

Grammatiche formali e automi

Le grammatiche formali sono un dispositivo per distinguere tra stringhe che sono frasi di una lingua e stringhe che non lo sono.

Un altro dispositivo per fare questa distinzione sono gli *automi*.

Cos'è un automa? È una sorta di modello astratto di un computer. Per capire più concretamente in cosa consiste un automa, esaminiamone un tipo particolare.

Automi finiti

Definizione informale

Gli *automi finiti*, o *macchine a stati finiti*, possono essere descritti intuitivamente così:

- un automa finito è un dispositivo computazionale astratto che riceve una stringa di simboli come *input* (ingresso), legge un simbolo alla volta da sinistra a destra e, dopo aver letto l'ultimo simbolo, si ferma e manifesta accettazione oppure rifiuto;
- in qualsiasi punto di questo processo, un automa finito è in uno stato, e il numero degli stati possibili per l'automa è finito;
- i calcoli di un automa finito sono guidati da un 'programma', cioè da un insieme finito di istruzioni per passare da uno stato all'altro mentre l'automa legge i simboli di ingresso;
- il processo di calcolo inizia sempre in uno stato designato, detto *stato iniziale*;
- tra gli stati possibili, ci sono anche degli *stati finali*. Se l'automa finisce in uno di questi stati dopo aver letto la stringa di ingresso, la stringa è accettata; altrimenti è rifiutata.

Automi deterministici

Se, per ogni stato in cui l'automa può essere e ogni simbolo che l'automa può leggere, c'è una e una sola transizione possibile ad uno stato, l'automa è detto *deterministico*.

Verso una definizione rigorosa

Ora che abbiamo un'idea intuitiva di cos'è un automa finito deterministico, definiremo questa nozione in modo più rigoroso.

Prima però dobbiamo fare un po' di ripasso.

Ripasso

Prodotto cartesiano

Il *prodotto cartesiano* degli insiemi A e B, denotato da $A \times B$, è l'insieme di tutte le coppie ordinate (x,y) , dove $x \in A$ e $y \in B$.

Per esempio, se $A = \{a, b, c\}$ e $B = \{1, 2\}$, allora $A \times B = \{(a, 1), (b, 1), (c, 1), (a, 2), (b, 2), (c, 2)\}$

Ripasso

Relazioni binarie

Una *relazione* dall'insieme A all'insieme B è un sottoinsieme di $A \times B$ (cioè, è un insieme di coppie ordinate (x,y) , dove $x \in A$ e $y \in B$).

Per esempio, se $A = \{a, b, c\}$ e $B = \{1, 2\}$, allora una relazione dall'insieme A all'insieme B è $\{(a, 1), (c, 1), (a, 2), (c, 2)\}$. Un'altra relazione dall'insieme A all'insieme B è $\{(a, 1), (b, 1), (c, 1), (a, 2), (b, 2), (c, 2)\}$

(N.B.: un sottoinsieme di un insieme S non è necessariamente un sottoinsieme *proprio* di S. Per esempio, $\{a, b, c\}$ è un sottoinsieme proprio di $\{a, b, c, d\}$. Inoltre, $\{a, b, c, d\}$ è un sottoinsieme di $\{a, b, c, d\}$, ma non un sottoinsieme proprio di $\{a, b, c, d\}$.)

Ripasso

Funzioni

Una *funzione* dall'insieme A all'insieme B è una relazione R da A a B con questa proprietà: per ogni $x \in A$, c'è *esattamente una* coppia ordinata in R che ha x come primo elemento.

Per esempio, se $A = \{a, b, c\}$ e $B = \{1, 2\}$, allora la relazione $R = \{(a, 1), (b, 1), (a, 2), (c, 2)\}$ non è una funzione, perché c'è più di una coppia ordinata in R che ha a come primo elemento.

Se $A = \{a, b, c\}$ e $B = \{1, 2\}$, allora la relazione $R = \{(a, 1), (c, 2)\}$ non è una funzione, perché non c'è alcuna coppia ordinata in R che ha b come primo elemento.

Se $A = \{a, b, c\}$ e $B = \{1, 2\}$, allora la relazione $R = \{(a, 1), (b, 1), (c, 1)\}$ è una funzione, perché, per ogni elemento di A, c'è esattamente una coppia ordinata in R che ha quell'elemento come primo elemento.

Automa finito deterministico

Definizione formale

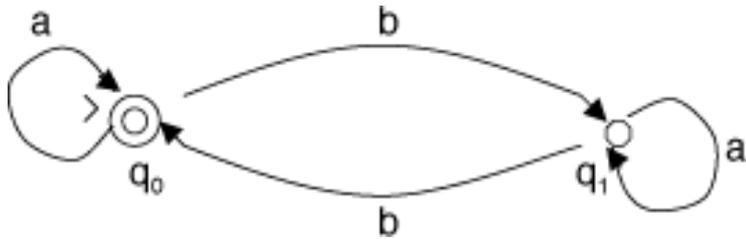
Un automa finito deterministico è una quintupla $M = (K, \Sigma, \delta, s, F)$, dove

- K è insieme finito di stati
- Σ è un alfabeto
- $s \in K$ è lo stato iniziale
- $F \subseteq K$ è l'insieme degli stati finali
- δ , la funzione di transizione, è una funzione da $K \times \Sigma$ a K (cioè, una funzione che associa uno stato a ogni coppia di stati e simboli dell'alfabeto)

Diagrammi di stato

Un modo conveniente per rappresentare graficamente gli automi finiti sono i *diagrammi di stato*.

