Teoria dei linguaggi e grammatiche

Un Alfabeto Σ è un insieme di elementi, detti simboli. Si denota con $|\Sigma|$ la cardinalità dell'alfabeto Σ.

Una stringa $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ è una sequenza di simboli. Si denota con $|\alpha|$ la lunghezza della stringa.

Un Linguaggio è un insieme di stringhe. Dato un linguaggio $\mathbf{L}$ si denota con $|\mathbf{L}|$ la sua cardinalità (finita o infinita).

Dato un linguaggio $\mathbf{L}$, una stringa $\alpha \in \mathbf{L}$ è detta frase di $\mathbf{L}$.

Operazioni sulle stringhe

Date due stringhe $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ e $\beta = \beta_1 \beta_2 \ldots \beta_n$, $\alpha = \beta$ se e solo se $\alpha_i = \beta_i$ per $1 \le i \le n$.

Date due stringhe $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ e $\beta = \beta_1 \beta_2 \ldots \beta_n$ si definisce concatenazione di α e β la stringa:

$\gamma = \alpha\beta = \alpha_1 \alpha_2 \ldots \alpha_n \beta_1 \beta_2 \ldots \beta_n$.

La concatenazione è un'operazione associativa ma non commutativa.

Si definisce stringa nulla la stringa ε tale che $|\varepsilon| = 0$. Data una qualunque stringa α, $\alpha\varepsilon = \varepsilon\alpha$.

Date le stringhe α, β, γ e la stringa $\delta = \alpha\beta\gamma$, le stringhe α, β, γ sono dette sottostringhe di δ, e le stringhe α e γ sono dette rispettivamente prefisso e suffisso di δ.

Data la stringa $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ la stringa $\alpha^R = \alpha_n \alpha_{n-1} \ldots \alpha_1$ è detta stringa riflessa di α. Data la stringa α, $(\alpha^R)^R = \alpha$, $(\alpha\beta)^R = \beta^R\alpha^R$.

Data la stringa α, la potenza ennesima α^n di α è la stringa ottenuta concatenando n-1 volte α con se stessa; $\alpha^0 = \varepsilon$.

Operazioni sui linguaggi

Dati due linguaggi $\mathbf{L}_1$ e $\mathbf{L}_2$, si definisce concatenazione di $\mathbf{L}_1$ e $\mathbf{L}_2$ il linguaggio $\mathbf{L} = \mathbf{L}_1\mathbf{L}_2 = \{ \alpha\beta \mid \alpha \in \mathbf{L}_1 \text{ e } \beta \in \mathbf{L}_2 \}$

Si definisce il linguaggio vuoto Φ tale che $\mathbf{L}\Phi = \Phi\mathbf{L}$; $|\Phi| = 0$.

Data il linguaggio $\mathbf{L}$, la potenza ennesima $\mathbf{L}^n$ di $\mathbf{L}$ è il linguaggio ottenuto concatenando n-1 volte $\mathbf{L}$ con se stesso; $\mathbf{L}^0 = \{\varepsilon\}$.

Dato il linguaggio $\mathbf{L}$, si definisce chiusura riflessiva di $\mathbf{L}$ rispetto alla concatenazione il linguaggio $\mathbf{L}^* = {}_{i=0}\cup^{\infty} \mathbf{L}^i$.

Proprietà della chiusura riflessiva:

$\mathbf{L}^* \supseteq \mathbf{L}$	monotonicità
$(x \in \mathbf{L}^*) \wedge (y \in \mathbf{L}^*) \Rightarrow xy \in \mathbf{L}^*$	chiusura
$(\mathbf{L}^*)^* = \mathbf{L}^*$	idempotenza
$(\mathbf{L}^*)^R = (\mathbf{L}^R)^*$	commutatività

Si definisce linguaggio universale (o monoide libero) di alfabeto Σ il linguaggio $\mathbf{L} = \Sigma^*$. Dato il linguaggio $\mathbf{L}$, si definisce complemento di $\mathbf{L}$ (scritto $\neg\mathbf{L}$) il suo complemento rispetto a Σ^*.

Dato il linguaggio $\mathbf{L}$, si definisce chiusura non riflessiva di $\mathbf{L}$ rispetto alla concatenazione il linguaggio:

$$\mathbf{L}^+ = {}_{i=1}\cup^{\infty} \mathbf{L}^i.$$

Si noti che: $\mathbf{L}^* \supseteq \mathbf{L}^+ \cup \{\varepsilon\}$, $\mathbf{L}^+ = \mathbf{L}\mathbf{L}^* = \mathbf{L}^*\mathbf{L}$.

Siano dati i linguaggi $\mathbf{L}$ ed $\mathbf{L}_b$, con alfabeti rispettivamente Σ e Δ, e sia $b \in \Sigma$. Si definisce sostituzione di $\mathbf{L}_b$ a b in $\mathbf{L}$ il linguaggio $\mathbf{L}'$ tale che:

$\mathbf{L}' = \{\alpha \in ((\Sigma - \{b\}) \cup \Delta)^* \mid \alpha = x_1y_1x_2y_2 \ldots y_{k-1}x_k$, con $x_1bx_2b \ldots bx_k \in \mathbf{L}$, $x_i \in (S - \{b\})^*$, $y_i \in \mathbf{L}_b$, $1 \le i \le k\}$

Dati i linguaggi $\mathbf{L}_1$ e $\mathbf{L}_2$ si definisce quoziente di $\mathbf{L}_1$ rispetto a $\mathbf{L}_2$ il linguaggio $\mathbf{L}$ tale che:

$$\mathbf{L} = \{\alpha \mid (\exists\, \beta \in \mathbf{L}_2) \wedge (\alpha\beta \in \mathbf{L}_1)\}$$

Espressioni e linguaggi regolari

Un linguaggio $\mathbf{L}_\Sigma$ di alfabeto $\Sigma = \{a_1, a_2, ..., a_k\}$ è detto regolare se può essere espresso mediante operazioni di concatenazione, unione e chiusura riflessiva applicate un numero finito di volte ai linguaggi unitari (i.e. composti da una sola stringa) $\mathbf{L}_1 = \{a_1\}$, $\mathbf{L}_2 = \{a_2\}$, ... $\mathbf{L}_k = \{a_k\}$, ed al linguaggio vuoto Φ.

Una espressione regolare (e.r.) x di alfabeto Σ è definita costruttivamente come segue:

1. $x = \Phi$
2. $x = \varepsilon$
3. $x = a$, dove $a \in \Sigma$
4. $x = yz$, dove y e z sono e.r.
5. $x = y \cup z$ dove y e z sono e.r.
6. $x = y^*$ dove y è e.r.
7. nient'altro è un'espressione regolare

Ogni linguaggio finito è regolare, in quanto unione di un numero finito di linguaggi unitari.

NOTA: il linguaggio delle espressioni regolari può essere esteso con gli operatori potenza (x^n) e chiusura transitiva (x^+).

Una sottoespressione (s.e) di una e.r. è così definita:

x_k è una s.e. di $(x_1 \cup .... x_k \cup .... x_n)$
x_k è una s.e. di $x_1 .... x_k .... x_n$
x è una s.e. di x^*, x^+, x^n
x è una s.e. di se stessa
ε è una s.e. di ogni espressione.

Date le e.r. x ed y, x implica y (scritto $x \Rightarrow y$) se il linguaggio definito da x include quello definito da y:
$(x_1 \cup .... x_k \cup .... x_n) \Rightarrow x_k \quad 1 \le k \le n$;
$(x)^* \Rightarrow xx ... x$ (ripetuto k volte, $k \ge 0$)
$(x)^+ \Rightarrow xx ... x$ (ripetuto k volte, $k > 0$)
$(x)^n \Rightarrow xx ... x$ (ripetuto n volte)

Date due e.r. x_1 ed x_2, x_1 diviene immediatamente x_2 (scritto $x_1 \rightarrow x_2$) se:

$x_1 = \alpha\beta\gamma$, $x_2 = \alpha\delta\gamma$, $\beta \Rightarrow \delta$

dove α, β, γ sono sottoespressioni di x_1

Nel modo solito si definiscono $x_1 \rightarrow^n x_2$, $x_1 \rightarrow^+ x_2$, $x_1 \rightarrow^* x_2$

Data una e.r. x, il linguaggio $\mathbf{L}(x)$ prodotto da x è:

$$\mathbf{L}(x) = \{ \alpha \in V_T^* \mid x \rightarrow^* \alpha \}$$

Grammatiche generative

Una grammatica è una quadrupla $G = \langle V_T, V_N, P, Z \rangle$ dove:

V_T = insieme di simboli terminali

V_N = insieme di simboli non terminali ($V_T \cap V_N = \varnothing$)

P = insieme di produzioni (o regole di riscrittura)

Z = assioma ($Z \in V_N$)

Le produzioni hanno la seguente struttura:

$$\alpha ::= \beta$$

dove α è detta parte sinistra e β è detta parte destra.

