

Esercizi su specifica di linguaggi formali e analisi sintattica

Esercizio 1 Scrivere un'espressione regolare e che denota le costanti reali (`double`) nel linguaggio java, prevedendo anche la notazione esponenziale. Si considerino i seguenti esempi: `3.0 E+2`, `-0.33 E-2`, `+5.001`, `.3`.

Soluzione Notiamo che:

- 1) il segno davanti al numero è opzionale
- 2) l'esponenziale posto al termine del numero è opzionale
- 3) deve esserci obbligatoriamente il punto (le costanti senza punto sono costanti intere e non reali)
- 4) deve esserci obbligatoriamente una cifra dopo il punto

L'espressione regolare sarà pertanto:

$$e = (+ | - | \varepsilon)(0 | 1 | 2 \dots | 9)^*(0 | 1 \dots | 9)(0 | 1 \dots | 9)^*(\varepsilon | E(+ | - | \varepsilon)(0 | 1 \dots | 9)^+)$$

che può anche essere scritta come

$$e = (+ | - | \varepsilon)(0 | 1 | 2 \dots | 9)^*(0 | 1 \dots | 9)^+(\varepsilon | E(+ | - | \varepsilon)(0 | 1 \dots | 9)^+)$$

Esercizio 2 Sia L il seguente linguaggio sull'alfabeto $\{a, b, c\}$:

$$L = \{a^n b^m c^n \mid n \geq 0, m \geq 0\}$$

Definire una grammatica non contestuale G tale che $\mathcal{L}(G) = L$.

Soluzione Una grammatica che genera il linguaggio L è la seguente:

$$\begin{aligned} S &\rightarrow aSc \\ S &\rightarrow B \\ S &\rightarrow \epsilon \\ B &\rightarrow bB \\ S &\rightarrow \epsilon \end{aligned}$$

Si osservi anche che non esiste una espressione regolare e tale che $\mathcal{L}(e) = L$.

Esercizio 3 Sia data la grammatica $G = \langle V_T, V_N, S, P \rangle$, in cui

$$P = \begin{cases} S \rightarrow A \\ S \rightarrow cSc \\ A \rightarrow aa \mid bb \end{cases}$$

1. determinare se la stringa $ccccbbcccc$ appartiene a $\mathcal{L}(G)$ e, in caso affermativo, scrivere una derivazione canonica sinistra della stringa.
2. determinare se la stringa $ccaabbcc$ appartiene a $\mathcal{L}(G)$ e, in caso affermativo, scrivere una derivazione canonica sinistra della stringa.

Soluzione Per quanto riguarda la stringa $ccccbbcccc$ possiamo notare che essa risulta ottenibile mediante la seguente derivazione:

$$S \rightarrow cSc \rightarrow ccScc \rightarrow cccSccc \rightarrow ccccScccc \rightarrow ccccAcccc \rightarrow ccccbbcccc \quad (1)$$

che corrisponde all'applicazione della prima produzione per quattro volte, e, successivamente, all'applicazione della seconda e della terza produzione della grammatica G .

Osservando ora la seconda stringa, ovvero $ccaabbcc$, notiamo che il processo di derivazione non ci consente di derivarla: infatti possiamo ottenere le seguenti derivazioni (2) o (3):

$$S \rightarrow cSc \rightarrow ccScc \rightarrow ccAcc \rightarrow ccaacc \quad (2)$$

$$S \rightarrow cSc \rightarrow ccScc \rightarrow ccAcc \rightarrow cbbcc \quad (3)$$

ma non abbiamo modo di derivare la stringa $ccaabbcc$ che, pertanto, non appartiene a $\mathcal{L}(G)$.

Esercizio 4 Sia data la seguente grammatica G :

$$\begin{aligned} \text{START} &\rightarrow \text{"inizio"} \text{ START "fine"} \mid \text{CORPO} \\ \text{CORPO} &\rightarrow \text{PARTE1 PARTE2} \\ \text{PARTE1} &\rightarrow \text{"atomo1"} \text{ PARTE1} \mid \epsilon \\ \text{PARTE2} &\rightarrow \text{"atomo2"} \mid \text{"atomo2"} \text{ PARTE2} \end{aligned}$$

Per ognuna delle seguenti stringhe, dire se la stringa appartiene al linguaggio generato da G , ed in caso affermativo scrivere una derivazione canonica sinistra della stringa:

1. "inizio" "inizio" "atomo1" "atomo1" "atomo2" "atomo2" "fine" "fine"
2. "inizio" "atomo1" "atomo2" "atomo2" "fine" "fine"

