Linguaggi formali e compilazione

Corso di Laurea in Informatica

A.A. 2012/2013

Linguaggi formali e compilazione

Analisi sintattica

Grammatiche libere

Scopo del parsing

- L'obiettivo della fase di parsing è innanzitutto di stabilire se una sequenza di token rappresenta una "frase" corretta del linguaggio e, nel caso, descriverne la struttura.
- Sulla base di che cosa possiamo stabilire, ad esempio, che

  ```
  while (a>0) {a=a-1}
  ```

  è una frase corretta in C/C++, mentre

  ```
  while (a>0) a=a-1}
  ```

  è una frase sintatticamente errata?
- La risposta (anche se solo parziale) è che una frase è corretta se e solo se è conforme alla *sintassi* del linguaggio.
- Il formalismo che si è imposto per la descrizione della sintassi dei linguaggi di programmazione è quello delle *grammatiche libere* (da contesto), in inglese *context-free grammar*.

Un pezzo di grammatica del C


```
iteration_statement
    : while '(' expression ')' statement
    | do statement while '(' expression ')' ';'
    | for '(' expression_statement expression_statement ')' statement
    | for '(' expression_statement expression_statement expression ')' statement
    ;
statement
    : labeled_statement
    | compound_statement
    | expression_statement
    | selection_statement
    | iteration_statement
    | jump_statement
    ;
compound_statement
    : '{' '}'
    | '{' statement_list '}'
    | '{' declaration_list '}'
    | '{' declaration_list statement_list '}'
    ;
```

Grammatiche formali

- Come le espressioni regolari, anche le *grammatiche formali* (da qui in avanti semplicemente *grammatiche*), sono uno strumento per la descrizione di linguaggi.
- Una grammatica è un formalismo *generativo* perché il linguaggio da essa definito coincide con l'insieme delle stringhe che possono essere "generate" usando determinate regole che fanno parte della grammatica stessa.
- Le grammatiche possono avere diversi *gradi di espressività*, e dunque definire linguaggi più o meno ricchi.
- Esiste però un forte compromesso fra espressività e possibilità di riconoscimento automatico, che vedremo ben rappresentato nel caso caso dei linguaggi di programmazione.

Definizione formale di grammatica

- Diamo ora la definizione generale di grammatica (formale).
- Una grammatica G è una quadrupla di elementi:

$$G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S}),$$

  dove
  - $\mathcal{N}$ è un insieme di simboli, detti *non terminali*;
  - $\mathcal{T}$ è un insieme di simboli *terminali*, $\mathcal{N} \cap \mathcal{T} = \Phi$;
  - $\mathcal{P}$ è l'insieme delle *produzioni*, cioè scritture della forma $X \rightarrow Y$, dove $X, Y \in (\mathcal{N} \cup \mathcal{T})^*$;
  - $\mathcal{S} \in \mathcal{N}$ è il *simbolo iniziale* (o *assioma*).
- Conviene anche definire l'insieme $\mathcal{V} = \mathcal{N} \cup \mathcal{T}$ come il *vocabolario* della grammatica.

Le produzioni

- La forma delle produzioni è ciò che caratterizza propriamente il "tipo" di grammatica, cioè la sua capacità espressiva.
- Se le produzioni sono del tipo: $A \rightarrow xB$ oppure $A \rightarrow x$, dove $x \in T$ e $A, B \in \mathcal{N}$, la grammatica è detta *lineare destra*.
- Se invece le produzioni sono del tipo: $A \rightarrow Bx$ oppure $A \rightarrow x$, dove $x \in T$ e $A, B \in \mathcal{N}$, la grammatica è detta *lineare sinistra*.
- Una *grammatica regolare* è una grammatica lineare (destra o sinistra).
- Il nome non è casuale. Infatti grammatiche regolari descrivono proprio i linguaggi regolari che già conosciamo.