Più produzioni aventi la stessa parte sinistra possono essere raggruppate usando la notazione:

$$\alpha ::= \beta_1 \mid \beta_2 \mid \ldots . \beta_n$$

Date due stringhe $\alpha \in (V_T \cup V_N)^+$, $\beta \in (V_T \cup V_N)^*$ si dice che β deriva da α oppure che α si riduce a β

($\alpha \Rightarrow \beta$) se e solo se:

$$\alpha = \alpha_1 \gamma \alpha_2 \ , \ \beta = \beta_1 \delta \beta_2 \ , \ \gamma ::= \delta$$

Conseguentemente:

$\alpha \Rightarrow^n \beta$ se e solo se $\alpha \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \Rightarrow \gamma_{n-1} \Rightarrow \beta$

$\alpha \Rightarrow^+ \beta$ se e solo se $\alpha \Rightarrow^n \beta$ con $n > 0$

$\alpha \Rightarrow^* \beta$ se e solo se $\alpha \Rightarrow^n \beta$ con $n \geq 0$

Data una grammatica G, il linguaggio generato da G è: $\mathbf{L}(G) = \{ \alpha \in V_T^* \mid Z \Rightarrow^* \alpha \}$.

Classi di linguaggi:

Tipo 0	$\alpha ::= \beta$, dove $\alpha \in V^*$ ($\alpha \in V_N^*$), $\beta \in V^*$ $V = V_T \cup V_N$	linguaggi ricorsivamente enumerabili	Macchina di Turing
Tipo 1	$\alpha ::= \beta$, dove $\alpha \in V^*$, $\beta \in V^*$ $\lvert\alpha\rvert \leq \lvert\beta\rvert$ $V = V_T \cup V_N$	linguaggi contestuali (context sensitive, CS)	Macchina di Turing con nastro finito
Tipo 2	$A ::= \beta$, dove $A \in V_N$, $\beta \in V^*$ $V = V_T \cup V_N$	linguaggi non contestuali (context free, CF)	Automa a pila non determi-nistico
Tipo 3	$A ::= xB$, dove $A \in V_N$, $x \in V_T$ $B \in (V_N \cup \{\varepsilon\})$	linguaggi regolari	Automa a stati finiti

Data una grammatica G ed un simbolo $A \in V_N$, il linguaggio generato da G partendo da A è: $\mathbf{L}_A(G) = \{ \alpha \in V_T^* \mid A \Rightarrow^* \alpha \}$.

Data una grammatica G ed un simbolo $A \in V_N$, una forma di frase generata da G partendo da A è una stringa $\alpha \in (V_T \cup V_N)^*$ tale che $A \Rightarrow^* \alpha$.

Data una grammatica G, una frase generata da G è una stringa $\alpha \in V_T^*$ tale che $Z \Rightarrow^* \alpha$, (si noti che $\alpha \in \mathbf{L}(G)$).

Linguaggi non contestuali

Una grammatica non contestuale (cf) G si dice ridotta se valgono le seguenti condizioni:

1) non consente derivazioni circolari del tipo $A \Rightarrow^+ A$ per qualunque $A \in V_N$
2) ogni non terminale A è raggiungibile dall'assioma: $Z \Rightarrow^* \alpha A \beta$, con $\alpha, \beta \in (V_T \cup V_N)^*$ per qualunque $A \in V_N$
3) ogni non terminale A genera un linguaggio non vuoto: $\mathbf{L}_A(G) \neq \Phi$

Un simbolo $A \in V_N$ si dice ricorsivo se esiste una derivazione $A \Rightarrow^+ \alpha A \beta$, con $\alpha, \beta \in (V_T \cup V_N)^*$. La ricorsione è diretta se $A \Rightarrow \alpha A \beta$, altrimenti è indiretta.

La derivazione $A \Rightarrow^+ b_1 b_2 \ldots b_n$, $b_i \in (V_T \cup V_N)$, $1 \le i \le n$, può essere descritta mediante un albero, detto albero sintattico, con radice A e foglie b_1 , $b_2 \ldots$, b_n (da sinistra a destra).

La derivazione $\alpha_0 \Rightarrow \alpha_1 \ldots \Rightarrow \alpha_n$, dove $\alpha_i = \beta_i A \gamma_i$, $\alpha_{i+1} = \beta_i \delta_i \gamma_i$, con $\delta_i, \beta_i, \gamma_i \in (V_T \cup V_N)^*$, $A \in V_N$, è detta canonica sinistra (destra) se $\beta_i \in V_T^*$ ($\gamma_i \in V_T^*$) per ogni i: $0 \le i \le n-1$, ed è denotata con $\alpha_0 \ {}_s\Rightarrow^+ \alpha_n$
($\alpha_0 \ {}_d\Rightarrow^+ \alpha_n$).

Una frase $\alpha \in \mathbf{L}(G)$ è detta sintatticamente ambigua se è generabile da G con n ($n \ge 2$) alberi sintattici distinti (n è detto grado di ambiguità); in tal caso anche la grammatica è detta ambigua (proprietà indecidibile).

Un linguaggio è inerentemente ambiguo se tutte le grammatiche che lo generano sono ambigue.

Due grammatiche G_1 e G_2 sono debolmente equivalenti se $\mathbf{L}(G_1) = \mathbf{L}(G_2)$; la proprietà di equivalenza debole è indecidibile.

Trasformazioni sintattiche conservative (cf)

1) *Sostituzione*

Siano $B ::= \beta_1 | \beta_2 | \ldots\ldots \beta_n$ tutte le produzioni della grammatica G aventi il simbolo B come parte sinistra. Se la produzione $A ::= \alpha B \gamma$ di G è sostituita dalle produzioni $A ::= \alpha\beta_1\gamma | \alpha\beta_2\gamma | \ldots\ldots \alpha\beta_n\gamma$, la grammatica G' così ottenuta è debolmente equivalente a G.

2) *Eliminazione delle ricorsioni sinistre*

Un nonterminale A è ricorsivo a sinistra (destra) se esiste una derivazione $A \Rightarrow^+ A\beta$, ($A \Rightarrow^+ \beta A$) con $\beta \neq \varepsilon$

Una produzione $A ::= Ab$ è detta immediatiamente ricorsiva (i.r.) a sinistra .

<u>Eliminazione delle produzioni i.r. a sinistra</u>

Sia G una grammatica contenente le produzioni:

(1) $A ::= A\beta_1 | A\beta_2 | \ldots A\beta_h | \gamma_1 | \gamma_2 | \ldots \gamma_k$

con $\beta_1, \beta_2, \ldots \beta_h \neq \varepsilon$

La grammatica G' ottenuta sostituendo le (1) con:

$A ::= \gamma_1 A' | \gamma_2 A' | \ldots \gamma_k A'$

$A' ::= \beta_1 A' | \beta_2 A' | \ldots \beta_h A' | \varepsilon$

è debolmente equivalente a G.

Eliminazione delle ricorsioni non immediate

Sia data la grammatica G ricorsiva a sinistra, con $V_N = \{ A_1, A_2, \ldots A_n \}$ e priva di ε-regole (non esistono produzioni $A_i ::= \varepsilon$).

Algoritmo

```
for i := 1 to n do
  begin
    for k := 1 to i -1 do
      begin
      sostituire ogni produzione della forma
                Ai ::= Ak α
      con le produzioni Ai ::= γ1 α| γ2 α| ... γh α
      dove Ak ::= γ1 | γ2 | ... γh sono tutte le
      produzioni presenti con parte sinistra Ak
      end
    eliminare le ricorsioni immediate a sinistra di Ai
  end
```

La grammatica G' ottenuta applicando il precedente algoritmo alla grammatica G è debolmente equivalente a G.