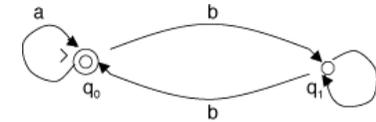
Un diagramma di stato è un grafo in cui gli stati sono rappresentati da nodi etichettati congiunti da frecce. I simboli dell'alfabeto sono le etichette delle frecce che si dipartono dai nodi. Lo stato iniziale è il nodo indicato dal simbolo $>$. Gli stati finali sono i nodi indicati da circoli concentrici. Se la lettura di un simbolo α causa una transizione da uno stato s a uno stato s' , nel diagramma c'è una freccia da s a s' etichettata α . Ecco un esempio di diagramma di stato:



Esercizio

Diagrammi di stato

Che automa finito corrisponde a questo diagramma di stato?

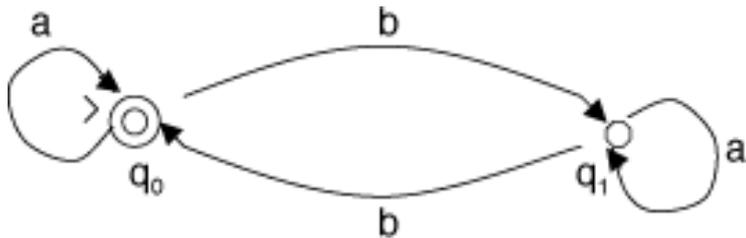


Specificate l'insieme degli stati K , l'alfabeto Σ , lo stato iniziale s , l'insieme F degli stati finali e la funzione di transizione δ da stati e simboli a stati:

$$\begin{aligned}
 K &= \{ \quad \} \\
 \Sigma &= \{ \quad \} \\
 s &= \\
 F &= \{ \quad \} \\
 \delta(q_0, a) &= \\
 \delta(q_0, b) &= \\
 \delta(q_1, a) &= \\
 \delta(q_1, b) &=
 \end{aligned}$$

Soluzione

Questo diagramma di stato



rappresenta l'automa finito seguente:

$$\begin{aligned}
 K &= \{ q_0, q_1 \} \\
 \Sigma &= \{ a, b \} \\
 s &= q_0 \\
 F &= \{ q_1 \} \\
 \delta(q_0, a) &= q_0 \\
 \delta(q_0, b) &= q_1 \\
 \delta(q_1, a) &= q_1 \\
 \delta(q_1, b) &= q_0
 \end{aligned}$$

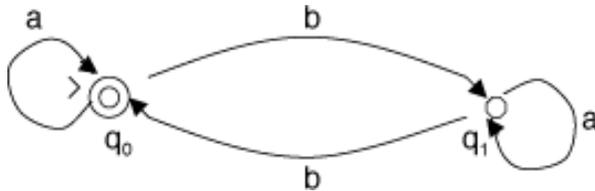
Accettazione

Automi finiti deterministici

- Una stringa è accettata da un automa finito deterministico se, dopo che è stata letta, l'automa è in uno stato finale.
- Il linguaggio accettato da un automa finito deterministico è l'insieme delle stringhe accettate dall'automa.

Esercizio**Accettazione di stringhe**

Considerate di nuovo questo diagramma di stato:

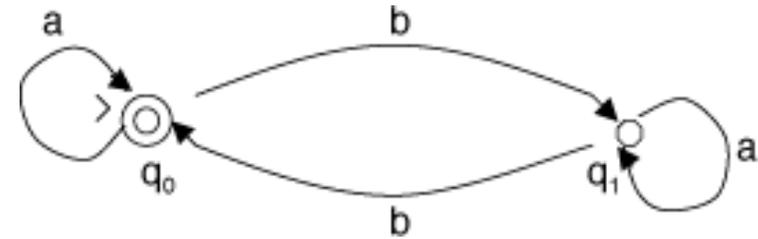


Quale delle stringhe seguenti sono accettate dall'automa che questo diagramma di stato rappresenta?

- (i) a
- (ii) aa
- (iii) ba
- (iv) bbabaab
- (v) bbabaa
- (vi) bbb
- (vii) aab
- (viii) aabb

Soluzione**Accettazione di stringhe**

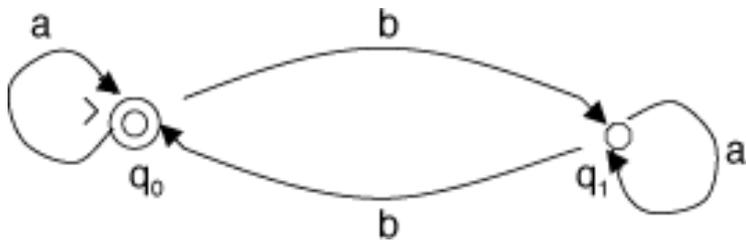
Le stringhe accettate dall'automa rappresentato da questo diagramma di stato sono le stringhe (i), (ii), (iv), (viii) (l'asterisco indica che una stringa è rifiutata):



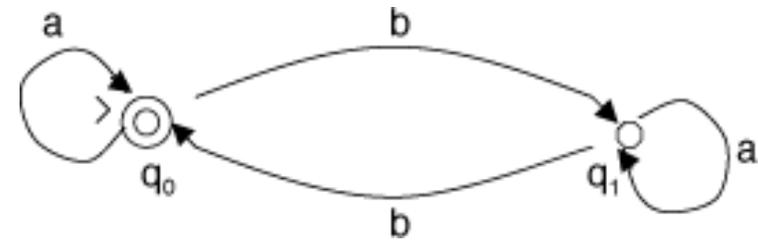
- (i) a
- (ii) aa
- (iii) *ba
- (iv) bbabaab
- (v) *bbabaa
- (vi) *bbb
- (vii) *aab
- (viii) aabb

Esercizio**Linguaggio accettato da un automa**

- Possiamo dare una descrizione del linguaggio accettato da questo automa?

**Soluzione****Linguaggio accettato da un automa**

- Il linguaggio accettato da questo automa è l'insieme delle stringhe che possono essere formate a partire dall'alfabeto { a, b } e che hanno un numero pari di b oppure non hanno b.