Le produzioni

- Per la definizione della sintassi dei linguaggi di programmazione, hanno invece particolare importanza i cosiddetti *linguaggi liberi da contesto* (o più semplicemente *linguaggi liberi*).
- I linguaggi liberi sono generabili da grammatiche (dette anch'esse *libere*) in cui le produzioni hanno la seguente forma generale

  $$A \to X$$

  dove $A \in \mathcal{N}$ e $X \in \mathcal{V}^*$, cioè in cui la parte sinistra è un qualunque simbolo non terminale mentre la parte destra è una qualunque stringa di terminali o non terminali.

Derivazioni

- Il meccanismo in base al quale le grammatiche "generano" linguaggi è quello delle derivazioni.
- Una *derivazione* è il processo mediante il quale, a partire dall'assioma ed applicando una sequenza di produzioni, si ottiene una stringa di $\mathcal{T}^*$, cioè una stringa composta da soli terminali.
- Le produzioni rappresentano infatti vere e proprie *regole di riscrittura*.
- Ad esempio, una produzione del tipo

$$E \rightarrow E{+}E$$

si può leggere nel seguente modo: il simbolo E può essere "riscritto" come $E{+}E$.

Derivazioni (continua)

- L'idea è che una grammatica descriva (generi) il linguaggio costituito proprio dalle sequenze di simboli terminali derivabili a partire dall'assioma $\mathcal{S}$.
- Consideriamo, ad esempio, la grammatica $G_5 = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S})$ così definita:
  - $\mathcal{N} = \{S, A, B\}$;
  - $\mathcal{T} = \{\mathrm{a}, \mathrm{b}\}$;
  - $\mathcal{S} = S$;
  - $\mathcal{P}$ contiene le seguenti produzioni:

$$\begin{array}{lcl} S & \rightarrow & \epsilon, \quad S \rightarrow A \\ A & \rightarrow & \mathrm{a}, \quad A \rightarrow \mathrm{a}A, \quad A \rightarrow B \\ B & \rightarrow & \mathrm{b}B, \quad B \rightarrow \mathrm{b} \end{array}$$

- Nel linguaggio generato da G_5 è inclusa la stringa ab perché:

$$S \Rightarrow A \Rightarrow \mathrm{a}A \Rightarrow \mathrm{a}B \Rightarrow \mathrm{ab}.$$

Nota sulla notazione...

- La scrittura $\alpha \Rightarrow \beta$ (dove $\alpha, \beta \in \mathcal{V}^*$) indica che β può essere ottenuta direttamente da α mediante l'applicazione di una produzione della grammatica.
- Ad esempio, con riferimento alla derivazione del lucido precedente, $aA \Rightarrow aB$ perché nella grammatica G_5 è presente la produzione $A \rightarrow B$.
- Non sarebbe stato corretto scrivere $aA \rightarrow aB$ (perché non esiste una tale produzione).
- Se α deriva β mediante l'applicazione di 0 o più produzioni si scrive $\alpha \overset{*}{\Rightarrow}_G \beta$.
- Ad esempio, in G_5, $aA \overset{*}{\Rightarrow}_G ab$.

Descrizione succinta di una grammatica

- Una grammatica può essere espressa in modo più succinto elencando le sole produzioni, qualora si convenga che le prime produzioni in elenco siano quelle relative all'assioma.
- Ad esempio, scrivendo

$$\begin{array}{lcl} E & \rightarrow & E+E \\ E & \rightarrow & E*E \\ E & \rightarrow & (E) \\ E & \rightarrow & \mathtt{id} \end{array}$$

  intendiamo la grammatica $G_1 = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S})$ in cui:
  - $\mathcal{N} = \{E\}$;
  - $\mathcal{T} = \{\mathtt{id}, +, *, (,)\}$;
  - $\mathcal{S} = E$;

  e le produzioni sono ovviamente quelle indicate.