Forme normali per linguaggi cf

Data una grammatica G possono essere costruite forme normali debolmente equivalenti.

1) *Forma normale di Chomsky*
   Le produzioni sono di due tipi:
   $A ::= BC$ oppure $A ::= a$
   con $A,B,C \in V_N$ e $a \in (V_T \cup \{\varepsilon\})$

2) *Forma normale di Greibach*
   Le produzioni sono del tipo:
   $A ::= a\alpha$
   con $a \in V_T$ e $\alpha \in (V_T \cup V_N)^*$

La forma di Greibach si applica a linguaggi privi di ε-regole e si ottiene applicando ripetutamente le trasformazioni di sostituzione ed eliminazione delle ricorsioni sinistre.

Grammatiche non contestuali estese (ecf)

Le produzioni di una grammatica ecf hanno la seguente struttura: $A ::= \alpha$, dove $A \in V_N$, α è una e.r. di alfabeto $(V_T \cup V_N)$.

Data la produzione $A ::= \alpha$, si dice che A produce β (scritto $A \Rightarrow \beta$) se $\alpha \Rightarrow \beta$, dove α e β sono e.r.

Grammatiche lineari

Una grammatica cf è detta lineare se ogni produzione ha la forma: $A ::= \alpha$, dove $\alpha \in V_T^* \times (V_N \cup \{\varepsilon\}) \times V_T^*$.

Una grammatica cf è detta lineare destra (sinistra) se ogni produzione ha la forma: $A ::= \alpha B$ ($A ::= B\alpha$), dove $\alpha \in V_T^*$, $B \in (V_N \cup \{\varepsilon\})$.

Una grammatica cf è detta strettamente lineare destra (sinistra) se ogni produzione ha la forma: $A ::= cB$ ($A ::= Bc$), dove $c \in V_T$, $B \in (V_N \cup \{\varepsilon\})$.

Per ogni grammatica lineare destra (sinistra) G esiste una grammatica strettamente lineare G' debolmente equivalente.

La classe dei linguaggi generati dalle grammatiche lineari destre e lineari sinistre coincide con quella dei linguaggi regolari.

Linguaggi deterministici

Si definiscono linguaggi deterministici i linguaggi non contestuali riconoscibili da un automa push-down deterministico.

Siano:

L_{cf} l'insieme dei linguaggi non contestuali
L_d l'insieme dei linguaggi deterministici
L_r l'insieme dei linguaggi regolari

Si ha: $L_{cf} \supset L_d \supset L_r$

(• il linguaggio $L_d = \{ a^n b^n \mid n > 0\}$ è deterministico ma non regolare;
• il linguaggio $L_d = \{ a^n b^n \mid n > 0\} \cup \{ a^n b^{2n} \mid n > 0\}$ è non contestuale ma non deterministico)

Proprietà:

- la famiglia dei linguaggi deterministici non è chiusa rispetto all'unione, all'intersezione ed alla concatenazione;
- il complemento di un linguaggio deterministico è deterministico;
- se L_d è deterministico e L_r è regolare, L_d/L_r e L_dL_r sono deterministici;

Analisi sintattica (parsing)

Si definiscono due metodologie principali:

- analisi ascendente (bottom-up): si costruisce la derivazione canonica destra della stringa procedendo dalle foglie verso la radice dell'albero sintattico (ordine riflesso);
- analisi discendente (top-down): si costruisce la derivazione canonica sinistra;

Analisi bottom-up LR(0)

Sia $Z \underset{d}{\Rightarrow}^{*} \beta A \gamma \underset{d}{\Rightarrow} \beta\alpha\gamma$ una derivazione canonica destra, con $\langle A ::= \alpha\rangle \in P$, $\alpha = \alpha_1 \alpha_2$. La stringa $\beta\alpha_1$ è un prefisso ascendente e la stringa $\langle A ::= \alpha_1 {}^{\nabla}\alpha_2\rangle$ è detta candidata per $\beta\alpha_1$.

Dato il prefisso ascendente α, $C(\alpha)$ è l'insieme delle candidate per α.

Data una grammatica G, l'insieme dei prefissi ascendenti costituisce un linguaggio regolare $\mathbf{L}_p(G)$ di alfabeto $\Sigma_p = V_T \cup V_N$ (riconoscibile mediante un automa a stati finiti).

Dati due insiemi di candidate C e C', C' è detto estensione di C (denotato C |- C') se e solo se:
$C' = C \cup \{\langle B ::= {}^{\nabla}\beta\rangle \mid \langle B ::= \beta\rangle \in P \text{ e } \langle A ::= \alpha_1 {}^{\nabla} B \alpha_2\rangle \in C\}$

Dato l'insieme di candidate C si definisce chiusura di C l'insieme $C^{\oplus}$ tale che:
$C \vdash^{+} C^{\oplus}$ e $C^{\oplus} = C'$, $\forall C' : C^{\oplus} \vdash C'$

Data la grammatica $G = \langle V_T, V_N, P, Z\rangle$, l'automa $\mathbf{LR}_0(G) = \langle Q_p, \Sigma_p, t_p, q_{p0}\rangle$ riconoscitore del linguaggio $\mathbf{L}_p(G)$ con prospezione 0 è definito nel modo seguente:

- se $p \in Q_p$, $h \in \Sigma_p$ e $C = \{ \langle A ::= \alpha_1 {}^{\nabla} h \alpha_2\rangle \mid \langle A ::= \alpha_1 {}^{\nabla} h \alpha_2\rangle \in p \}$, allora $\exists q \in Q_p$: $q = K^{\oplus}$, dove $K = \{ \langle A ::= \alpha_1 h {}^{\nabla} \alpha_2\rangle \mid \langle A ::= \alpha_1 {}^{\nabla} h \alpha_2\rangle \in C \}$ e $t_p(p,h) = q$.
- se $C = \{ \langle Z ::= {}^{\nabla}\alpha\rangle \mid \langle Z ::= \alpha\rangle \in P\}$, $q_{p0} = C^{\oplus}$.

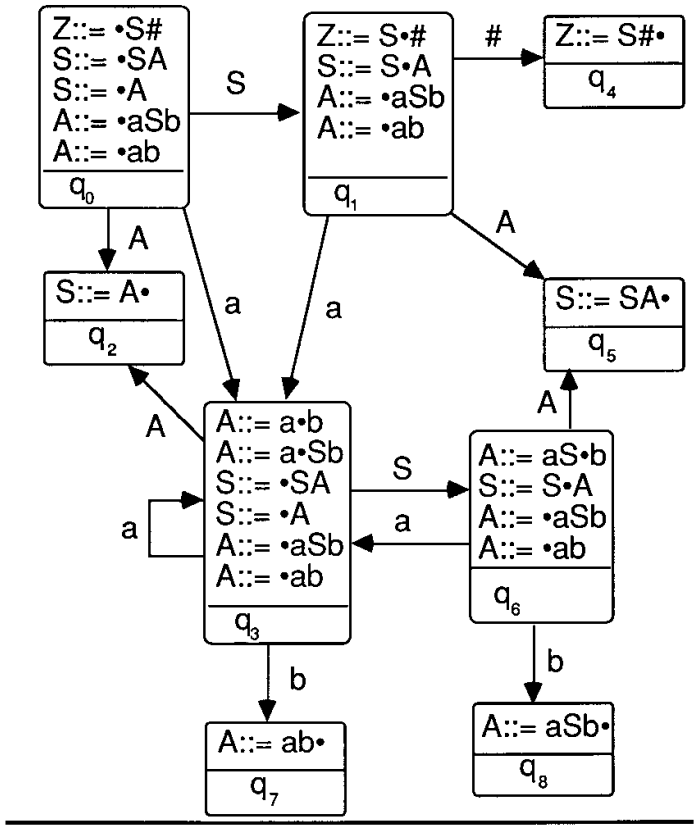
Dato l'automa $\mathbf{LR}_0(G)$ uno stato q si definisce:

- stato di riduzione se contiene la candidata di riduzione $\langle A ::= \alpha {}^{\nabla}\rangle$
- stato di spostamento se contiene la candidata di spostamento $\langle A ::= \alpha_1 {}^{\nabla} \alpha_2\rangle$

Uno stato di riduzione che contenga più di una candidata, oppure uno stato che sia contemporaneamente di spostamento e di riduzione è detto stato inadeguato per l'analisi LR(0).