La ragione è questa. L'automa passa dallo stato q_0 allo stato q_1 e dallo stato q_1 allo stato q_0 quando legge b. Praticamente, l'automa ignora gli a, in quanto quando legge un a rimane nello stato in cui è, e conta invece i b. Dal momento che q_0 è l'unico stato finale, per arrivare a uno stato finale l'automa deve leggere un numero pari di b oppure non leggere neppure un b.

Configurazioni

In ogni dato momento mentre sta leggendo la stringa di ingresso, un automa finito è in una certa *configurazione* che è identificata dal suo stato corrente e dalla parte della stringa di ingresso che rimane ancora da leggere.

Quindi, le configurazioni di un automa finito possono essere rappresentate da coppie (q, ω) dove q è uno stato (lo stato in cui si trova l'automa) e ω la parte della stringa di ingresso che non è ancora stata letta.

Le configurazioni c e c' di un automa M stanno nella relazione binaria \vdash_M (la relazione di *transizione*) se e solo se l'automa può *passare in una singola mossa* da c a c' (in questo caso scriviamo: $c \vdash_M c'$).

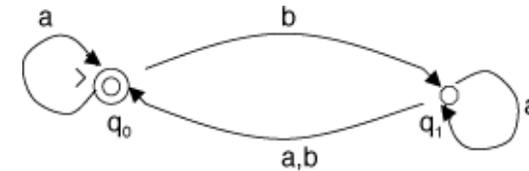
Per esempio, le transizioni attraverso le quali la stringa bbabaab è accettata dall'automa che stiamo considerando (chiamiamolo M) sono rappresentate in (1) per mezzo di \vdash_M :

$$(1) \quad (q_0, bbabaab) \vdash_M (q_1, babaab) \vdash_M (q_0, abaab) \vdash_M (q_0, baab) \vdash_M (q_1, aab) \vdash_M (q_1, ab) \vdash_M (q_1, b) \vdash_M (q_0, \epsilon)$$

Esercizio

Determinismo

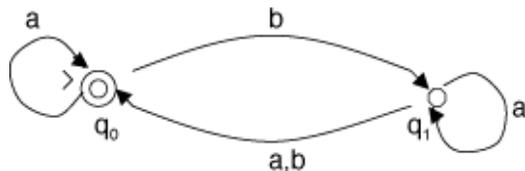
- L'automa rappresentato da questo diagramma di stato è deterministico?



Soluzione

Determinismo

- L'automa rappresentato da questo diagramma di stato non è deterministico, in quanto da q_1 c'è più di una transizione possibile se l'automa legge il simbolo 'a'.



Automa finito non deterministico

Definizione formale

Un automa finito non deterministico è una quintupla $M = (K, \Sigma, \delta, s, F)$, dove

- K è insieme finito di stati
- Σ è un alfabeto
- $s \in K$ è lo stato iniziale
- $F \subseteq K$ è l'insieme degli stati finali
- δ , la relazione di transizione, è un sottoinsieme di $(K \times (\Sigma \cup \{ \epsilon \})) \times K$ (cioè, è un insieme di coppie ordinate (a, b) , dove a è una coppia ordinata il cui primo elemento è uno stato in K e il secondo è un simbolo in Σ oppure è la stringa vuota, e b è uno stato in K).

Automa finito non deterministico

Commenti sulla definizione

- Nella definizione di automa finito non deterministico non abbiamo più una *funzione* di transizione da stati e simboli dell'alfabeto a stati, ma una *relazione* di transizione da stati e simboli dell'alfabeto (accresciuto della stringa vuota) a stati. Quindi, la definizione non richiede più che, quando l'automa si trova in uno stato e legge un simbolo dell'alfabeto, ci sia un unico stato in cui l'automa può passare.
- Nella definizione di automa finito non deterministico la relazione di transizione non è l'insieme $(K \times (\Sigma \cup \{e\})) \times K$, ma un sottoinsieme di $(K \times (\Sigma \cup \{e\})) \times K$. Quindi, la definizione non richiede più che, quando l'automa si trova in uno stato e legge un simbolo dell'alfabeto, ci sia almeno uno stato in cui l'automa può passare o rimanere. Per certi simboli di ingresso può non esserci alcuna mossa possibile (l'automa si blocca).
- La definizione ammette la possibilità che l'automa passi da uno stato s a uno stato s' senza leggere niente. Questo accade quando la transizione da s a s' è determinata dalla stringa vuota (vale a dire, nel caso in cui nella relazione di transizione abbiamo la coppia $\delta(s,e) = s'$).

Accettazione

Automi finiti non deterministici

- Una stringa è accettata da un automa finito non deterministico se è *possibile* che, dopo che la stringa è stata letta, l'automa sia in uno stato finale.
- Il linguaggio accettato da un automa finito non deterministico è l'insieme delle stringhe accettate dall'automa.

Potere di riconoscimento degli automi finiti

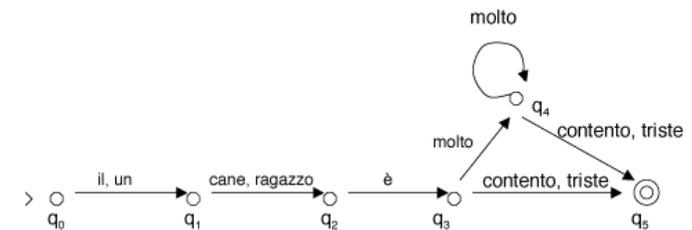
Come dovrebbe essere chiaro dalle definizioni, gli automi finiti deterministici sono un tipo particolare di automa finito non deterministico: cioè sono un tipo di automa finito non deterministico in cui la relazione δ è una funzione da $K \times \Sigma$ a K .

Sorprendentemente, è possibile mostrare (non lo faremo qui) che gli automi finiti non deterministici non hanno un potere di riconoscimento maggiore di quelli deterministici:

- un automa finito non deterministico può sempre essere convertito in un automa finito deterministico che accetta esattamente lo stesso linguaggio.

