Linguaggi Formali e Traduttori

– grammatiche LL(1) e parsificazione top-down –

Alcuni esercizi risolti

Nota: nello pseudo-codice delle funzioni che seguono si intende che il return avviene alla fine del codice (salvo uscite per l'eccezione ERRORE).

1. Eliminare la ricorsione sinistra dalle grammatiche con le produzioni seguenti:

P_1: $S \to R \mid 0$
$R \to RC \mid R0 \mid C$
$C \to 1 \mid 2 \mid 3$

P_2: $S \to Ss\text{-} \mid S\{S\} \mid D \mid \varepsilon$
$D \to TL\text{-}R$
$R \to \varepsilon \mid D$
$L \to L;m \mid m$
$T \to i \mid r$

N.B. {s, -, {, }, ;, m, i, r} è l'insieme dei simboli terminali

Produzioni senza ricorsioni sinistre di grammatiche equivalenti:

P_1': $S \to R \mid 0$
$R \to CR'$
$R' \to CR' \mid 0R' \mid \varepsilon$
$C \to 1 \mid 2 \mid 3$

P_2': $S \to DS' \mid S'$ $S' \to s\text{-}S' \mid \{S\}S' \mid \varepsilon$
$D \to TL\text{-}R$
$R \to \varepsilon \mid D$
$L \to mL'$ $L' \to ;mL' \mid \varepsilon$
$T \to i \mid r$

2. Per ognuna delle seguenti grammatiche, specificate dall'insieme delle produzioni, verificare che sia LL(1) e costruire l'analizzatore a discesa ricorsiva.

P_1: $N \rightarrow D\ K$ $\quad$ P_2: $T \rightarrow S\ T \mid \varepsilon$

$K \rightarrow N \mid \varepsilon$ $\quad$ $S \rightarrow id := E;$

$D \rightarrow 0 \mid 1$ $\quad$ $E \rightarrow id\ G$

$G \rightarrow + id\ G \mid \varepsilon$

Produzioni P_1	Insiemi guida	
$N \rightarrow D\ K$	{0, 1}	
$K \rightarrow N$	{0, 1}	
$K \rightarrow \varepsilon$	{$}	
$D \rightarrow 0$	{0}	
$D \rightarrow 1$	{1}	La grammatica è LL(1)

Analizzatore a discesa ricorsiva:

```
Program discesa_ricorsiva( )
    cc ← PROSS( )
    N( )
    if (cc = '$') "stringa accettata"
    else ERRORE (…)
```

```
function N( )
    if (cc ∈ {'0', '1'})
        D( )
        K( )
    else ERRORE (…)
```

```
function K( )
    if (cc ∈ {'0', '1'})
        N( )
    elseif (cc = '$') do nothing
    else ERRORE (…)
```

```
function D( )
    if (cc = '0')
        cc ← PROSS( )
    elseif (cc = '1')
        cc ← PROSS( )
    else ERRORE (…)
```

Produzioni P_2	Insiemi guida	
$T \rightarrow S\ T$	{id}	
$T \rightarrow \varepsilon$	{$}	
$S \rightarrow id := E;$	{id}	
$E \rightarrow id\ G$	{id}	
$G \rightarrow + id\ G$	{+}	
$G \rightarrow \varepsilon$	{;}	La grammatica è LL(1)

Analizzatore a discesa ricorsiva:

```
Program discesa_ricorsiva( )
    cc ← PROSS( )
    T( )
    if (cc = '$') "stringa accettata"
    else ERRORE (…)

function S( )
  if (cc = 'íd') cc ← PROSS( )
                 if (cc = ':=') cc ← PROSS( )
                 else ERRORE (…)
                 E( )
                 if (cc = ';') cc ← PROSS( )
                 else  ERRORE (…)
  else ERRORE (…)

function G( )
  if (cc = '+')  cc ← PROSS( )
                 if (cc = 'id') cc ← PROSS( )
                 else ERRORE (…)
                 G( )
  elseif  (cc = ';') do nothing
  else ERRORE (…)

function T( )
  if (cc = 'id')
        S( )
        T( )
  elseif  (cc = '$') do nothing
  else ERRORE (…)

function E( )
  if (cc = 'id')
        cc ← PROSS( )
        G( )
  else ERRORE (…)
```

3. Per ognuna delle seguenti grammatiche:

$G_1 = (\{S,A\}, \{a,b,c\}, \{S \rightarrow aAbS \mid c\ A,\ A \rightarrow aScA \mid \varepsilon\}, S)$

$G_2 = (\{S,A\}, \{a,b,c,d\}, \{S \rightarrow aA \mid \varepsilon,\ A \rightarrow bAd \mid S\}, S)$

- Calcolare gli insiemi FIRST e FOLLOW delle variabili;
- Calcolare gli insiemi guida delle produzioni;
- Scrivere il parsificatore a discesa ricorsiva.

Grammatica G_1: $F(S) = F(aAbS) \cup F(cA) = \{a, c\}$ $FW(S) = \{c, \$\}$

$F(A) = F(aScA) \cup F(\varepsilon) = \{a, \varepsilon\}$ $FW(A) = \{b\} \cup FW(S) = \{b, c, \$\}$

	Insieme guida
$S \rightarrow aAbS$	{a}
$S \rightarrow c\ A$	{c}
$A \rightarrow aScA$	{a}
$A \rightarrow \varepsilon$	{b, c, $}

```
function S( )
    if (cc = 'a')  cc ← PROSS( )
                   A( )
                   if (cc = 'b') cc ← PROSS( )
                   else ERRORE (…)
                   S( )
    elseif (cc = 'c') cc ← PROSS( )
                      A (…)
    else ERRORE (…)
```

```
function A( )
    if (cc = 'a')  cc ← PROSS( )
                   S( )
                   if (cc = 'c') cc ← PROSS( )
                   else ERRORE (…)
                   A( )
    elseif  (cc ∈ {'b', 'c', '$'}) do nothing
    else ERRORE (…)
```