Esempio

Data la grammatica G_1 con P = { Z::= S#, S::= SA | A, A::= aSb | ab } e assioma Z, l'automa $\mathbf{LR}_0(G_1)$ è:



Nessuno stato è inadeguato.

Una grammatica G è detta LR(0) (left-to-right con prospezione 0) se l'automa $\mathbf{LR}_0(G)$ non contiene stati inadeguati.

Data una grammatica G di tipo LR(0), l'automa $\mathbf{A}_{LR(0)}(G) = \langle Q, I, \Gamma, t, Z_0, q_0 \rangle$ riconoscitore del linguaggio $\mathbf{L}(G)$ è definito nel modo seguente:

- $Q = \{q_0, q_F\}$
- $I = V_T$
- $\Gamma = Q_p \cup \Sigma_p$
- $Z_0 = q_{p0}$
- $\langle q_0, a\alpha, h_i\gamma \rangle \vdash \langle q_0, \alpha, h_k a h_i \gamma \rangle$ se $h_i \in Q_p$ è uno stato di spostamento e $t_p(h_i, a) = h_k$
- $\langle q_0, \alpha, h_{i_n} b_n \ldots h_{i_1} b_1 h_{i_0} \rangle \vdash \langle q_0, \alpha, h_k A h_{i_0} \rangle$ se $h_{i_n} \in Q_p$ è uno stato di riduzione con candidata $\langle A ::= b_1 b_2 \ldots b_n^{\nabla} \rangle$ $(A \neq Z)$ e $t_p(h_{i_0}, A) = h_k$
- $\langle q_0, \varepsilon, h_{i_n} b_n \ldots h_{i_1} b_1 Z_0 \rangle \vdash \langle q_F, \varepsilon, Z_0 \rangle$ se $h_{i_n} \in Q_p$ è uno stato di riduzione con candidata $\langle Z ::= b_1 b_2 \ldots b_n^{\nabla} \rangle$ Z è l'assioma di G.

L'insieme $L_{LR(0)}$ dei linguaggi generati da grammatiche LR(0) e' costituito da linguaggi deterministici privi di prefissi (se $\mathbf{L} \in L_{LR(0)}$ e $\alpha \in \mathbf{L}$, $\alpha\beta \notin \mathbf{L}$ per $\forall \beta \neq \varepsilon$); quindi $L_d \supset L_{LR(0)}$

Ogni linguaggio deterministico può essere trasformato in un equivalente linguaggio privo di prefissi terminando ogni frase con la marca di fine stringa "#". *Tuttavia una grammatica LR(0) può risultare complessa ed inefficiente dal punto di vista del trattamento degli errori* .

Analisi bottom-up LR(1)

Ad ogni candidata LR(0) viene associato un insieme di simboli terminali detto insieme di prospezione . Una candidata LR(1) ha quindi la forma:

$$\langle A ::= \alpha_1 {}^{\nabla}\alpha_2, \{a_1 , a_2 , \dots a_n\}\rangle \qquad a_i \in V_T \cup \{\#\}$$

dove $\{a_1 , a_2 , \dots a_n\}$ è l'insieme di prospezione.

$\langle A ::= \alpha_1 {}^{\nabla}\alpha_2, \Pi\rangle$ è candidata di un prefisso ascendente $\beta\alpha_1$ se per ogni $a_i \in \Pi$ esistono le derivazioni destre:

$$Z \underset{d}{\Rightarrow}{}^{*} \beta A \gamma \underset{d}{\Rightarrow} \beta\alpha_1\alpha_2\gamma \quad \text{e} \quad \gamma = a_i\delta \quad (\gamma = \varepsilon \text{ se } a_i = \#)$$

L'automa $\mathbf{LR}_1(G)$ riconoscitore dei prefissi ascendenti può essere calcolato in modo analogo al caso LR(0) :

- se $p \in Q_p$, $h \in \Sigma_p$ e $C = \{ \langle A ::= \alpha_1 {}^{\nabla}h\alpha_2, \Pi\rangle \mid \langle A ::= \alpha_1 {}^{\nabla}h\alpha_2, \Pi\rangle \in p \}$, allora $\exists q \in Q_p$: $q = K^{\oplus}$, dove $K = \{ \langle A ::= \alpha_1 h {}^{\nabla}\alpha_2, \Pi\rangle \mid \langle A ::= \alpha_1 {}^{\nabla}h\alpha_2, \Pi\rangle \in C \}$ e $t_p(p,h) = q$.
- se $C = \{ \langle Z ::= {}^{\nabla}\alpha, \{\#\}\rangle \mid \langle Z ::= {}^{\nabla}\alpha\rangle \in P \}$, $q_{p0} = C^{\oplus}$.

ridefinendo il concetto di estensione nel modo seguente:

- dati due insiemi di candidate C e C', C' è detto estensione di C (denotato C |- C') se e solo se:

$$C' = C \cup \{\langle B ::= {}^{\nabla}\beta, \Pi'\rangle \mid \langle B ::= \beta\rangle \in P \text{ e } \langle A ::= \alpha_1 {}^{\nabla}B\alpha_2, \{a_1, a_2, \dots a_n\}\rangle \in C\}$$

dove: $b \in \Pi'$ se esiste la derivazione $\alpha_2 \underset{d}{\Rightarrow}{}^{+} b\gamma$

$a_i \in \Pi'$ se esiste la derivazione $\alpha_2 \underset{d}{\Rightarrow}{}^{+} \varepsilon$

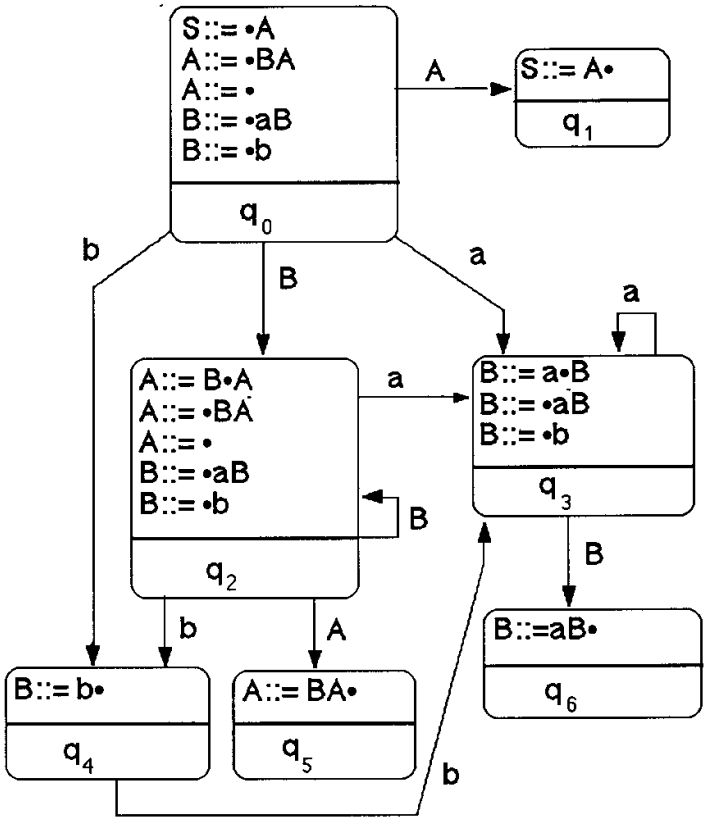
Una grammatica G è detta LR(1) se per ogni stato q dell'automa $\mathbf{LR}_1(G)$ contenente una candidata di riduzione $\langle A ::= \alpha^{\nabla}, \Pi_1\rangle$ valgono entrambe le seguenti condizioni:

- t(q,a) non è definito per alcun $a \in \Pi_1$
- se in q vi è un'altra candidata di riduzione $\langle B ::= \beta^{\nabla}, \Pi_2\rangle$, $\Pi_1 \cap \Pi_2 = \Phi$

Uno stato che non soddisfa questa condizione è detto inadeguato per l'analisi LR(1).

