Grammatiche generative libere da contesto - III

Prof. A. Morzenti

aa 2008-2009

EQUIVALENZA DEBOLE E FORTE (STRUTTURALE)

EQUIVALENZA DEBOLE: due grammatiche debolmente equivalenti se generano lo stesso linguaggio: L(G) = L(G').

G e G' potrebbero assegnare strutture (quindi alberi sintattici) diverse alla stessa frase

Struttura assegnata a una frase e` importante: usata da traduttori e interpreti

EQUIVALENZA FORTE o STRUTTURALE: due grammatiche G e G' sono equivalenti in senso forte o strutturale se

- L(G) = L(G') (Eq. DEBOLE) e inoltre
- G e G' strutturalmente equivalenti (alberi scheletrici condensati)

Eq.FORTE $\rightarrow$ Eq.DEBOLE MA Eq.FORTE$\neq$Eq. DEBOLE

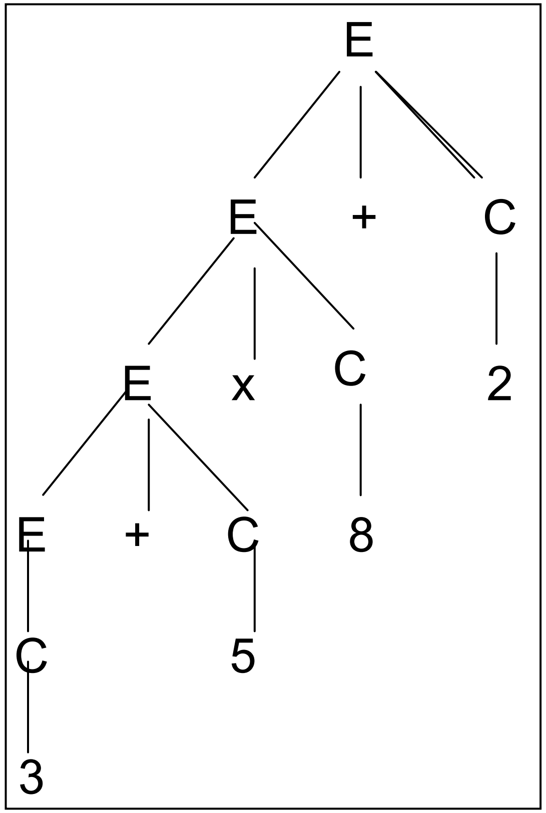
(quindi $\neg$(Eq.DEBOLE$\rightarrow$Eq.FORTE)

Eq. FORTE e` DECIDIBILE

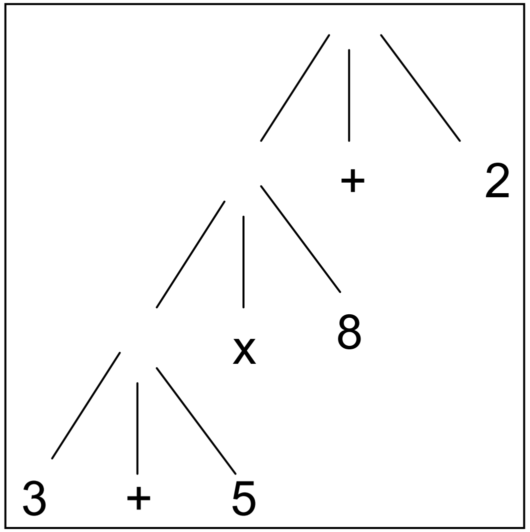
Eq. DEBOLE ***NON*** e` DECIDIBILE

ESEMPIO: Equivalenza strutturale di espressioni aritmetiche – 3 + 5 x 8 + 2

$$G_1: \quad E \to E + C \quad E \to E \times C \quad E \to C$$
$$C \to 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9$$