Altri esempi di derivazione

Consideriamo la grammatica G_1 appena introdotta. Allora:

- $E{+}E \Rightarrow_{G_1} \mathtt{id}{+}E$ tramite l'applicazione della produzione $E \rightarrow \mathtt{id}$ alla prima occorrenza di E.
- $E{+}E \overset{*}{\Rightarrow}_{G_1} \mathtt{id} + \mathtt{id}$ tramite l'applicazione della produzione $E \rightarrow \mathtt{id}$ ad entrambe le occorrenze di E.
- $E \overset{*}{\Rightarrow}_{G_1} \mathtt{id} + (E)$ in quanto $E \Rightarrow_{G_1} E{+}E \Rightarrow_{G_1} E{+}(E) \Rightarrow_{G_1} \mathtt{id} + (E)$.
- $E \overset{*}{\Rightarrow}_{G_1} \mathtt{id} + (\mathtt{id})$, in quanto $E \Rightarrow_{G_1} E{+}E \Rightarrow_{G_1} E{+}(E) \Rightarrow_{G_1} \mathtt{id} + (E) \Rightarrow_{G_1} \mathtt{id} + (\mathtt{id})$.
- Una derivazione alternativa per $\mathtt{id} + (\mathtt{id})$ è $E \Rightarrow_{G_1} E{+}E \Rightarrow_{G_1} \mathtt{id}{+}E \Rightarrow_{G_1} \mathtt{id} + (E) \Rightarrow_{G_1} \mathtt{id} + (\mathtt{id})$.

Descrizione succinta e metasimboli

- Possiamo economizzare ancora sulla descrizione di una grammatica “fondendo” produzioni che hanno la stessa parte sinistra.
- È consuetudine, infatti, usare la scrittura $X \rightarrow Y|Z$ al posto di $X \rightarrow Y$ e $X \rightarrow Z$.
- Usando anche questa convenzione, la grammatica G_5 può essere descritta nel seguente modo compatto:

$$\begin{array}{lcl} S & \rightarrow & \epsilon \mid A \\ A & \rightarrow & \mathrm{a} \mid \mathrm{a}A \mid B \\ B & \rightarrow & \mathrm{b}B \mid \mathrm{b} \end{array}$$

- Si noti come nella descrizione di una grammatica si utilizzino simboli che non sono terminali ne’ non terminali, come ad esempio $\rightarrow$ e $\mid$.
- Tali simboli prendono il nome di *metasimboli*.

Frasi e linguaggio generato

Sia $G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S})$ una grammatica.

- Si chiama *forma di frase* di G una qualunque stringa α di $\mathcal{V}^*$ tale che $\mathcal{S} \stackrel{*}{\Rightarrow}_G \alpha$.
- Se $\alpha \in \mathcal{T}^*$ allora si dice che α è anche una *frase* di G.
- Dagli esempi precedenti possiamo conlcudere (ad esempio) che:
  - le stringhe $\mathtt{id} + (E)$ e $\mathtt{id} + (\mathtt{id})$ sono forme di frase di G_1;
  - le stringhe $\mathtt{ab}B$, $\mathtt{abb}B$ e $\mathtt{ab}$ sono forme di frase di G_5;
  - $\mathtt{id} + (\mathtt{id})$ e $\mathtt{ab}$ sono anche frasi;
  - $\mathtt{ba}A$ non è una forma di frase di G_5.
- Il linguaggio generato da G, spesso indicato con $L(G)$, è l'insieme delle frasi di G.

Esempi

- La grammatica G_5 genera il linguaggio $L_5 = \{a^n b^m | n, m \geq 0\}$.
- La grammatica G_1 genera il linguaggio delle espressioni aritmetiche composte da `+`, `*`, `(`, `)` e `id`.
- Le stringhe più corte in $L(G_1)$ sono $\texttt{id}, \texttt{id} + \texttt{id}, \texttt{id} * \texttt{id}, (\texttt{id})$.

Linguaggio generato da una grammatica

- Per dimostrare formalmente che una data grammatica G genera un dato linguaggio L (cioè per provare che $L = L(G)$) si procede solitamente per induzione.
- Dapprima si formula una congettura sulla forma delle frasi di $L(G)$, provando alcune semplici derivazioni.
- Dopodiché si dimostra separatamente che:
  - se X è generata dal linguaggio, allora X ha la particolare forma congetturata;
  - se X è una stringa con quella particolare forma, allora esiste una derivazione in grado di generarla.
- Il procedimento può essere (relativamente) complesso anche per grammatiche molto semplici.