Esempio

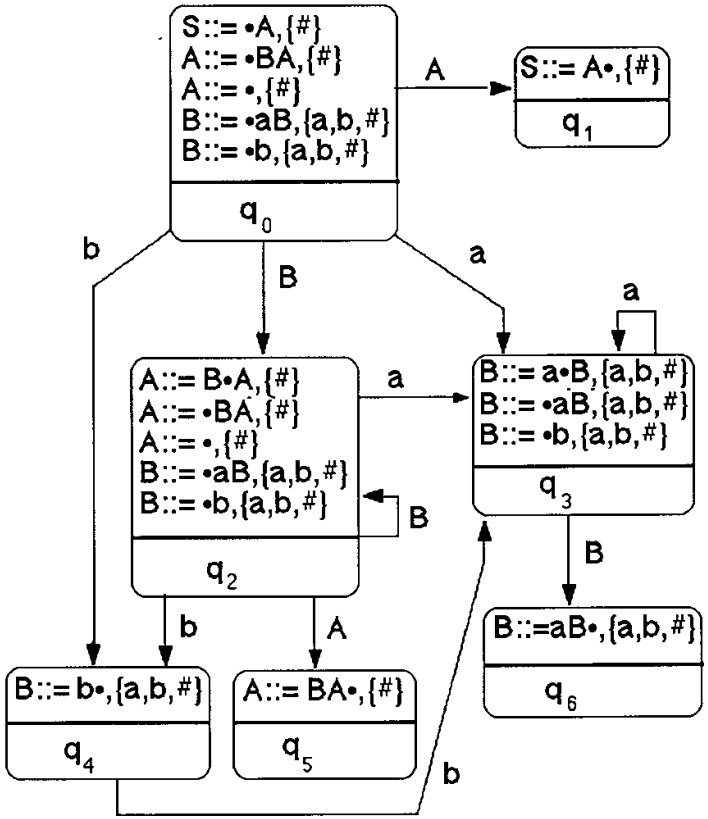
Data la grammatica G_2 con P = { S::= A, A::= BA | ε, B::= aB | b } e assioma S, l'automa $\mathbf{LR}_0(G_2)$ è:



Gli stati q_0 e q_2 sono inadeguati nell'analisi LR(0)

Esempio

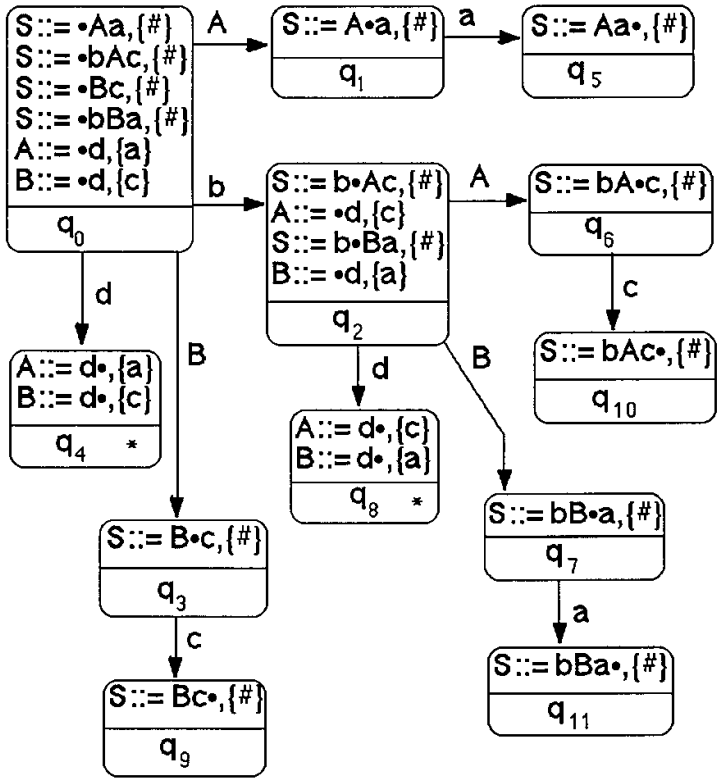
automa $\mathbf{LR}_1(G_2)$ con insiemi di prospezione per analisi LR(1)



Gli stati q_0 e q_2 sono adeguati per l'analisi LR(1)

Esempio

Data la grammatica G_3 con P = { S::= Aa | bAc | Bc | bBa, A::= d, B::= d } e assioma S, l'automa $\mathbf{LR}_1(G_3)$ è:



NOTA: l'automa per il riconoscimento LR(0) non distinguerebbe gli stati q_4 e q_8

Data una grammatica G di tipo LR(1), l'automa $\mathbf{A}_{LR(1)}(G) = \langle Q,I,\Gamma,t,Z_0,q_0\rangle$ riconoscitore del linguaggio $\mathbf{L}(G)$ è definito nel modo seguente:

- $Q = \{q_0,q_F\}$
- $I = V_T$
- $\Gamma = Q_p \cup \Sigma_p$
- $Z_0 = q_{p0}$
- $\langle q_0, a\alpha, h_i\gamma\rangle$ |- $\langle q_0, \alpha, h_k a h_i\gamma\rangle$ se $h_i \in Q_p$ contiene la candidata $\langle A ::= \beta_1{}^{\nabla} a\beta_2,\Pi\rangle$ e $t_p(h_i,a) = h_k$
- $\langle q_0, a\alpha, h_{i_n} b_n \ldots h_{i_1} b_1 h_{i_0}\rangle$ |- $\langle q_0, a\alpha, h_k A h_{i_0}\rangle$ se $h_{i_n} \in Q_p$ contiene la candidata $\langle A ::= b_1 b_2 \ldots b_n{}^{\nabla},\Pi\rangle$ $(A \neq Z)$, $t_p(h_{i_0},A) = h_k$ e $a \in \Pi$
- $\langle q_0, \#, h_{i_n} b_n \ldots h_{i_1} b_1 Z_0\rangle$ |- $\langle q_F, \#, Z_0\rangle$ se $h_{i_n} \in Q_p$ contiene la candidata $\langle Z ::= b_1 b_2 \ldots b_n{}^{\nabla},\{\#\}\rangle$ Z è l'assioma di G.

L'insieme $L_{LR(1)}$ dei linguaggi generati da grammatiche LR(1) coincide con l'insieme dei linguaggi deterministici.

Analisi bottom-up SLR(1)

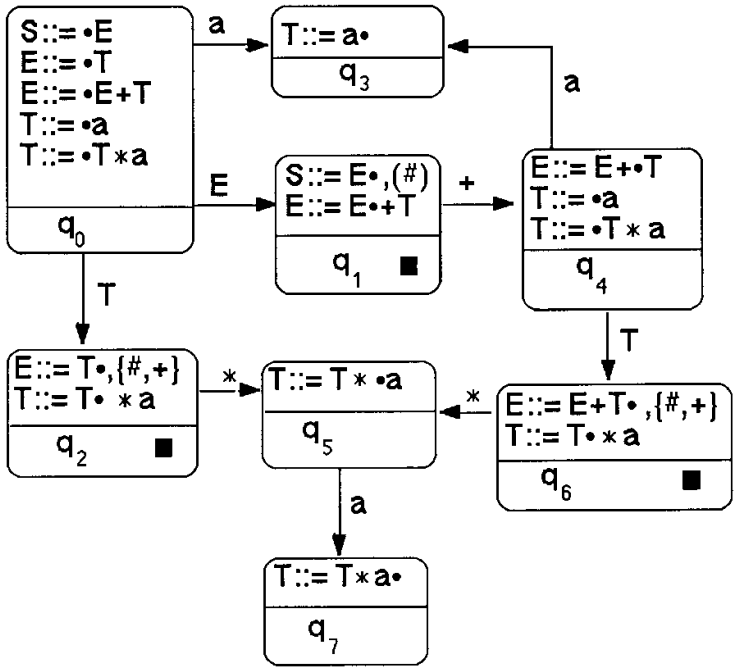
Sia A un simbolo nonterminale della grammatica G; si definisce l'insieme $\mathbf{S}(A)$ dei seguiti di A nel modo seguente:

$\mathbf{S}(A) = \{ a \in V_T \mid Z \Rightarrow^* \alpha A a \beta \}$, $\quad \# \in \mathbf{S}(A)$ se $Z \Rightarrow^* \alpha A$

Per ogni candidata $\langle A ::= \alpha^{\nabla}, \Pi \rangle$ si avrà $\Pi = \mathbf{S}(A)$. L'insieme di prospezione è calcolato solo per le candidate di riduzione appartenenti a stati inadeguati per l'analisi LR(0).