Un esercizio

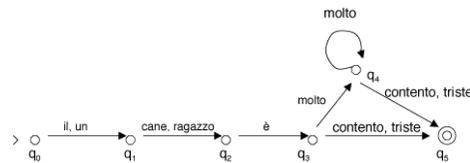
Questo grafo rappresenta un automa finito non deterministico:



Quale automa?

$K = \{ \quad \}$
 $\Sigma = \{ \quad \}$
 $s =$
 $F = \{ \quad \}$
 $\delta(q_0, il) =$
 $\delta(q_0, un) =$
 $\delta(q_1, cane) =$
 \dots

La soluzione



$$K = \{ q_0, q_1, q_2, q_3, q_4, q_5 \}$$

$$\Sigma = \{ \text{il, un, cane, ragazzo, è, molto, contento, triste} \}$$

$$s = q_1$$

$$F = \{ q_5 \}$$

$$\delta(q_0, \text{il}) = q_1$$

$$\delta(q_0, \text{un}) = q_1$$

$$\delta(q_1, \text{cane}) = q_2$$

$$\delta(q_1, \text{ragazzo}) = q_2$$

$$\delta(q_2, \text{è}) = q_3$$

$$\delta(q_3, \text{contento}) = q_5$$

$$\delta(q_3, \text{triste}) = q_5$$

$$\delta(q_4, \text{molto}) = q_4$$

$$\delta(q_4, \text{molto}) = q_4$$

$$\delta(q_4, \text{contento}) = q_5$$

$$\delta(q_4, \text{triste}) = q_5$$

Riconoscere un numero infinito di frasi

L'esempio precedente mostra che gli automi finiti sono in grado di riconoscere un numero infinito di frasi dell'italiano.

Per esempio, l'automa precedente accetta un numero infinito di frasi di questa forma:

- (2) a. il cane è molto triste
- b. il cane è molto molto triste
- c. il cane è molto molto molto triste
- d. il cane è molto molto molto ... triste

Se, o

Supponiamo ora di voler costruire un automa finito che riconosce tutte e solo le frasi dell'italiano che hanno la forma *se...*, *allora...* e *o...o...* e che utilizzano il vocabolario dell'automa precedente.

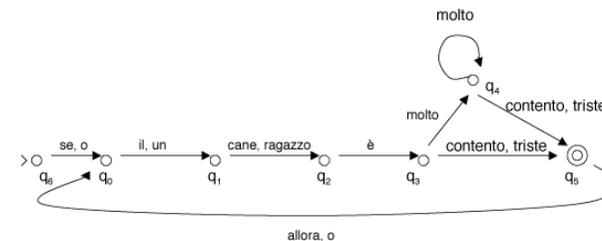
Per esempio, l'automa deve accettare frasi come quelle in (3):

- (3) a. o il cane è triste o il ragazzo è contento
- b. se il cane è triste allora il ragazzo è contento

Come si fa?

Primo tentativo

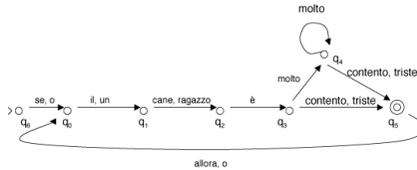
Questo automa accetta le sequenze di parole in (3):



- (3) a. o il cane è triste o il ragazzo è contento
- b. se il cane è triste allora il ragazzo è contento

Problemi per il primo tentativo

Il problema con l'automa che abbiamo costruito è che, oltre ad accettare le frasi in (3), accetta anche quelle in (4):



- (3) a. o il cane è triste o il ragazzo è contento
b. se il cane è triste allora il ragazzo è contento

- (4) a. o il cane è triste
b. o il cane è triste allora il ragazzo è contento
c. se il cane è triste
d. se il cane è triste o il ragazzo è contento

Questo non va bene se vogliamo che il nostro automa riconosca solo frasi dell'italiano.

Diagnosi

La ragione per cui l'automa che abbiamo costruito ha difficoltà a distinguere le frasi in (3) dalle sequenze in (4) è questa:

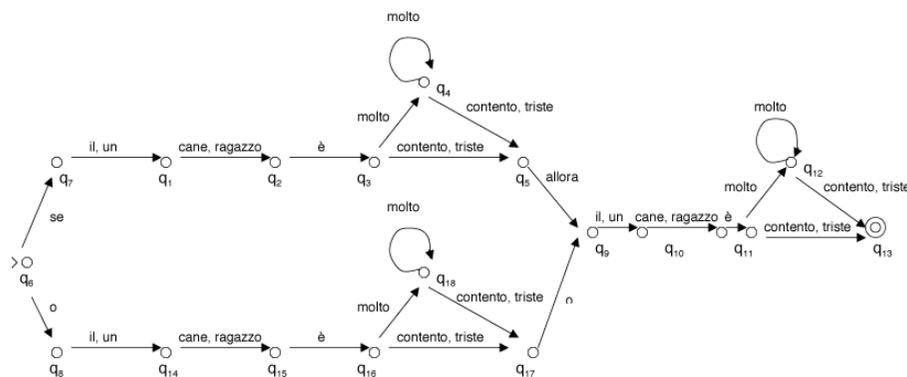
- come per tutti gli automi finiti, ogni transizione del nostro automa è determinata dall'ultima parola (simbolo) che ha letto. Ma con frasi della forma in (3) è cruciale che a ogni occorrenza di *allora* corrisponda un'occorrenza precedente di *se* e non una di *o*, e che a ogni occorrenza di *o* corrisponda un'occorrenza di *o*, e non una di *se*. Il nostro automa è incapace di ricordare se la sequenza è iniziata con *se* oppure con *o* oppure con un articolo. Per questo accetta delle sequenze che non sono frasi dell'italiano.