Un esempio

- Dimostriamo formalmente che la grammatica G_5 genera il linguaggio $L_5 = \mathrm{a}^*\mathrm{b}^*$.
- Ricordiamo che G_5 è:

$$
\begin{array}{rcl}
S & \rightarrow & \epsilon \mid A \\
A & \rightarrow & \mathrm{a} \mid \mathrm{a}A \mid B \\
B & \rightarrow & \mathrm{b}B \mid \mathrm{b}
\end{array}
$$

- G_5 genera "solo" stringhe del tipo $\mathrm{a}^n\mathrm{b}^m$.
    - $S \Rightarrow \epsilon$
    - $S \Rightarrow A \stackrel{k}{\Rightarrow} \mathrm{a}^k A \Rightarrow \mathrm{a}^{k+1}, \qquad k \geq 0$
    - $S \Rightarrow A \stackrel{k}{\Rightarrow} \mathrm{a}^k A \Rightarrow \mathrm{a}^k B \stackrel{h}{\Rightarrow} \mathrm{a}^k \mathrm{b}^h B \Rightarrow \mathrm{a}^k \mathrm{b}^{h+1}$, $h, k \geq 0$
- G_5 genera "tutte" le stringhe del tipo $\mathrm{a}^n\mathrm{b}^m$.
    - Segue dall'arbitrarietà di k e h sopra.

Gerarchia di Chomsky

- La seguente tabella descrive la gerarchia di grammatiche, ad ognuna delle quali viene associato l'automa riconoscitore e la classe di linguaggi corrispondente.
- La progressione è dalla grammatica meno espressiva (tipo 3) a quella più espressiva (tipo 0).

Tipo	**Grammatica**	**Automa**	**Linguaggio**
3	Regolare	Automa finito	Regolare
2	Libera	Automa a pila	Libero
1	Dipendente dal contesto	Automa limitato linearmente	Dipendente dal contesto
0	Ricorsiva	Macchina di Turing	Ricorsivamente enumerabile

Tipi dei linguaggi

- Si può dimostrare che se un linguaggio è generabile da una grammatica lineare (tipo 3) allora è regolare.
- Se invece un linguaggio L è generato da una grammatica G di tipo i ($i = 0, .., 2$), allora L è "al più" di tipo i.
- Infatti L potrebbe essere generabile anche da una grammatica più semplice (di tipo $i + 1$).
- Il linguaggio L_5 è regolare perché generato da una grammatica lineare, come abbiamo appena dimostrato.
- Il linguaggio $L(G_1)$ è invece al più libero perché generabile da una grammatica libera.

Tipi dei linguaggi

- Posto $\Sigma = \{a, b, c\}$, Il linguaggio su Σ^* costituito dalle stringhe in cui ogni a è seguita una b è al più libero perché generato da

$$\begin{aligned} S &\rightarrow \epsilon \mid R \\ R &\rightarrow b \mid c \mid ab \mid RR \end{aligned}$$

- In realtà L è regolare, in quanto definibile anche mediante l'espressione regolare $(\mathbf{b} + \mathbf{c} + \mathbf{ab})^*$.
- Il linguaggio L_{11} su $\{a, b\}$ costituito da tutte le stringhe α palindrome (cioè tali che $\alpha^R = \alpha$) è libero in quanto generabile dalla grammatica

$$S \rightarrow aSa, S \rightarrow bSb, S \rightarrow a, S \rightarrow b, S \rightarrow \epsilon$$

Si può poi dimostrare che L_{11} non è regolare.

Un esempio più complesso: le e.r. come linguaggio libero

- Come sappiamo, le espressioni regolari sono un formalismo per definire linguaggi.
- Ad esempio, l'espressione regolare $\mathbf{0}(\mathbf{0}+\mathbf{1})^*$, sull'alfabeto $\mathcal{B}$, definisce il linguaggio delle stringhe binarie che iniziano con 0.
- Per essere "manipolabili" automaticamente (ad esempio, per passare da un'espressione regolare al corrispondente automa nondeterministico), le espressioni regolari devono essere riconosciute come *ben formate*.
- Ad esempio, l'espressione $\mathbf{0}(\mathbf{0}+^*\mathbf{1}$ è ben formata? Potremmo procedere alla costruzione dell'automa?