Esempio

Data la grammatica G_4 con P = { S::= E, E::= T | E+T, T::= a | T* a } e assioma S, l'automa $\mathbf{SLR}_1(G_4)$ è:



Gli stati q_1, q_2, q_6 sono inadeguati per l'analisi LR(0) e adeguati per l'analisi SLR(1).

Una grammatica è detta SLR(1) se per ogni candidata di riduzione $\langle A::= \alpha^{\nabla}, \mathbf{S}(A) \rangle$ in uno stato q inadeguato per l'analisi LR(0) valgono le seguenti condizioni:

- in q non vi è un'altra candidata $\langle B::= \beta^{\nabla} a\delta \rangle$ con $a \in \mathbf{S}(A)$
- se in q vi è un'altra candidata $\langle B::= \alpha^{\nabla}, \mathbf{S}(B) \rangle$, $\mathbf{S}(A) \cap \mathbf{S}(B) = \Phi$

Per il calcolo di $\mathbf{S}(A)$ si possono usare le seguenti regole:

- $\# \in \mathbf{S}(Z)$, dove Z è l'assioma
- $a \in \mathbf{S}(A)$ se $\langle B::= \alpha A \beta \rangle \in P$ e $a \in \mathbf{I}(\beta)$
- $a \in \mathbf{S}(A)$ se $\langle B::= \alpha A \rangle \in P$ $(A \neq B)$ e $a \in \mathbf{S}(B)$
- $a \in \mathbf{S}(A)$ se $\langle B::= \alpha A \beta \rangle \in P$ $(A \neq B)$, $\beta \Rightarrow^{+} \varepsilon$ e $a \in \mathbf{S}(B)$

dove $\mathbf{I}(\beta) = \{b \in V_T \mid \beta \Rightarrow^{*} b\gamma\}$ (insieme degli inizi di β) è calcolabile come segue:

- $a \in \mathbf{I}(\beta)$ se $\beta = a\gamma$
- $a \in \mathbf{I}(\beta)$ se $\beta = A\gamma$, $\langle A::= \delta \rangle \in P$ e $a \in \mathbf{I}(\delta)$
- $a \in \mathbf{I}(\beta)$ se $\beta = A\gamma$, $A \Rightarrow^{+} \varepsilon$ e $a \in \mathbf{I}(\gamma)$

Analisi bottom-up LALR(1)

Per ogni candidata $\langle A::= \alpha^{\nabla}, \Pi \rangle$ appartenente allo stato q si avrà:

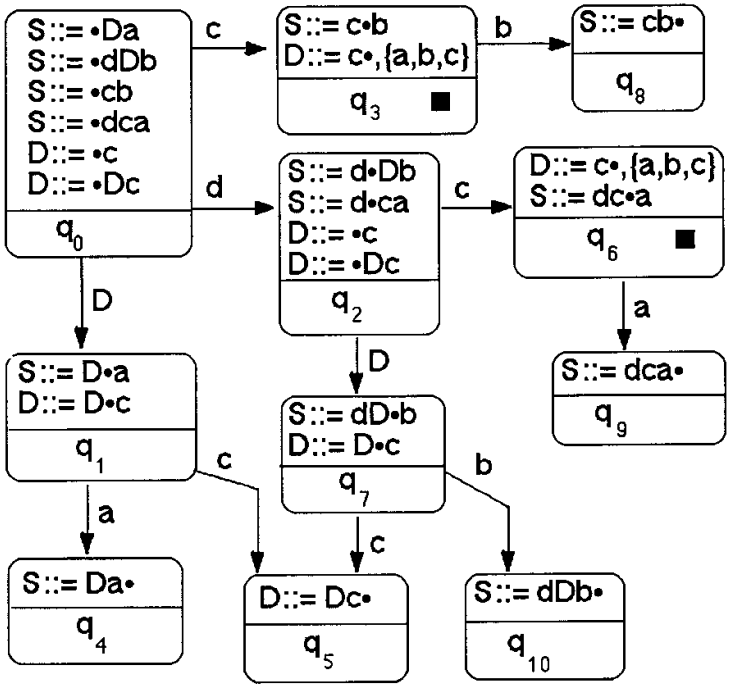
$\Pi = \{a \in V_T \mid Z \Rightarrow_d^{*} \gamma A a \beta \Rightarrow \gamma\alpha a\beta$ e $t_p(q_{p0}, \gamma\alpha) = q\}$

dove t_p è la funzione di transizione di $\mathbf{LR}_0(G)$.

L'insieme di prospezione è calcolato solo per le candidate di riduzione appartenenti a stati inadeguati per l'analisi LR(0).

Esempio

Data la grammatica G_5 con P = { S::= Da | dDb | cb | dca, D::= c | Dc } e assioma S, l'automa $\mathbf{SLR}_1(G_5)$ è:



Gli stati q_3 e q_6 sono inadeguati per l'analisi LR(0) e SLR(1).

Una grammatica è detta LALR(1) se per ogni candidata di riduzione $\langle A::= \alpha^{\nabla}, \Pi_A \rangle$ in uno stato q inadeguato per l'analisi LR(0) valgono le seguenti condizioni:

- in q non vi è un'altra candidata $\langle B::= \beta^{\nabla} a\delta \rangle$ con $a \in \Pi_A$
- se in q vi è un'altra candidata $\langle B::= \alpha^{\nabla}, \Pi_B \rangle$, $\Pi_A \cap \Pi_B = \Phi$

Sia G_X l'insieme delle grammatiche di tipo x; si ha:

- $G_{LR(1)} \supset G_{LALR(1)} \supset G_{SLR(1)} \supset G_{LR(0)}$

Analisi top-down LL(1)

Data una grammatica G e una produzione $\langle A::= \alpha \rangle$ si definisce insieme guida di $\langle A::= \alpha \rangle$ l'insieme:

$\mathbf{G}(\langle A::= \alpha \rangle) = \mathbf{I}(\alpha)$, se non esiste $\alpha \Rightarrow^* \varepsilon$

$\mathbf{G}(\langle A::= \alpha \rangle) = \mathbf{I}(\alpha) \cup \mathbf{S}(A)$, se esiste $\alpha \Rightarrow^* \varepsilon$

Una grammatica G è LL(1) se, per ogni coppia di produzioni $\langle A::= \alpha \rangle$ e $\langle A::= \beta \rangle$:

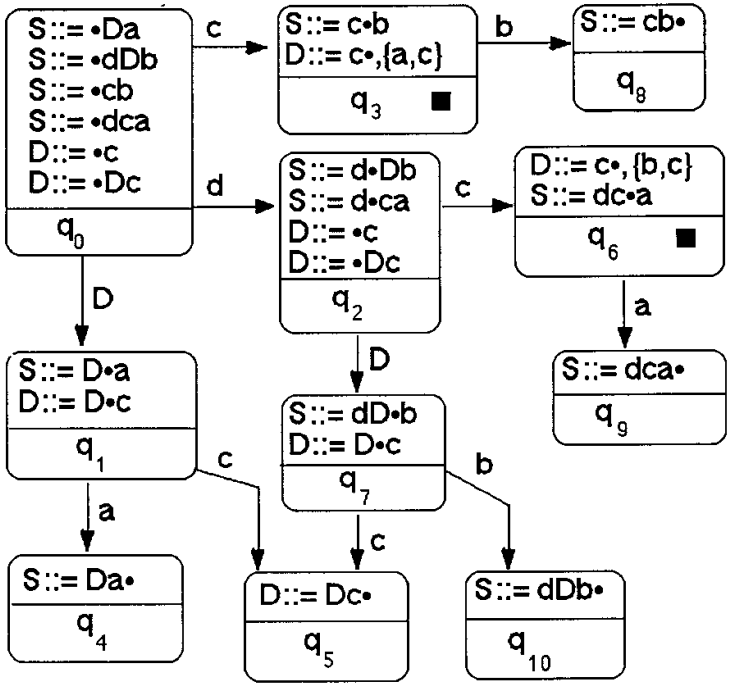
$\mathbf{G}(\langle A::= \alpha \rangle) \cap \mathbf{G}(\langle A::= \beta \rangle) = \Phi$

NOTA: una grammatica LL(1) non può avere produzioni ricorsive a sinistra.