- (3) a. o il cane è triste o il ragazzo è contento
b. se il cane è triste allora il ragazzo è contento

- (4) a. o il cane è triste
b. o il cane è triste allora il ragazzo è contento
c. se il cane è triste
d. se il cane è triste, o il ragazzo è contento

Secondo tentativo

Per evitare il problema che abbiamo incontrato, proviamo a costruire un automa diverso:



Problema risolto

L'automa precedente accetta correttamente le sequenze in (3) e rifiuta quelle in (4):

- (3) a. o il cane è triste o il ragazzo è contento
b. se il cane è triste allora il ragazzo è contento

- (4) a. o il cane è triste
b. o il cane è triste allora il ragazzo è contento
c. se il cane è triste
d. se il cane è triste, o il ragazzo è contento

Problemi per il secondo tentativo

Il problema con il secondo tentativo è che le frasi della forma *se... , allora...* e *o...o...* possono essere incassate l'una nell'altra come in (5):

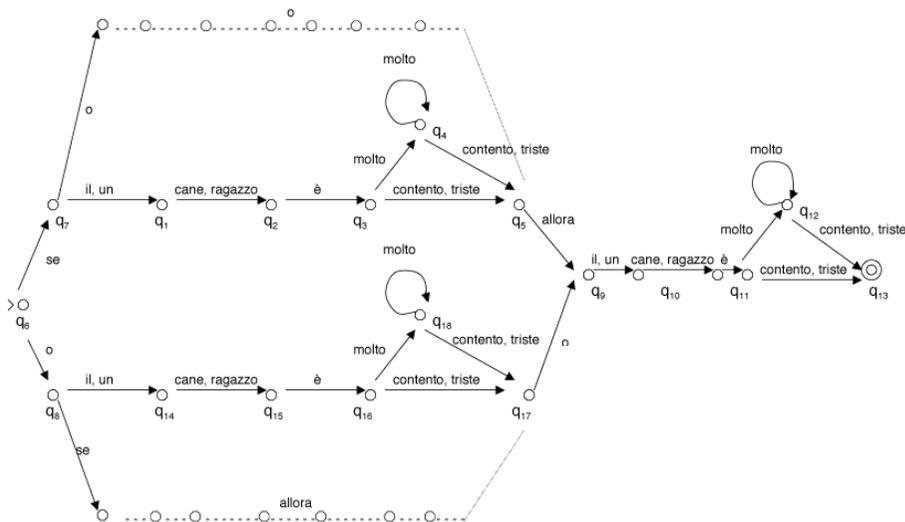
- (5) a. se o il cane è triste o il ragazzo è contento, allora il cane è triste
 b. o se il cane è triste, allora il ragazzo è contento o se il ragazzo è contento, allora il cane è triste.

Il nostro automa non accetta queste frasi.

Una soluzione insoddisfacente

Potremmo cercare di risolvere il problema posto da (5) modificando ulteriormente il diagramma. Per esempio potremmo introdurre due nuove diramazioni dagli stati in cui l'automa si trova dopo il *se* e l'*o* iniziali, per accomodare le nuove frasi della forma in (5) (nella figura mi limito a schizzare queste nuove diramazioni senza completarle).

- (5) a. se il cane è triste o il ragazzo è contento, allora il cane è triste
 b. o se il cane è triste, allora il ragazzo è contento o il ragazzo è contento



Perché è insoddisfacente

Se completiamo opportunamente il diagramma di stato precedente, l'automa accetterà le frasi in (5):

- (5) a. se o il cane è triste o il ragazzo è contento, allora il cane è triste
 b. o se il cane è triste, allora il ragazzo è contento o il ragazzo è contento

Il problema è che, in linea di principio, non c'è limite in italiano al numero di frasi della forma *se... , allora...* e *o...o...* che possono essere incassate l'una nell'altra.

Dunque, se adottiamo la strategia precedente, per riconoscere tutte le frasi di questa forma, l'automa dovrebbe avere un numero infinito di stati. Questo è impossibile: per definizione, un automa finito ha solo un numero finito di stati.

L'italiano e gli automi finiti

Tirando le somme: a quanto pare gli automi finiti hanno delle difficoltà con le frasi italiane della forma *se... , allora... e o... o...*

Il problema è che, con frasi di questa forma, l'automa deve far corrispondere a ogni occorrenza di *allora* una occorrenza di *se* e a ogni occorrenza di *o* una occorrenza di *o*. Queste occorrenze sono a una certa distanza l'una dall'altra (per questa ragione si dice che frasi come queste contengono delle 'dipendenze a lunga distanza') e l'automa non riesce a tenere il conto.

Queste considerazioni suggeriscono che

- l'italiano non è un linguaggio accettato da un automa finito.

Una prova formale, basata su esempi analoghi, che non esiste un automa finito che accetta l'inglese è stata data da Chomsky (1956, 1957).

Lingue naturali e automi finiti

Dal momento che, per quel che si sa, tutte le lingue naturali umane hanno la capacità di esprimere delle 'dipendenze a lunga distanza' del tipo esemplificato dalle frasi della forma *se... , allora... e o... o...*, il risultato precedente viene ritenuto una prova che

- le lingue naturali umane non sono linguaggi accettati da automi finiti.

Una domanda

Gli automi finiti (deterministici o no), a causa del modo in cui sono fatti, hanno dunque certe limitazioni, cioè sono incapaci di riconoscere certi linguaggi.

Per esempio, non sono in grado di accettare linguaggi che contengono 'dipendenze a lunga distanza' come quelle che abbiamo visto nelle frasi italiane della forma *se... , allora... e o... o...*

Esistono altri tipi di automi che possono accettare linguaggi di questo genere?

Altri tipi di automi

Esistono altre classi di automi che sono stati studiati e che sono in grado di accettare linguaggi che esibiscono il tipo di dipendenze a 'lunga distanza' che abbiamo osservato. Per esempio,

- gli automi *pushdown*
- gli automi *linear bounded*
- le macchine di Turing

Questi automi sono in ordine crescente quanto a potere di riconoscimento. Cioè, è possibile dimostrare questo: l'insieme dei linguaggi riconosciuti dagli automi finiti è incluso propriamente nell'insieme dei linguaggi riconosciuti dagli automi *pushdown*, che è incluso propriamente nell'insieme dei linguaggi riconosciuti dagli automi *linear bounded*, che è incluso propriamente nell'insieme dei linguaggi riconosciuti dalle macchine di Turing.