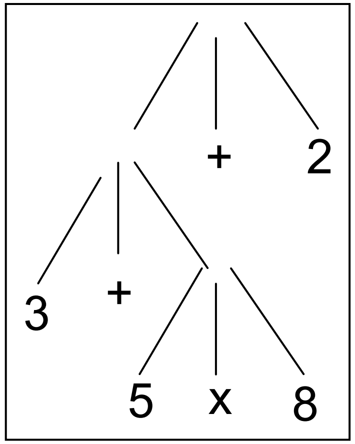


versione scheletrica condensata

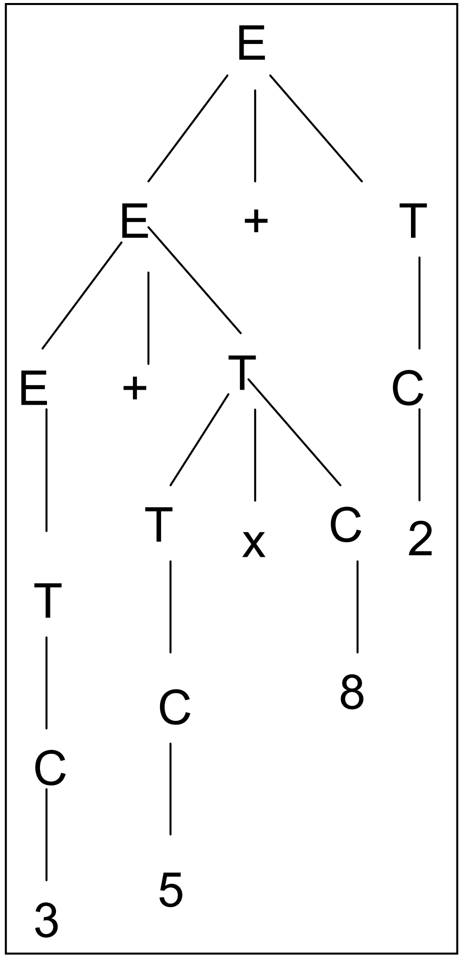


NB: regole di categorizzazione

$$G_2: \quad E \to E + T \quad E \to T \quad T \to T \times C \quad T \to C$$
$$C \to 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9$$



vers. schel. condensata



G_1 e G_2 non equivalenti in senso strutturale

interpretazioni semantiche: G_1: (((3 + 5) x 8) + 2)
G_2: ((3 + (5 x 8)) + 2)

Solo G_2 è adeguata strutturalmente rispetto alle regole di precedenza degli operatori (NB: G_2 è più complessa) obbliga a generare prodotto da n.t. T <u>dopo</u> E quindi assegna al prodotto priorita` maggiore della somma

Nella definizione formale di un linguaggio non si può prescindere dalla ADEGUATEZZA STRUTTURALE perché di solito la grammatica serve come supporto per l'interpretazione semantica o per una traduzione guidata da sintassi.

G_3 equivalente strutturalmente alla precedente:

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T+T \mid C+T \mid E+C \mid T+C \mid C+C \mid T\times C \mid C\times C \mid C \\ T \rightarrow T\times C \mid C\times C \mid C \\ C \rightarrow 0\mid 1\mid 2\mid 3\mid 4\mid 5\mid 6\mid 7\mid 8\mid 9 \end{array}}$$

G_3 ha più regole: non sfrutta tutte **regole di categorizzazione** della G_2

Categorizzazioni e tassonomie riducono la complessità della descrizione.

EQUIVALENZA STRUTTURALE IN SENSO LATO

$$\{S \to Sa \mid a\} \quad \{X \to aX \mid a\}$$

$$L = a^+$$

$$\underbrace{\underbrace{a}\,a} \qquad \underbrace{a\,\underbrace{a}}$$

Non c'è equivalenza strutturale, ma c'è equivalenza strutturale in senso debole: a ogni albero lineare a sinistra della prima grammatica corrisponde un albero lineare a destra della seconda (generano due alberi specularmente identici, ottenibili sistematicamente uno dall'altro girando le ricorsioni da sinistra a destra).

FORME NORMALI DELLE GRAMMATICHE

Impongono restrizioni alla forma delle regole senza ridurre la classe dei linguaggi generati.

