Grammatiche libere estese con espressioni regolari

Prof. A. Morzenti

aa 2008-2009

GRAMMATICHE LIBERE ESTESE CON ESPRESSIONI REGOLARI

MAGGIORE LEGGIBILITÀ DELLE ESPRESSIONI REGOLARI grazie alla semplicita` degli operatori di iterazione (stella e croce) e di scelta (unione).

Per migliorare la concisione e la leggibilita` delle grammatiche si usano espressioni regolari nella parte destra delle regole. Si creano cosi` le

GRAMMATICHE LIBERE ESTESE (Extended BNF o EBNF) spesso preferite nei manuali dei linguaggi tecnici.

Le regole di EBNF consentono la creazione di DIAGRAMMI SINTATTICI che possono essere pensati come SCHEMI DI FLUSSO degli algoritmi di analisi sintattica.

NB: La famiglia LIB è chiusa rispetto alle operazioni regolari, quindi la capacità generativa delle grammatiche estese è uguale a quella delle grammatiche libere.

ESEMPIO: lista di dichiarazioni di variabili

character testo1, testo2; *real* temp, result;

integer alfa, beta2, gamma;

$\Sigma = \{c, i, r, v, \ ',' \ , \ ';'\}$ dove v è il nome di una variabile

$\left(\left(c \mid i \mid r \right) v (, v)^{*} ; \right)^{+}$

$S \rightarrow SE \mid E \quad E \rightarrow AF; \quad A \rightarrow c \mid i \mid r \quad F \rightarrow v, F \mid v$

La grammatica è più lunga e toglie evidenza all'esistenza di due liste gerarchiche.
C'è inoltre arbitrarietà nella scelta dei metasimboli *(A, E, F)*, il cui nome e` irrilevante.

UNA GRAMMATICA LIBERA ESTESA (EBNF) $G = \{ V, \Sigma, P, S \}$ contiene esattamente $|V|$ regole, ognuna nella forma $A \rightarrow \eta$, dove η è una e.r. di alfabeto $V \cup \Sigma$. Per maggiore leggibilità si possono utilizzare anche gli altri operatori derivati (croce, elev. a potenza, opzionalità).

ESEMPIO: Linguaggio Simil-Algol
B: blocco; D; dichiarazione;
I=parteImperativa; F=frase;
a=assegnamento; c=char; i=int;
r=real; v=variabile; b=begin; e=end

$$B \rightarrow b[D]Ie$$
$$D \rightarrow ((c \mid i \mid r)v(,v)^*;)^+$$
$$I \rightarrow F(;F)^*$$
$$F \rightarrow a \mid B$$

Versioni più compatte, forse meno leggibili. B non eliminabile: i blocchi sono annidati

Sostituiti D e I con loro def.

$$B \rightarrow b\left[((c \mid i \mid r)v(,v)^*;)^+\right]F(;F)^*e$$

Semplificato eliminando [ ]

$$B \rightarrow b((c \mid i \mid r)v(,v)^*;)^*F(;F)^*e$$

eliminato F

$$B \rightarrow b((c \mid i \mid r)v(,v)^*;)^*(a \mid B)(;(a \mid B))^*e$$

una sola regola!

DERIVAZIONI E ALBERI NELLE GRAMMATICHE LIBERE ESTESE

Derivazione definita come se la parte destra di una regola estesa fosse una e.r. che definisce un insieme in generale illimitato di stringhe.
Oppure come se la grammatica G fosse equivalente a G' con un numero illimitato di regole.

$$A \to (aB)^+$$

$$A \to aB \mid aBaB \mid ....$$

RELAZIONE DI DERIVAZIONE PER LE GRAMMATICHE ESTESE:

Date le stringhe η_1 e $\eta_2 \in (\Sigma \bigcup V)^*$

si dice che η_2 *deriva* (immediatamente) da η_1

$\eta_1 \Rightarrow \eta_2$, se le due stringhe si fattorizzano come:

$\eta_1 = \alpha A \gamma$, $\eta_2 = \alpha \vartheta \gamma$ ed esiste una regola

$A \to e$: $e \overset{*}{\Rightarrow} \vartheta$

Si noti che η_1 e η_2 (e $\alpha A \gamma$ e $\alpha \vartheta \gamma$) non contengono operatori delle e.r. né parentesi. La stringa *e* è una e.r

NB: nell'albero e nelle forme di frase generate non compare mai alcuna e.r.

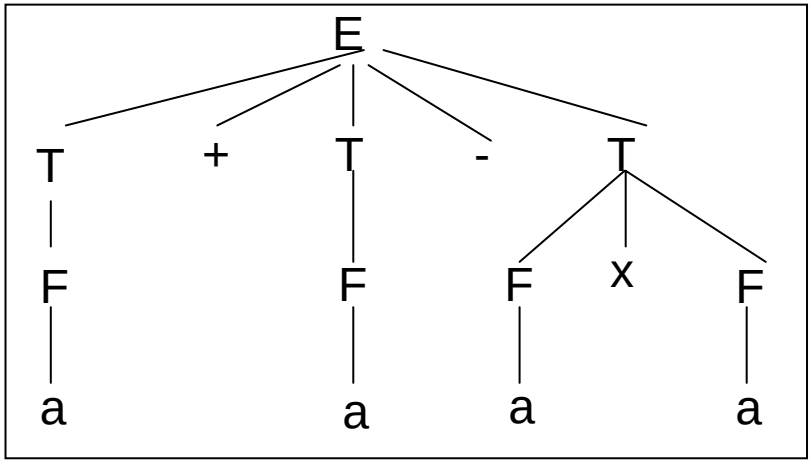
ESEMPIO: Derivazione estesa per espressioni aritmetiche
La grammatica estesa G genera le espressioni aritmetiche con i quattro operatori infissi, le parentesi tonde e la variabile *a*.