Qui, ci limitiamo tuttavia a considerare gli automi finiti e lasciamo gli altri tipi di automi per un'altra occasione.

Un esercizio

Introduciamo questa notazione: l'espressione ' $(x)^n$ ' sta per una stringa che contiene un numero n di occorrenze della stringa x (se x consiste in un solo simbolo, possiamo tralasciare le parentesi).

Per esempio, ' $(ab)^3$ ' sta per la stringa 'ababab'.

Consideriamo ora il linguaggio che consiste esattamente nell'insieme di tutte le stringhe $(ab)^n$, per ogni numero $n \geq 1$.

Chiaramente, questo linguaggio è infinito:

ab
abab
ababab
abababab
...

Esiste una grammatica regolare (tipo 3) che lo genera?

Grammatiche di tipo 3 (regolari)

Ripasso

Rammentiamo la definizione di grammatica regolare.

Una grammatica di tipo 3 (o regolare) è una grammatica formale le cui regole sono della forma in (a) o della forma in (b):

$$\text{a. } A \rightarrow xB$$

$$\text{b. } A \rightarrow x$$

dove A e B sono simboli non-terminali e x è una stringa arbitraria in V_T^* .

La soluzione

Il linguaggio precedente è generato, ad esempio, da questa grammatica regolare:

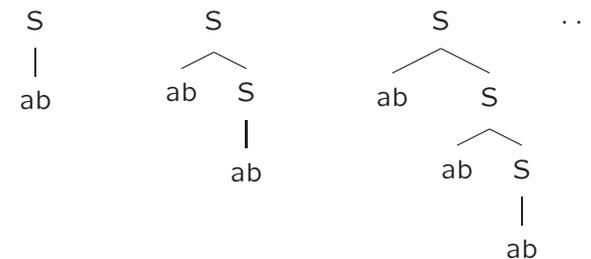
$G_9 = (V_T, V_N, S, R)$, dove

- V_T (l'alfabeto terminale) è l'insieme $\{a, b\}$
- V_N (l'alfabeto non terminale) è l'insieme $\{S\}$
- S è il simbolo iniziale
- R è l'insieme di regole seguente:

$$\text{(i) } S \rightarrow abS$$

$$\text{(ii) } S \rightarrow ab$$

Derivazioni

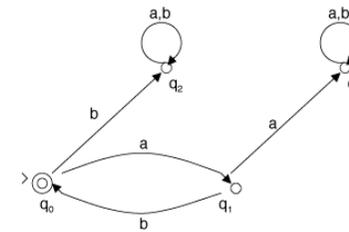


Costruire l'automa

È possibile specificare un automa finito che accetta lo stesso linguaggio generato della grammatica precedente?

Soluzione

Questo automa finito deterministico accetta lo stesso linguaggio generato dalla grammatica precedente:



Un fatto non accidentale

Corrispondenza tra grammatiche regolari e automi finiti

Questo fatto che abbiamo appena osservato, che un linguaggio generato da una grammatica regolare è accettato da un automa finito, non è accidentale. Infatti, è possibile dimostrare che

- un linguaggio è generato da una grammatica regolare se e solo se è accettato da un automa finito.

Una conseguenza

Grammatiche regolari e condizionali

In precedenza, abbiamo visto che gli automi finiti non possono riconoscere i frammenti dell'italiano che contengono frasi della forma *se...*, *allora...* e *o...o...*

Il fatto che le grammatiche regolari generino esattamente gli stessi linguaggi accettati dagli automi finiti ha dunque la conseguenza che le grammatiche regolari non possono generare i frammenti dell'italiano che contengono frasi di questo genere.

Corrispondenza tra grammatiche e automi

È possibile mostrare che a ogni tipo di grammatica che abbiamo introdotto corrisponde un tipo automa che accetta esattamente i linguaggi generati da quel tipo di grammatica.

Gli automi *pushdown* non deterministici accettano esattamente gli stessi linguaggi generati dalle grammatiche *context-free*.

Gli automi *linear bounded* non deterministici accettano esattamente gli stessi linguaggi generati dalle grammatiche *context-sensitive* senza la stringa vuota.

Le macchine di Turing accettano esattamente gli stessi linguaggi generati dalle grammatiche di tipo 0.

Un altro esercizio

Consideriamo ora il linguaggio che consiste esattamente nell'insieme di tutte le stringhe $a^n b^n$, per ogni numero $n \geq 0$.

Chiaramente, questo linguaggio è infinito:

ab
aabb
aaabbb
aaaabbbb
...

Esiste una grammatica regolare (tipo 3) che lo genera?

Una grammatica regolare

Consideriamo la grammatica seguente:

$G_{10} = (V_T, V_N, S, R)$, dove

- V_T (l'alfabeto terminale) è l'insieme $\{a, b\}$
- V_N (l'alfabeto non terminale) è l'insieme $\{S\}$
- S è il simbolo iniziale
- R è l'insieme di regole seguente:

- (i) $S \rightarrow aS$
- (ii) $S \rightarrow bS$
- (iii) $S \rightarrow \epsilon$

Buoni risultati

La grammatica precedente genera correttamente queste stringhe:

$$(6) \quad S \Rightarrow aS \Rightarrow abS \Rightarrow ab$$

$$(7) \quad S \Rightarrow aS \Rightarrow aaS \Rightarrow aabS \Rightarrow aabbS \Rightarrow aabb$$

$$(8) \quad S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaS \Rightarrow aaabS \Rightarrow aaabbS \Rightarrow aaabbbS \Rightarrow aaabbb$$

$$(9) \quad \dots$$

Cattivi risultati

Sfortunatamente, la grammatica precedente genera anche diverse stringhe indesiderate. Ad esempio, genera le stringhe:

$$(10) \quad S \Rightarrow aS \Rightarrow abS \Rightarrow abaS \Rightarrow abaaS \Rightarrow abaa$$

$$(11) \quad S \Rightarrow bS \Rightarrow b$$

Queste stringhe non appartengono al linguaggio che consiste esattamente nell'insieme di tutte le stringhe $a^n b^n$, per ogni numero $n \geq 0$.