Utili per la dimostrazione di teoremi più che per la progettazione di linguaggi.

Vediamo alcune trasformazioni usate per portare una grammatica in forma normale ma utili anche per la costruzione di analizzatori sintattici

GRAMMATICA di partenza: $\boxed{G = \{\Sigma, V, P, S\}}$

ESPANSIONE di un NONTERMINALE
(sua ELIMINAZIONE dalle regole della grammatica)

Grammatica $A \to \alpha B \gamma \quad B \to \beta_1 | \beta_2 | \ldots | \beta_n$

Diventa $A \to \alpha\beta_1\gamma \mid \alpha\beta_2\gamma \mid \ldots\alpha\beta_n\gamma$

Derivazione $A \Rightarrow \alpha B \gamma \Rightarrow \alpha\beta_i\gamma$

Diventa $A \Rightarrow \alpha\beta_i\gamma$

ELIMINAZIONE DELL'ASSIOMA DALLE PARTI DESTRE:

È sempre possibile restringere le parti destre delle regole a essere stringhe in (Σ U (V \ {S})), cioè prive dell'assioma S.
Basta introdurre un nuovo assioma S_0 e la regola $S_0 \rightarrow S$

NONTERMINALI ANNULLABILI e FORMA NORMALE SENZA REGOLE VUOTE

Un nonterminale è annullabile se esiste una derivazione: $\boxed{A \overset{+}{\Rightarrow} \varepsilon}$

$Null \subseteq V$ è l'insieme dei nt annullabili

calcolo dell'insieme $Null$

$A \in Null$ if $A \rightarrow \varepsilon \in P$

$A \in Null$ if $(A \rightarrow A_1 A_2 \dots A_n \in P \text{ con } A_i \in V \setminus \{A\}) \wedge \forall A_i (A_i \in Null)$

ESEMPIO – Determinazione dei nonterminali annullabili

$$S \rightarrow SAB \mid AC \quad A \rightarrow aA \mid \varepsilon \quad B \rightarrow bB \mid \varepsilon \quad C \rightarrow cC \mid c$$

$$Null = \{A, B\}$$

NB: Se fosse $S \rightarrow AB$ allora sarebbe annullabile anche S.

FORMA NORMALE NON ANNULLABILE (senza regole vuote) vale la condizione che nessun nt diverso dall'assioma sia annullabile.
L'assioma è annullabile solo se la stringa vuota appartiene al linguaggio.

COSTRUZIONE DELLA FORMA NORMALE NON ANNULLABILE:

1) Calcolo dell'insieme *Null.*
2) Per ogni regola di P si aggiungono come regole alternative quelle ottenute cancellando dalla parte destra, in tutti i modi possibili, i simboli annullabili.
3) Si tolgono le regole A → ε, tranne che per A = S.
4) Pulizia della grammatica e rimozione delle circolarità.

ESEMPIO (continuato)

cancellati A e B in tutti i modi, eliminate regole $X \to \varepsilon$

eliminata regola $S \to S$

$Annull.$	$G\ orig$	$G''\ da\ pulire$	$G''\ senza\ reg.\ vuote$
F	$S \to SAB \mid AC$	$S \to SAB \mid SA \mid SB \mid S \mid AC \mid C$	$S \to SAB \mid SA \mid SB \mid AC \mid C$
V	$A \to aA \mid \varepsilon$	$A \to aA \mid a$	$A \to aA \mid a$
V	$B \to bB \mid \varepsilon$	$B \to bB \mid b$	$B \to bB \mid b$
F	$C \to cC \mid c$	$C \to cC \mid c$	$C \to cC \mid c$

REGOLE DI COPIATURA (o categorizzazione) E LORO ELIMINAZIONE

Se A $\rightarrow$B la classe sintattica B è inclusa nella classe sintattica A.
Eliminando regole di copiatura grammatiche equivalenti con alberi meno profondi.