E → [+ | -] T ((+ | -) T)* T → F ((× | /) F)* F → (a | ’(‘ E ’)’)

derivazione sinistra:

$$E \Rightarrow T + T - T \Rightarrow F + T - T \Rightarrow a + T - T \Rightarrow a + F - T \Rightarrow$$

$$\Rightarrow a + a - T \Rightarrow a + a - F \times F \Rightarrow a + a - a \times F \Rightarrow a + a - a \times a$$



Per una EBNF il grado di un nodo risulta in generale illimitato.
Essendo l’albero piu` largo esso ha in generale profondità ridotta.

AMBIGUITÀ NELLE GRAMMATICHE ESTESE

Una grammatica ambigua non estesa è ambigua anche in forma estesa, ma l'ambiguità può sorgere in modo peculiare dalle e.r.

$a^*b \mid ab^*$ numerata $a_1^*b_2 \mid a_3b_4^*$

è ambigua perché ab è

derivabile come a_1b_2 o come a_3b_4

$S \to a^*b \mid ab^*$ è ambigua

GRAMMATICHE PIÙ GENERALI – CLASSIFICAZIONE DI CHOMSKY

Grammatica	*Forma delle regole*	*Fam. del ling*	*Tipo di riconoscitore*
Tipo 0	$\beta \to \alpha$ dove $\alpha,\beta \in (\Sigma \cup V)^+$ e $\beta \neq \varepsilon$	Linguaggi ricorsivamente Enumerabili.	Macchina di Turing.
Tipo 1 (contestuale, dip. dal contesto, context-sensitive)	$\beta \to \alpha$ dove $\alpha,\beta \in (\Sigma \cup V)^+$ e $\lvert\beta\rvert \leq \lvert\alpha\rvert$	Linguaggi contestuali (dipendenti dal contesto).	Macchina di Turing con nastro di lunghezza uguale a quella della stringa da riconoscere.

CLASSIFICAZIONE DI CHOMSKY (continua)

Grammatica	*Forma delle regole*	*Fam. del ling*	*Tipo di riconoscitore*
Tipo 2 o libera da contesto o non contestuale o BNF (context-free).	$A \to \alpha$ dove A è un nt e $\alpha \in (\Sigma \bigcup V)^*$	Linguaggi liberi (dal contesto) LIB o non contestuali o algebrici.	Automa con memoria a pila (non deterministico).
Tipo 3 o unilineare (lineare a destra o a sinistra).	Lin a dx: $A \to uB$ Lin a sn: $A \to Bu$ dove A è nt, $u \in \Sigma^*$ e $B \in (V \bigcup \varepsilon)$	Linguaggi regolari, Razionali, REG, a stati finiti	Automa finito, a stati finiti, macchina sequenziale

Le famiglie di linguaggi sono in relazione di contenimento stretto dall'alto verso il basso della tabella precedente.

Differenze: forma delle regole e proprietà degli automi riconoscitori

tipo 0, 1 e 2: automi a memoria illimitata

tipo 3: automi a memoria limitata

CHIUSURA RISPETTO A:
UNIONE, CONC., STELLA, INV. SPECULARE, INTERS. CON LING. REG.: 0, 1, 2, 3

CHIUSURA RISPETTO AL COMPLEMENTO: 1, 3

TIPO 0: non esiste un algoritmo generale per decidere se una stringa app. al ling. (il problema è semi-decidibile). Esiste invece per quelli di tipo 1.
TIPO 3: è decidibile se due grammatiche definiscano lo stesso linguaggio.
TIPO 0 e 1: la derivazione non è più rappresentabile come albero.

ESEMPI GRAMMATICHE DI TIPO 1:

$$L = \left\{ a^n b^n c^n \mid n \geq 1 \right\}$$

$$G(\text{tipo 1}):\ 1.\ S \to aSBC \quad 3.\ CB \to BC \quad 5.\ bC \to bc$$

$$2.\ S \to abC \quad 4.\ bB \to bb \quad 6.\ cC \to cc$$

A differenza delle grammatiche libere, non vale la proprietà che il linguaggio generato mediante derivazioni sinistre coincide con quello generato mediante derivazioni destre (i.e., l'ordine di applicazione delle regole e` rilevante).

Altro esempio di linguaggio che richiede grammatica di tipo 1:
REPLICHE O LISTE DI CONCORDANZE

$$L_{replica} = \left\{ uu \mid u \in \Sigma^+ \right\}$$

$$\Sigma = \left\{ a, b \right\} \quad x = abbbabbb$$

Grammatiche contestuali (tipo 1): raramente usate per la progettazione di linguaggi artificiali e compilatori (eccezione Algol-68 definito mediante gramm. contestuale a due livelli. Costrutti utili come repliche o concordanze tra liste vengono definite con metodi di natura semantica (gramm. ad attributi o traduzioni guidate da sintassi).