Un esempio più complesso: le e.r. come linguaggio libero

- Stabilire quali sono le espressioni regolari ben formate su, ad esempio, l'alfabeto $\mathcal{B}$ non può essere fatto usando lo stesso formalismo delle espressioni regolari.
- Cioè non può esistere un'espressione regolare sull'alfabeto $\{\mathbf{0}, \mathbf{1}, (,), +, *, \epsilon\}$ che comprenda tutte e sole le espressioni regolari su $\mathcal{B}$.
- Questa cosa si può fare invece usando una grammatica libera!

Un esempio più complesso: le e.r. come linguaggio libero

- La grammatica G_2 così definita

$$
\begin{array}{lcl}
E & \rightarrow & T \mid T+E \\
T & \rightarrow & F \mid FT \\
F & \rightarrow & (E) \mid (E)^* \mid A \mid A^*, \\
A & \rightarrow & \mathbf{0} \mid \mathbf{1}
\end{array}
$$

genera tutte le stringhe sull'alfabeto $\mathbf{0}, \mathbf{1}, (,), +, *, \epsilon$ che sono espressioni regolari ben formate sull'alfabeto $\mathcal{B}$.

Esempio (continua)

- La stringa (epressione regolare) $\mathtt{0(0+1)}^*$ appartiene a $L(G_2)$ in quanto esiste la derivazione:

$$
\begin{array}{lll}
E \Rightarrow_{G_2} & T & \\
\Rightarrow_{G_2} & FT & \text{Usando } T \to FT \\
\Rightarrow_{G_2} & FF & \text{Usando } T \to F \\
\Rightarrow_{G_2} & AF & \text{Usando } F \to A \\
\Rightarrow_{G_2} & A(E)^* & \text{Usando } F \to (E)^* \\
\Rightarrow_{G_2} & \mathtt{0}(E)^* & \text{Usando } A \to \mathtt{0} \\
\Rightarrow_{G_2} & \mathtt{0}(T+E)^* & \text{Usando } E \to T+E \\
\Rightarrow_{G_2} & \mathtt{0}(T+T)^* & \text{Usando } E \to T \\
\Rightarrow_{G_2} & \mathtt{0}(F+T)^* & \text{Usando } T \to F \\
\Rightarrow_{G_2} & \mathtt{0}(F+F)^* & \text{Usando } T \to F \\
\Rightarrow_{G_2} & \mathtt{0}(A+F)^* & \text{Usando } F \to A \\
\Rightarrow_{G_2} & \mathtt{0}(A+A)^* & \text{Usando } F \to A \\
\Rightarrow_{G_2} & \mathtt{0}(\mathtt{0}+A)^* & \text{Usando } A \to \mathtt{0} \\
\Rightarrow_{G_2} & \mathtt{0}(\mathtt{0}+\mathtt{1})^* & \text{Usando } A \to \mathtt{1}
\end{array}
$$

Backus-Naur Form (BNF)

- Nella descrizione della sintassi dei linguaggi di programmazione, i simboli non terminali, detti anche *variabili sintattiche*, vengono spesso rappresentati mediante un identificatore descrittivo racchiuso fra parentesi angolate.
- Esempio

$$
\begin{aligned}
\langle \text{comando if} \rangle \quad \rightarrow \quad & \text{if } \langle \text{espressione booleana} \rangle \text{ then} \\
& \qquad \langle \text{lista di comandi} \rangle \\
& \text{endif } \mid \\
& \text{if } \langle \text{espressione booleana} \rangle \text{ then} \\
& \qquad \langle \text{lista di comandi} \rangle \\
& \text{else} \\
& \qquad \langle \text{lista di comandi} \rangle \\
& \text{endif}
\end{aligned}
$$