Copia(A) $\subseteq$ V: insieme dei n.t. in cui il n.t. A si può ricopiare anche transitivamente

$$Copia(A) = \{B \in V \mid A \overset{*}{\Rightarrow} B\}$$

Esempio tipico: *frase_iterativa* $\rightarrow$ *frase_while* | *frase_for* | *frase_repeat*

NB: regole di copiatura consentono di portare a fattor comune certe parti
Quindi riducono le dimensioni della grammatica
Nelle grammatiche dei linguaggi tecnici le copiature di solito sono sfruttate

1) Calcolo di *Copia* (ipotizzando la grammatica priva di regole vuote), espresso dalle seguenti clausole logiche applicate fino al raggiungimento di un punto fisso.

$$\boxed{\begin{array}{l} A \in Copia(A) \\ C \in Copia(A) \text{ if } (B \in Copia(A)) \wedge (B \to C \in P) \end{array}}$$

(NB: non avendo regole vuote si può escludere il caso $B \to CD$ e $D \Rightarrow^* \varepsilon$)

2) Si costruiscono le regole della grammatica G’, equivalente a G ma priva di Ricopiature.

$$P' := P \setminus \{A \to B \mid A, B \in V\} \quad \text{--togli regole di copiatura}$$

$$P' := P' \cup \{A \to \alpha \mid \alpha \in ((\Sigma \cup V)^* \setminus V)\} \quad \text{--aggiungi nuove regole}$$

$$\textit{dove } B \in Copia(A) \wedge (B \to \alpha) \in P$$

$$\text{La derivazione } A \overset{*}{\Rightarrow} B \Rightarrow \alpha \text{ diviene } A \overset{*}{\Rightarrow} \alpha$$

ESEMPIO – Espressioni aritmetiche senza copiature.

$$E \rightarrow E + T \,|\, T \quad T \rightarrow T \times C \,|\, C$$

$$C \rightarrow 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9$$

$$Copia(E) = \{E, T, C\} \quad Copia(T) = \{T, C\}$$

$$Copia(C) = \{C\}$$

La grammatica equivalente senza copiature:

T∈Copia(E) C∈Copia(E)

$$E \rightarrow E + T \,|\, T \times C \,|\, 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9$$

$$T \rightarrow T \times C \,|\, 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9$$ C∈Copia(E)

$$C \rightarrow 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9$$

FORMA NORMALE DI CHOMSKY: le regole sono di due tipi

1) Regole *omogenee binarie:* $A \to BC$ *dove* $B,C \in V$
2) Regole *terminali con parte destra unitaria* $A \to a$, $a \in \Sigma$
NB: Gli alberi sintattici hanno nodi interni di grado 2 e genitori di foglie di grado 1

Procedimento per rendere G (assunta priva di n.t. annullabili) in F.N.Chomsky

Se la stringa vuota è nel linguaggio, si aggiunge la regola: $S \to \varepsilon$

Si applica iterativamente seguente processo

ogni regola di tipo $\longrightarrow$ $A_0 \to A_1 A_2 \dots A_n$

si aggiunge una regola … $\longrightarrow$ $A_0 \to \langle A_1 \rangle \langle A_2 \dots A_n \rangle$ $\longleftarrow$ $\langle A_2 \dots A_n \rangle$ nuovo n.t. introdotto allo scopo

e anche un'altra regola … $\longrightarrow$ $\langle A_2 \dots A_n \rangle \to A_2 \dots A_n$

…alla fine, prima o poi … $\longrightarrow$ se A_1 è terminale: $\langle A_1 \rangle \to A_1$

ESEMPIO: Conversione in forma di Chomsky

$$S \rightarrow dA \mid cB \quad A \rightarrow dAA \mid cS \mid c \quad B \rightarrow cBB \mid dS \mid d$$

$$S \rightarrow <d> A \mid <c> B$$

$$A \rightarrow <d><AA> \mid <c> S \mid c$$

$$B \rightarrow <c><BB> \mid <d> S \mid d$$

$$<d> \rightarrow d \qquad <c> \rightarrow c$$

$$<AA> \rightarrow AA \qquad <BB> \rightarrow BB$$

TRASFORMAZIONE DELLE RICORSIONI SINISTRE (S-ricorsioni) IN DESTRE

Costruzione di forma NON RICORSIVA A SINISTRA
(NB: necessaria per la costruzione di ***analizzatori sintattici discendenti***).