Un'altra grammatica regolare

Consideriamo ora la grammatica regolare seguente:

$G_{11} = (V_T, V_N, S, R)$, dove

- V_T (l'alfabeto terminale) è l'insieme $\{a, b\}$
- V_N (l'alfabeto non terminale) è l'insieme $\{S, A\}$
- S è il simbolo iniziale
- R è l'insieme di regole seguente:

$$(i) \quad S \rightarrow aS$$

$$(ii) \quad S \rightarrow aA$$

$$(iii) \quad A \rightarrow bA$$

$$(iv) \quad A \rightarrow \epsilon$$

Buoni risultati

Di nuovo, la grammatica precedente genera correttamente queste stringhe:

$$(12) \quad S \Rightarrow aA \Rightarrow abA \Rightarrow ab$$

$$(13) \quad S \Rightarrow aS \Rightarrow aaA \Rightarrow aabA \Rightarrow aabbA \Rightarrow aabb$$

$$(14) \quad S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaabA \Rightarrow aaabbA \Rightarrow aaabbbA \Rightarrow aaabbbb$$

$$(15) \quad \dots$$

Altri buoni risultati

Inoltre, a differenza della grammatica regolare G_{10} , la grammatica G_{11} non genera le stringhe

abaa
b

Infatti, in base alle regole (i-iv), per introdurre b , dobbiamo avere nella derivazione una stringa che contiene il simbolo non terminale A , ma la regola (ii) che introduce A a partire dal simbolo iniziale S introduce anche a . Dunque, non possiamo avere stringhe che consistono solo di b .

Inoltre, per derivare una stringa che inizia con ab , dobbiamo applicare le regole (ii-iii):

$$(16) \quad S \Rightarrow aA \Rightarrow abA \Rightarrow \dots$$

A questo punto, tuttavia, non abbiamo nessuna regola che ci consenta di introdurre a , quindi non possiamo derivare $abaa$.

Cattivi risultati

Sfortunatamente, la grammatica precedente, oltre a non generare la stringa vuota (che è una stringa del linguaggio che vogliamo generare), genera ancora delle stringhe indesiderate.

Per esempio, genera queste stringhe:

$$(17) \quad S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaabA \Rightarrow aaab$$

$$(18) \quad S \Rightarrow aA \Rightarrow abA \Rightarrow abbA \Rightarrow abb$$

Dunque, anche la grammatica G_{11} , come la grammatica G_{10} , non riesce a far coincidere il numero degli a con il numero dei b nelle stringhe che genera.

Una domanda

A quanto pare, non è affatto ovvio come generare il linguaggio desiderato per mezzo di una grammatica regolare.

Vediamo se riusciamo a fare di meglio con un altro tipo di grammatica.

È possibile scrivere una grammatica *context-free* che genera il linguaggio che consiste esattamente nell'insieme delle stringhe $a^n b^n$, per ogni numero $n \geq 0$?

Grammatiche di tipo 2 (context-free)

Ripasso

Rammentiamo la definizione di grammatica di tipo 2 (o *context-free*). Queste grammatiche hanno regole della forma seguente:

$$A \rightarrow \omega,$$

dove A è un simbolo non terminale e ω è una stringa arbitraria.

Risposta

Una grammatica context-free

Ecco una grammatica *context-free* che genera il linguaggio che consiste esattamente nell'insieme delle stringhe $a^n b^n$, per ogni numero $n \geq 0$.

$G_{12} = (V_T, V_N, S, R)$, dove

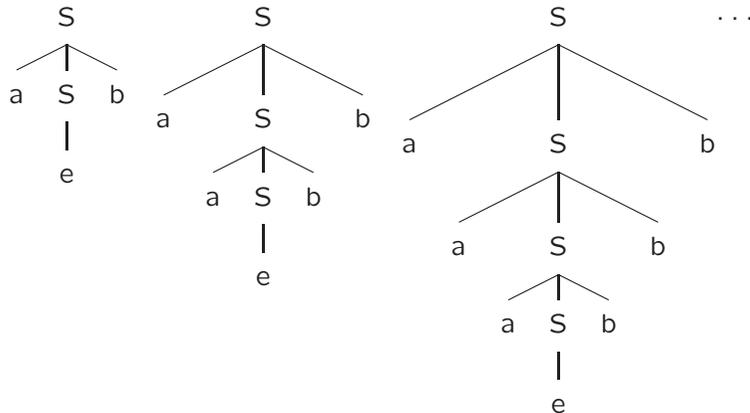
- V_T (l'alfabeto terminale) è l'insieme $\{a, b\}$
- V_N (l'alfabeto non terminale) è l'insieme $\{S\}$
- S è il simbolo iniziale
- R è l'insieme di regole seguente:

$$(i) \quad S \rightarrow aSb$$

$$(ii) \quad S \rightarrow \epsilon$$

Obiettivo raggiunto

È possibile mostrare che la grammatica precedente genera esattamente l'insieme di tutte le stringhe $a^n b^n$ (per ogni numero $n \geq 0$).



Un fatto non accidentale

Linguaggi regolari e linguaggi context free

Il fatto che i nostri tentativi di generare il linguaggio precedente con una grammatica regolare siano andati a vuoto non è casuale.

È possibile dimostrare che il linguaggio in questione *non* può essere generato da una grammatica regolare. Quindi, esistono dei linguaggi generati dalle grammatiche *context-free* che non possono essere generati dalle grammatiche regolari.