Altri esempi

- Usando la BNF la grammatica G_1 verrebbe descritta dalle seguenti produzioni

$$
\begin{array}{rcl}
\langle\text{espressione}\rangle & \rightarrow & \langle\text{espressione}\rangle + \langle\text{espressione}\rangle \mid \\
\langle\text{espressione}\rangle & \rightarrow & \langle\text{espressione}\rangle * \langle\text{espressione}\rangle \mid \\
\langle\text{espressione}\rangle & \rightarrow & (\langle\text{espressione}\rangle) \mid \text{id}
\end{array}
$$

- Una grammatica per le chiamate di procedura in Java

$$
\begin{array}{rcl}
\langle\text{chiamata}\rangle & \rightarrow & \text{id}(\langle\text{parametri-opzionali}\rangle) \\
\langle\text{parametri-opzionali}\rangle & \rightarrow & \langle\text{lista-di-parametri}\rangle \mid \epsilon \\
\langle\text{lista-di-parametri}\rangle & \rightarrow & \langle\text{lista-di-parametri}\rangle, \langle\text{parametro}\rangle \mid \\
 & & \langle\text{parametro}\rangle
\end{array}
$$

Altre convenzioni

- Elementi opzionali possono essere inclusi fra i meta simboli [e].
- Ad esempio, potremo usare la scrittura

$$\begin{aligned}\langle\text{comando if}\rangle \quad \rightarrow \quad & \text{if } \langle\text{espressione booleana}\rangle \text{ then} \\ & \quad \langle\text{lista di comandi}\rangle \\ & [\,\text{else} \\ & \quad \langle\text{lista di comandi}\rangle\,] \\ & \text{endif}\end{aligned}$$

- Elementi ripetitivi possono essere inclusi fra i metasimboli { e }.
- Ad esempio

$$\langle\text{list di comandi}\rangle \quad \rightarrow \quad \langle\text{comando}\rangle \,\{\, ;\ \langle\text{comando}\rangle \,\}$$

Altre convenzioni (continua)

- Più recentemente, nella BNF i simboli terminali vengono scritti in grassetto.
- In questo modo diventa possibile sopprimere l'uso delle parentesi angolate intorno alle variabili sintattiche, migliorando la leggibilità complessiva. Le variabili sintattiche continuano ad essere scritte in corsivo.
- Ad esempio, potremo scrivere

$$
\begin{array}{lll}
\textit{comando if} & \rightarrow & \textbf{if}\ \textit{espressione booleana}\ \textbf{then} \\
 & & \quad \textit{lista di comandi} \\
 & & [\ \textbf{else} \\
 & & \quad \textit{lista di comandi}\ ] \\
 & & \textbf{endif}
\end{array}
$$

Altre convenzioni (continua)

- Nel caso in cui possano sorgere ambiguità, i simboli terminali vengono racchiusi fra doppi apici.
- Un esempio è costituita dal caso di simboli terminali coincidenti con qualche metasimbolo.
- Esempio (tratto dalla sintassi del C):

$$\textit{comando composto} \rightarrow {}''\{'' \, \{ \, \textit{dichiarazione} \, \} \, \{ \, \textit{comando} \, \} \, {}''\}''$$

Esercizi proposti

- Fornire una grammatica libera per l'insieme delle stringhe costituite da parentesi correttamente bilanciate (ad esempio, ()(()) e (()) devono far parte del linguaggio, mentre ())(non deve farne parte).
- Fornire una grammatica libera per il linguaggio $L_{12} = \{\mathrm{a}^n \mathrm{b}^{2n} | n \geq 0\}$ sull'alfabeto $\{\mathrm{a}, \mathrm{b}\}$.
- Si consideri la seguente grammatica G_I

$$
\begin{array}{rcl}
S & \rightarrow & I \mid A \\
I & \rightarrow & \textbf{if } B \textbf{ then } S \mid \textbf{if } B \textbf{ then } S \textbf{ else } S \\
A & \rightarrow & \mathrm{a} \\
B & \rightarrow & \mathrm{b}
\end{array}
$$

e si fornisca una derivazione per la stringa

if b **then** **if** b **then** a **else** a