Caso 1 (semplice): TRASFORMAZIONE DELLE S-RICORSIONI ***IMMEDIATE***

$$\begin{cases} A \to A\beta_1 \mid A\beta_2 \mid \ldots \mid A\beta_h \\ \text{dove nessun } \beta_i \text{ è vuoto} \\ A \to \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k \end{cases}$$

introdotto nuovo n.t. A’

$$\begin{cases} A \to \gamma_1 A' \mid \gamma_2 A' \mid \ldots \mid \gamma_k A' \mid \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k \\ A' \to \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_h A' \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_h \end{cases}$$

derivazione nella gramm. con S-ricors. $\longrightarrow A \Rightarrow A\beta_2 \Rightarrow A\beta_3\beta_2 \Rightarrow \gamma_1\beta_3\beta_2$

deriv. nella gramm. priva di S-ricors. $\longrightarrow A \Rightarrow \gamma_1 A' \Rightarrow \gamma_1\beta_3 A' \Rightarrow \gamma_1\beta_3\beta_2$

ESEMPIO – Spostamento a destra di s-ricorsioni immediate.

$$E \to E+T \mid T \qquad T \to T*F \mid F \qquad F \to (E) \mid i$$

E e T sono imm. ricorsivi a sinistra

$$E \to TE' \mid T \qquad E' \to +TE' \mid +T$$

$$T \to FT' \mid F \qquad T' \to *FT' \mid *F \qquad F \to (E) \mid i$$

semplice trasf. non sempre adeguata:

$$E \to T+E \mid T \qquad T \to F*T \mid F \qquad F \to (E) \mid i$$

rovesciare specularmente le regole qui funziona, ma non in generale

2) TRASFORMAZIONE DELLE S-RICORSIONI NON IMMEDIATE

IPOTESI: G in forma omogenea (nella parte dx solo n.t. o solo t.) e non annullabile, con regole terminali di lunghezza unitaria
(ossia analoghe a quella di Chomsky senza il vincolo che vi siano due nt).

L'ALGORITMO usa due cicli for annidati:

1) espansione per ottenere s-ricorsioni immediate
2) sostituzione delle s-ricorsioni immediate con d-ricorsioni

Conviene pensare l'alfabeto n.t. come ordinato da 1 a *m:*

$$V = \{A_1, A_2, ..., A_m\}$$

A_1 è l'assioma

IDEA dell'algoritmo: modificare le regole in modo che, se una regola inizia con un n.t., e.g., $A_i \rightarrow A_j\alpha$, allora risulti, dopo la trasformazione, $j > i$.

ALGORITMO DI ELIMINAZIONE DELLE S-RICORSIONI ANCHE NON IMMEDIATE

for *i*:=1 **to** *m* **do**
  for *j*:=1 **to** *i* –1 **do**
    sostituisci a ogni regola della forma $A_i \to A_j\alpha$ --NB qui i>j
    le regole:
    $A_i \to \gamma_1 \alpha | \gamma_2 \alpha | \ldots | \gamma_k \alpha$
    (creando possibili s-ricorsioni immediate)
    dove $A_j \to \gamma_1 | \gamma_2 | \ldots | \gamma_k$ sono alternative di A_j
  end do
  elimina con l'algoritmo precedente, le eventuali
  s-ricorsioni immediate apparse nelle alternative di A_i,
  creando il nuovo nt A'_i
end do

Con semplice modifica gli stessi algoritmi possono trasformare d-ric in s-ric, operazione talvolta richiesta per algoritmi di analisi sintattica ascendente.