Le grammatiche LL(1) sono un sottoinsieme delle grammatiche LR(1): $G_{LR(1)} \supset G_{LL(1)}$

Esempio

Data la grammatica G_5 con P = { S::= Da | dDb | cb | dca, D::= c | Dc } e assioma S, l'automa $\mathbf{LALR}_1(G_5)$ è:



Gli stati q_3 e q_6 sono adeguati per l'analisi LALR(1).

Data una grammatica G di tipo LL(1), l'automa $\mathbf{A}_{LL(1)}(G) = \langle Q,I,\Gamma,t,Z_0,q_0\rangle$ riconoscitore del linguaggio $\mathbf{L}(G)$ è definito nel modo seguente:

- $Q = \{q_0,q_1,q_F\}$
- $I = V_T$
- $\Gamma = V_T \cup V_N$
- $\langle q_0, \alpha, Z_0\rangle$ |- $\langle q_1, \alpha, ZZ_0\rangle$, dove Z è l'assioma di G
- $\langle q_1, a\alpha, A\gamma\rangle$ |- $\langle q_1, a\alpha, \beta\gamma\rangle$ se $\langle A::= \beta\rangle$ è una produzione, $a \in \mathbf{G}(\langle A::= \beta\rangle)$
- $\langle q_1, a\alpha, a\gamma\rangle$ |- $\langle q_1, \alpha, \gamma\rangle$
- $\langle q_1, \#, Z_0\rangle$ |- $\langle q_F, \#, Z_0\rangle$

Esempio

Le seguenti grammatiche sono LL(1):

G_6 = { S::= Ab [a,b], S::= cS [c], A::= aA [a], A::= ε [b] }
G_7 = { S::= aBb [a], S::= BS [bc], B::= b [b], S::= c [c] }

Alcune grammatiche non LL(1) possono essere trasformate in grammatiche LL(1) debolmente equivalenti con operazioni di:

- fattorizzazione sinistra
- sostituzione sinistra
- eliminazione di ricorsione a sinistra

Trattamento degli errori sintattici

Obiettivi:

- consentire la prosecuzione dell'analisi del testo
- suggerire correzioni

Qualunque trasformazione di una stringa in un'altra può essere ottenuta mediante una o più applicazioni delle seguenti operazioni di edizione :

- cancellazione del simbolo x nella posizione i : $[x]_{i-}$
- inserimento del simbolo x nella posizione i : $[x]_{i+}$
- sostituzione di x con y nella posizione i : $[x/y]_i$
- scambio di due simboli adiancenti nelle posizioni i e i+1: $[x \leftrightarrow y]_i$

Date due stringhe α e β , si definisce distanza di edizione $d(\alpha,\beta)$ il numero di operazioni di edizione necessario per trasformare α in β (eventualmente si attribuisce un peso ad ogni operazione e si considera la somma pesata).

Esempio : α = "loop", β = "loot", γ = "cloo", δ = "loto"

- $d(\alpha,\beta) = 1$: $[p/t]_4$
- $d(\alpha,\gamma) = 2$: $[c]_{1+}$, $[p]_{5-}$
- $d(\alpha,\delta) = 2$: $[p/t]_4$, $[o \leftrightarrow t]_3$ oppure $[o/t]_3$, $[p/o]_4$

Si definisce correzione a distanza minima (c.d.m) la sostituzione di una stringa errata (i.e. non appartenente al linguaggio) con una scelta fra quelle corrette a distanza minima.

La c.d.m. è adeguata per la correzione di errori ortografici (individuazione della parola corretta dato il dizionario) e può essere utilizzata in fase di analisi lessicale.

Un analizzatore sintattico in grado di effettuare la c.d.m. risulta in genere non deterministico, con complessità $O(n^3)$, quindi non accettabile.

Correzione con analizzatori deterministici

Si definisce errore di prefisso , nell'analisi da sinistra a destra, il primo simbolo che fa sì che la parte di stringa già analizzata non sia prefisso di alcuna frase del linguaggio.

Esempio

- frase corretta: IF A = B THEN .....
- frase errata: IFA = B THEN
  prefisso valido: IFA = B
  errore di prefisso: THEN .....

Gli analizzatori deterministici segnalano l'errore di prefisso e tentano una correzione locale.

Analisi deterministica bottom-up

Situazione dell'analizzatore sintattico al momento della segnalazione dell'errore:

- contenuto dello stack: $Z_0a_1h_1a_2h_2 \ldots a_nh_n$
- stringa residua in ingresso: $b_0b_1 \ldots \#$
- non esiste transizione $t_p(h_n,b_0)$
- se l'analizzatore ha prospezione > 0, b_0 non appartiene al contesto di alcuna riduzione

Strategia 1:

si rimuovono (se necessario) gli elementi dallo stack fino ad incontrare uno stato h_i tale che esista la transizione $t_p(h_i,A) = h_k$, con A non terminale, e da h_k possa proseguire l'analisi con b_0; si simula il riconoscimento di A e si prosegue. Se non esiste alcun stato h_i, si ripristina lo stack e si rimuove b_0, ripetendo l'operazione precedente con il simbolo b_1, e così via fino alla fine della stringa.

Strategia 2:

si inserisce prima di b_0 un simbolo terminale a tale che esista la transizione $t_p(h_i,a) = h_k$ e da h_k possa proseguire l'analisi con b_0

Strategia 3:

si aggiungono alla grammatica una o più produzioni di errore, della forma:

A ::= ⟨error⟩ a

dove ⟨error⟩ è considerato terminale e $a \in V_T \cup \{\varepsilon\}$. Nel corso dell'analisi, in presenza di un errore si procede come segue:

- viene inserito ⟨error⟩ prima di b_0;
- si rimuovono (se necessario) gli elementi dallo stack fino ad incontrare uno stato h_i tale che esista la transizione $t_p(h_i,\langle error\rangle) = h_k$
- si rimuovono (se necessario) i simboli della stringa in ingresso fino ad incontrare il simbolo $b_i = a$
- si effettua la riduzione secondo la produzione di errore e si rimuovono (se necessario) i simboli della stringa in ingresso fino ad incontrare un simbolo che consenta di riprendere l'analisi.

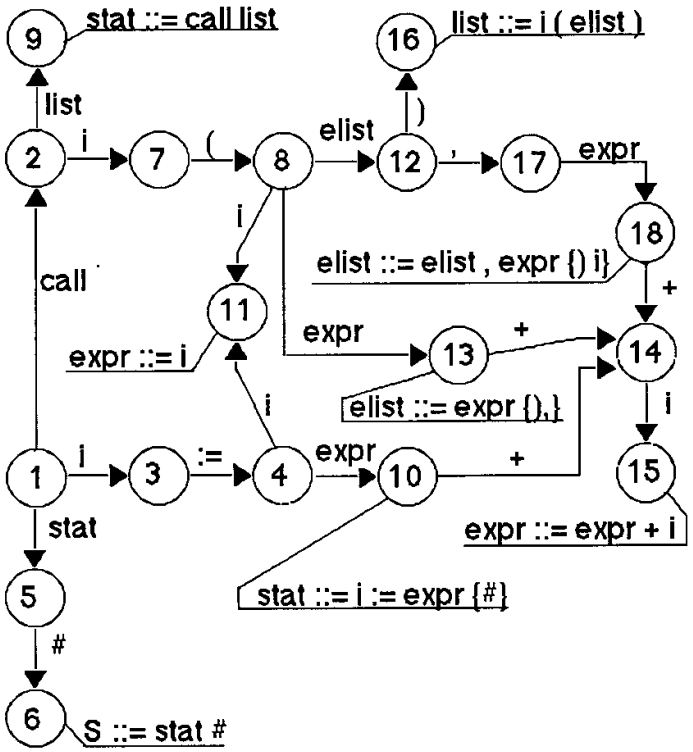
Analisi deterministica LL(1)

Dato l'automa $\mathbf{A}_{LL(1)}(G)$ si possono avere le seguenti situazioni di errore:

- la configurazione corrente di $\mathbf{A}_{LL(1)}(G)$ è $\langle q_1, b_0b_1....\#, A\gamma\rangle$, $b_0 \notin \mathbf{G}(\langle A ::= \beta\rangle)$ per qualunque produzione avente A come parte sinistra.
- la configurazione corrente di $\mathbf{A}_{LL(1)}(G)$ è $\langle q_1, b_0b_1....\#, a\gamma\rangle$, $b_0 \notin a$.

