogamente per le altre classi di linguaggi.

Problemi fondamentali (rivisitati)

Decidibilità dei problemi e **complessità** dei problemi decidibili.

- problema dell'appartenenza: dati L e $x \in A^*$, $x \in L$ (linguaggi ricorsivi vs. linguaggi ricorsivamente enumerabili)?
- problema del vuoto: $L = \emptyset$?
- problema dell'universalità: $L = A^*$ (se vale la chiusura rispetto alla complementazione, è equivalente a $A^* - L = \emptyset$?)?
- problema dell'inclusione: $L_1 \subseteq L_2$ (se vale la chiusura rispetto alla complementazione, è equivalente a $L_1 \cap (A^* - L_2) = \emptyset$?)?
- problema dell'equivalenza: $L_1 = L_2$ (equivalente a $L_1 \subseteq L_2 \wedge L_2 \subseteq L_1$?)?

Linguaggi formali e macchine

Risultati relativi alle proprietà di chiusura, alla decidibilità e alla complessità dei linguaggi si ricavano sfruttando le seguenti **corrispondenze**:

linguaggi regolari / automi a stati finiti (analizzatori lessicali, forme normali per teorie logiche, etc.)

linguaggi liberi dal contesto / pushdown automata (parser)

linguaggi sensibili al contesto/ linear bounded automata

linguaggi ricorsivamente enumerabili / macchine di Turing

Linguaggi formali, automi e logiche

Obiettivo: studio dei linguaggi formali (principalmente linguaggi regolari) nell'ambito della logica matematica.

Connessione tra automi e logica: gli automi a stati finiti (su parole finite) e la logica monadica al second'ordine (interpretata su parole/modelli finiti) hanno lo **stesso potere espressivo**. Le trasformazioni da formule ad automi e viceversa sono trasformazioni effettive.

Gli automi a stati finiti possono essere visti come una **forma normale** per le formule della logica monadica al second'ordine.

La corrispondenza tra automi e logica può essere estesa al **caso infinito**: gli automi a stati finiti (su parole infinite) e la logica monadica al second'ordine (interpretata su parole/modelli infiniti) hanno lo **stesso potere espressivo**.

Decidibilità elementare e non elementare

Il problema della (in)soddisfacibilità delle formule della logica monadica al second'ordine (lo stesso vale per il frammento al prim'ordine della logica monadica) è decidibile, ma **non elementarmente decidibile**.

Un problema si dice **elementarmente decidibile** se la sua complessità computazionale (in termini di tempo di calcolo), su un dato di ingresso di dimensione n, è superiormente limitata da una funzione $2^{2^{..^{2^n}}}$, per un qualche numero finito di esponenti.

Logiche classiche e logiche temporali

Passaggio dalle logiche monadiche al primo e al second'ordine alle corrispondenti **logiche temporali**, che preserva la completezza espressiva.

Vantaggio: il problema della (in)soddisfacibilità da non elementarmente decidibile diventa elementarmente decidibile.

Le formule della logica temporale proposizionale possono essere viste come una **forma normale** per le formule della logica monadica al prim'ordine (vale un risultato analogo per la logica monadica al second'ordine).

Applicazioni: algoritmi di verifica della soddisfacibilità e algoritmi di model checking.

Bibliografia

J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley Pub. Co., 1979.

M. Franceschet e A. Montanari, Linguaggi formali, automi e logiche (draft), Note al corso di Metodi formali dell'informatica: sistemi reattivi, 2000.

M.O. Rabin, Decidable Theories, in Handbook of Mathematical Logic, North-Holland, 1977.

W. Thomas, Automata on Infinite Objects, Handbook of Theoretical Computer Science, J. van Leeuwen (ed.), Elsevier Science Pub., 1990.

W. Thomas, Languages, Automata, and Logic, Handbook of Formal Languages, Vol. III, 1997.