ESEMPIO – Applichiamo l'algoritmo alla grammatica G_3

$$A_1 \rightarrow A_2 a \mid b \quad A_2 \rightarrow A_2 c \mid A_1 d \mid e$$

$$A_1 \Rightarrow A_2 a \Rightarrow A_1 da$$ NB s-ricorsione non immediata

i=1 Elimina le s-ricorsioni immediate di A_1 (non ve ne sono).

grammatica immutata

i=2 j=1 Sostituisci a $A_2 \rightarrow A_1 d$ le regole ottenute con l'espansione di A_1

$$\boxed{\begin{array}{l} A_1 \rightarrow A_2 a \mid b \\ A_2 \rightarrow A_2 c \mid A_2 ad \mid bd \mid e \end{array}}$$

Elimina la s-ricorsione immediata ottenendo G'_3

$$\boxed{\begin{array}{l} A_1 \rightarrow A_2 a \mid b \\ A_2 \rightarrow bdA' \mid eA' \mid bd \mid e \\ A' \rightarrow cA' \mid adA' \mid c \mid ad \end{array}}$$

FORMA NORMALE IN TEMPO REALE e DI GREIBACH

Nella FORMA NORMALE IN TEMPO REALE ogni regola inizia con un simbolo terminale.

$$\boxed{A \to a\alpha \;\; \text{dove} \;\; a \in \Sigma, \alpha \in \left\{\Sigma \bigcup V\right\}^*}$$

La FORMA NORMALE DI GREIBACH è un caso particolare del precedente.

Ogni regola comincia con un terminale seguito da zero o più nonterminali.

$$\boxed{A \to a\alpha \;\; \text{dove} \;\; a \in \Sigma, \alpha \in V^*}$$

“IN TEMPO REALE” - il termine è legato a una proprietà dell’algoritmo di analisi sintattica: ad ogni passo esso legge e consuma un carattere terminale. Il numero di passi necessari per completare l’analisi è esattamente uguale alla lunghezza della stringa da analizzare.

ALGORITMO di TRASFORMAZIONE IN FORMA NORMALE IN TEMPO REALE e in FORMA NORMALE DI GREIBACH

Ipotesi: grammatica priva di nt annullabili

1) eliminazione delle s-ricorsioni
2) con trasformazioni elementari, espansione dei nt event. pres. in prima posizione
3) introduzione di nuovi nt al posto dei terminali che event. cadono in posizioni diverse dalla prima

Se non si esegue l'ultimo passo dell'algoritmo precedente, nelle regole possono rimanere simboli terminali in posizioni successive alla prima: la grammatica è allora nella forma normale in tempo reale, pur se non in quella di Greibach.

ESEMPIO –Grammatica: $A_1 \to A_2 a \quad A_2 \to A_1 c \mid bA_1 \mid d$

1) Eliminazione delle s-ricorsioni.

$$A_1 \to A_2 a \quad A_2 \to A_2 ac \mid bA_1 \mid d$$

ha solo s-ricors. immediate

$$A_1 \to A_2 a \quad A_2 \to bA_1A'_2 \mid dA'_2 \mid d \mid bA_1 \quad A'_2 \to acA'_2 \mid ac$$

2) Sostituzione dei nt in prima pos fino a far comparire un nt in prima pos.

$$A_1 \to bA_1A'_2 a \mid dA'_2 a \mid da \mid bA_1 a$$

$$A_2 \to bA_1A'_2 \mid dA'_2 \mid d \mid bA_1$$

$$A'_2 \to acA'_2 \mid ac$$

3) Introduzione di nuovi nt al posto dei ter in pos diverse dalla prima.

$$A_1 \to bA_1A'_2 <a> \mid dA'_2 <a> \mid d <a> \mid bA_1 <a>$$

$$A_2 \to bA_1A'_2 \mid dA'_2 \mid d \mid bA_1$$

$$A'_2 \to a <c> A'_2 \mid a <c>$$

$$<a> \to a \quad <c> \to c$$