Esempio

Data la grammatica G_8 con P = { S::= Stat #; Stat::= call List | i ← expr; List::= i (Elist); Elist::= Elist , Expr | Expr; Expr::= Expr + i | i } l'automa $LARL_1(G_8)$ è:



L'automa $A_{LARL(1)}(G_8)$ opera le seguenti correzioni:

- "call i := i + i + i #" ⇒ "call i (?) #" (strategia 1)
- "call i ( ) #" ⇒ "call i (?) #" (strategia 1)
- ":= i + i + i #" ⇒ "i := i + i + i #" (strategia 2)

Strategia dell'albero connesso: si rimuovono i simboli dallo stack fino ad incontrarne uno che consenta di proseguire l'analisi con b_0; se non esiste, si ripristina lo stack e si rimuove b_0 , ripetendo l'operazione precedente con il simbolo b_1, e così via fino alla fine della stringa.

Esempio. Data la grammatica G_9 :

S::= Stat #{call i}
Stat::= call List {call} | i ::= expr {i}
List::= i (elist) {i}
Elist::= Expr Tail {i}
Tail::= , Expr Tail {,} | ε {)}
Expr::= i Right {i}
Right::= + Expr {+} | ε {,) #}

L'automa $A_{LL(1)}(G_9)$ opera le seguenti correzioni:

- "call i := i + i + i #" ⇒ "call i (i + i + i) #"
- "call i (i) i + i + i) #" ⇒ "call i (i) #"

Sincrotriple

Data una stringa β si definisce l'insieme delle fini $\mathbf{F}(\beta) = \{b \in V_T \mid \beta \Rightarrow^* \gamma b, \gamma \in (V_T \cup V_N)^*\}$

Per ogni nonterminale A si definisce un insieme $\mathbf{MA}(A)$ di marche di apertura ed un insieme $\mathbf{MC}(A)$ di marche di chiusura , con $\mathbf{I}(A) \supseteq \mathbf{MA}(A)$, $\mathbf{F}(A) \supseteq \mathbf{MC}(A)$.

Si definisce sincrotripla la terna $\langle b,c,A\rangle$, con $A \in V_N$, $b \in \mathbf{MA}(A)$, $c \in \mathbf{MC}(A)$.

Metodo delle sincrotriple: ad ogni sottoalbero dell'albero sintattico può essere associata una lista di sincrotriple attivabili, detta sincrotabella. L'analizzatore sintattico può trovarsi in uno *stato di analisi* o in uno stato di errore: entrambi gli stati possono essere attivati ricorsivamente.

Stato di analisi attivato dalla marca a_a della sincrotripla $\langle a_a,a_c,A\rangle$:

- se incontra la marca a_c esce dallo stato di analisi e ritorna nello stato di errore precedente (termina se non esistono stati di errore precedenti)
- se incontra un errore e b_a è l'ultima marca di apertura incontrata, attiva un stato di errore con sincrotripla corrente $\langle b_a,b_c,B\rangle$.

Stato di errore; ignora tutti i simboli in ingresso finché:

- incontra la marca di chiusura della sincrotripla corrente e ritorna nello stato di analisi precedente
- incontra la marca di apertura della sincrotripla $\langle c_a,c_c,C\rangle$ ed attiva un nuovo stato di analisi
- incontra il simbolo '#' e termina

Vincoli sulla definizione delle sincrotriple della stessa sincrotabella

1) due sincrotriple non possono avere la stessa marca di apertura
2) la marca di apertura di una sincrotripla non può essere marca di chiusura della stessa o di un'altra sincrotripla
3) due sincrotriple non possono essere parzialmente sovrapposte: data le sincrotriple $\langle a_a,a_c,A\rangle$ e $\langle b_a,b_c,B\rangle$, per ogni stringa α tale che $A \Rightarrow^* a_a\alpha a_c$ $b_a \in \alpha$ se e solo se $b_c \in \alpha$

Traduzioni sintattiche

Dati due linguaggi $\mathbf{L}_1$ e $\mathbf{L}_2$, la traduzione da $\mathbf{L}_1$ (linguaggio sorgente) a $\mathbf{L}_2$ (linguaggio oggetto) può essere definita come una relazione binaria $\mathbf{L}_1\ \tau\ \mathbf{L}_2$.

Date due grammatiche G_1 e G_2 , tali che:

- $V_{N1} = V_{N2} = V_N$, $Z_1 = Z_2 = Z$
- per ogni $\langle A ::= \alpha_1 \rangle \in P_1$ esiste una $\langle A ::= \alpha_2 \rangle \in P_2$ tale che $\pi_{V_N}(\alpha_1) = \pi_{V_N}(\alpha_2)$ [dove $\pi_{V_N}(\alpha)$ è la proiezione di α su V_N] e viceversa. Le coppie di produzioni $\langle A ::= \alpha_1 , A ::= \alpha_2 \rangle$ si dicono corrispondenti.

L'insieme delle coppie di produzioni corrispondenti costituisce uno schema di traduzione .

Le due grammatiche possono essere fuse in un'unica grammatica di traduzione $G_t = \langle V_{T1} \cup V_{T2} , V_N, P_t, Z \rangle$ tale che:

- $P_1 = \pi_{V_1}(P_t)$, $P_2 = \pi_{V_2}(P_t)$
- $\mathbf{L}(G_1) = \{ \alpha \mid \gamma \in \mathbf{L}(G_t)$ e $\alpha = \pi_{V_1}(\gamma) \}$
- $\mathbf{L}(G_2) = \{ \alpha \mid \gamma \in \mathbf{L}(G_t)$ e $\alpha = \pi_{V_2}(\gamma) \}$
- $\beta = \tau(\alpha)$ se e solo se esiste $\gamma \in \mathbf{L}(G_t)$ tale che: $\gamma \in \mathbf{L}(G_t)$, $\alpha = \pi_{V_1}(\gamma)$, $\beta = \pi_{V_2}(\gamma)$

dove $V_1 = V_{T1} \cup V_N$ e $V_2 = V_{T2} \cup V_N$

Esempio :

Schema di traduzione — Grammatica di trad.

G_1	G_2	G_t
E::= E + T	E::= ⊕ E T	E::= ⊕ E + T
E::= T	E::= T	E::= T
T::= T * F	T::= ⊗ T F	T::= ⊗ T * F
T::= F	T::= F	T::= F
F::= (E)	F::= E	F::= (E)
F::= i	F::= id	F::= id i

Traduzioni postfisse

Uno schema di traduzione è detto postfisso se le produzioni della grammatica oggetto sono del tipo:

$A ::= \gamma\omega$ dove $\gamma \in (V_{T1} \cup V_N)^*$, $\omega \in V_{T2}^*$

Dato uno schema di traduzione non postfisso, è sempre possibile definirne uno equivalente postfisso; la produzione $A ::= \alpha b \beta$, con $\alpha, \beta \in (V_{T1} \cup V_{T2} \cup V_N)^*$, $b \in V_{T2}$, è equivalente alla coppia: $A ::= \alpha B \beta$, $B ::= \varepsilon b$

Traduttori deterministici bottom-up

Richiedono grammatiche di traduzione basate su schemi postfissi. Un traduttore si ottiene modificando un analizzatore come segue:

- in corrispondenza dell'operazione di riduzione secondo la candidata $A ::= \gamma^{\nabla}\omega$, dove $\gamma \in (V_{T1} \cup V_N)^*$, $\omega \in V_{T2}^*$ si produce la stringa ω.

Traduttori deterministici top-down

Possono operare anche con schemi di traduzione non postfissi. Un traduttore si ottiene modificando un analizzatore come segue:

- se in cima allo stack vi è un simbolo $b \in V_{T2}$, il traduttore lo estrae dallo stack e lo aggiunge alla stringa di uscita.