D'altra parte, le grammatiche regolari non sono altro che un tipo particolare di grammatica *context-free*. Dunque, ogni linguaggio generato da una grammatica regolare è generato da una grammatica *context-free*.

Ne segue che

- (a) l'insieme dei linguaggi generati dalle grammatiche *context-free* include propriamente l'insieme dei linguaggi generati dalle grammatiche regolari.

La relazione di inclusione

linguaggi context-free



La gerarchia di Chomsky

grammatiche

Anche per i linguaggi generati dagli altri tipi di grammatiche che abbiamo introdotto vale una relazione di inclusione analoga. In particolare, oltre al fatto (a), è possibile provare i fatti (b) e (c):

- (a) l'insieme dei linguaggi generati dalle grammatiche regolari (tipo 3) è incluso propriamente nell'insieme dei linguaggi generati dalle grammatiche *context-free* (tipo 2);
- (b) l'insieme dei linguaggi generati dalle grammatiche *context-free* (tipo 2) che non contengono la stringa vuota è incluso propriamente nell'insieme dei linguaggi generati dalle grammatiche *context-sensitive* (tipo 1);
- (c) l'insieme dei linguaggi generati dalle grammatiche *context-sensitive* (tipo 2) è incluso propriamente nell'insieme dei linguaggi generati dalle grammatiche *non ristrette* (tipo 0);

Questa relazione di inclusione tra i linguaggi generati dai diversi tipi di grammatiche prende il nome di *gerarchia di Chomsky*.

Una riflessione

Si noti che, in un certo senso, il linguaggio che consiste nell'insieme delle stringhe $a^n b^n$ (per ogni numero $n \geq 0$) è simile al caso delle frasi della forma *se... allora... e o... o...*

Nel caso delle frasi della forma *se... allora... e o... o...*, a ogni occorrenza di *allora* deve corrispondere un'occorrenza di *se* che non è immediatamente adiacente ad *allora* e a ogni occorrenza di *o* deve corrispondere un'altra occorrenza di *o* che non è immediatamente adiacente.

Nel caso di $a^n b^n$, ad ogni occorrenza di *b* deve corrispondere un'occorrenza di *a* e queste occorrenze, tranne che per una coppia, non sono immediatamente adiacenti.

Le grammatiche regolari e gli automi finiti, come abbiamo visto, non sono in grado di riconoscere correttamente le frasi della forma *se... allora... e o... o...*. Dal momento che identificano esattamente la stessa classe di linguaggi, non è sorprendente che siano incapaci di identificare altri linguaggi con 'dipendenze a lunga distanza' dello stesso tipo, come l'insieme delle stringhe $a^n b^n$ (per ogni $n \geq 0$).

Una domanda

Abbiamo visto invece che le grammatiche *context-free* riescono a generare il linguaggio che consiste nell'insieme delle stringhe $a^n b^n$ (per ogni numero $n \geq 0$).

Se la riflessione precedente è corretta, questo suggerisce che le grammatiche *context-free* dovrebbero anche essere in grado di generare il linguaggio che consiste in tutte e solo le frasi dell'italiano della forma *se... allora... e o... o...* (basate sul vocabolario limitato che abbiamo considerato).

È così?

Una grammatica context-free

$G_{12} = (V_T, V_N, S, R)$, dove

- V_T (l'alfabeto terminale) è l'insieme {contento, triste, ragazzo, cane, il, un, è}
- V_N (l'alfabeto non terminale) è l'insieme {S, NP, AP, VP, N, A, V}
- S è il simbolo iniziale
- R è l'insieme di regole seguente:

S → se S allora S

S → o S o S

S → NP VP

VP → V AP

NP → D N

AP → A

A → contento

A → triste

N → ragazzo

N → cane

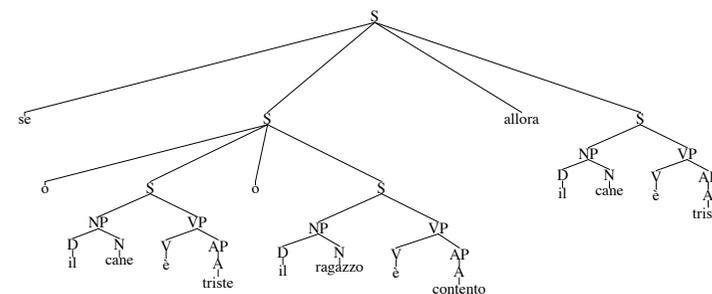
D → il

D → un

V → è

Grammatiche context-free e dipendenze a distanza

La grammatica precedente non ha difficoltà a generare frasi della forma *se... allora...* e frasi della forma *o... o...* incassate l'una nell'altra:



Grammatiche regolari, context-free e linguaggi naturali

Il fatto che le grammatiche *context free*, a differenza delle grammatiche regolari, generino correttamente frasi della forma *se... allora...* e *o...o...* ci permette di trarre una conclusione importante per quanto riguarda le grammatiche delle lingue naturali umane:

- le grammatiche delle lingue naturali umane non sono grammatiche regolari; le grammatiche delle lingue naturali umane devono avere almeno il potere generativo di una grammatica *context-free*.

Riassumendo

- Oltre alle grammatiche formali, abbiamo introdotto un altro dispositivo per distinguere tra stringhe che sono frasi di una lingua e stringhe che non lo sono: gli automi.
- Ci siamo concentrati sugli automi finiti, deterministici e non.
- Abbiamo osservato che esiste una relazione tra automi finiti e grammatiche regolari: un linguaggio è accettato da un automa finito se e solo se è generato da una grammatica regolare. Abbiamo visto un esempio di questo fatto.
- Abbiamo visto che gli automi finiti e le grammatiche regolari non possono generare certi linguaggi che sono invece generati dalle grammatiche *context-free*.
- Abbiamo visto che le lingue naturali umane non possono essere generate da grammatiche regolari (o accettate da automi finiti) e richiedono una grammatica che abbia almeno il potere generativo di una grammatica *context-free*.