Esercizio 5 Sia data la grammatica $G = \langle V_T, V_N, S, P \rangle$ in cui

$$P = \begin{cases} S \rightarrow S'rS' \mid r \\ S' \rightarrow S'iS' \mid f \end{cases}$$

Dire se le seguenti stringhe appartengono a $\mathcal{L}(G)$, ed in caso affermativo scrivere una derivazione canonica sinistra per la stringa: $s_1 = frfif$, $s_2 = fifrfif$, $s_3 = frfifrfif$, $s_4 = fififrfif$,

Soluzione Le stringhe s_1 , s_2 e s_4 appartengono a $\mathcal{L}(G)$, mentre per la terza stringa abbiamo che $s_3 \notin \mathcal{L}(G)$. Di seguito diamo le derivazioni per le stringhe s_1 ed s_2 , lasciando al lettore l'individuazione di una derivazione per s_4 . Una derivazione canonica sinistra per $frfif$ è la seguente:

$$S \rightarrow S'rS' \rightarrow frS' \rightarrow frS'iS' \rightarrow frfiS' \rightarrow frfif$$

Una derivazione canonica sinistra per $fifrfif$ è la seguente:

$$S \rightarrow S'rS' \rightarrow S'iS'rS' \rightarrow fiS'rS' \rightarrow fifrS' \rightarrow fifrS'iS' \rightarrow fifrfiS' \rightarrow fifrfif$$

Esercizio 6 Data la seguente grammatica G :

$$\begin{aligned} S &\rightarrow ASB \mid C \\ A &\rightarrow Aac \mid a \mid aa \\ B &\rightarrow Bb \mid Bbc \mid b \\ C &\rightarrow c \end{aligned}$$

Scrivere una grammatica G' tale che G' non presenti ricorsione sinistra diretta e $\mathcal{L}(G') = \mathcal{L}(G)$.

Soluzione

$$\begin{aligned} S &\rightarrow ASB \mid C \\ A &\rightarrow aA' \mid aaA' \\ A' &\rightarrow \epsilon \mid acA' \\ B &\rightarrow bB' \\ B' &\rightarrow \epsilon \mid bB' \mid bcB' \\ C &\rightarrow c \end{aligned}$$

Esercizio 7 Data la seguente grammatica G :

$$\begin{aligned} S &\rightarrow ASB \mid C \\ A &\rightarrow abAc \mid ab \\ B &\rightarrow bB \mid b \\ C &\rightarrow cDAc \mid cDd \\ D &\rightarrow dd \mid edd \end{aligned}$$

Scrivere una grammatica G' tale che G' non presenti prefissi comuni e $\mathcal{L}(G') = \mathcal{L}(G)$.

Soluzione

$$\begin{aligned} S &\rightarrow ASB \mid C \\ A &\rightarrow abA' \\ A' &\rightarrow Ac \mid \epsilon \\ B &\rightarrow bB' \\ B' &\rightarrow B \mid \epsilon \\ C &\rightarrow cDC' \\ C &\rightarrow Ac \mid d \\ D &\rightarrow dd \mid edd \end{aligned}$$

Esercizio 8 Data la seguente grammatica G :

$$\begin{aligned} S &\rightarrow ASB \mid SBC \mid SC \mid D \\ A &\rightarrow Aac \mid Aa \mid aa \\ B &\rightarrow Bb \mid Bbc \mid b \\ C &\rightarrow cD \mid cDd \\ D &\rightarrow dd \mid de \mid def \end{aligned}$$

Scrivere una grammatica G' tale che: (i) G' non presenti né ricorsione sinistra diretta né prefissi comuni; (ii) $\mathcal{L}(G') = \mathcal{L}(G)$.

Esercizio 9 Si consideri il frammento del linguaggio Java costituito dalle stringhe che corrispondono alla definizione di un singolo metodo `void` che accetta solo parametri di tipo `int` e, nelle istruzioni, solo dichiarazioni di variabili di tipo `int` e assegnazioni di variabili con espressioni uguali a singoli valori interi o a singole variabili.

Un esempio di stringa appartenente a questo linguaggio è il seguente:

```
void metodo1 (int x) {
    int y;
    y=1;
    x=y;
}
```

Altro esempio:

```
void metodo2 (int x, int p) {
    int y;
    int z;
    z = p;
    y = x;
    z = 12;
}
```

(1) Scrivere una grammatica non contestuale per tale linguaggio, dividendo la specifica del lessico del linguaggio (che va definita mediante espressioni regolari) dalla specifica della sintassi vera e propria; (2) scrivere una specifica JavaCC corrispondente alla specifica scritta al punto 1.

Soluzione Specifica del lessico:

```
IDENT = (a | ... | z | A | ... | Z) (a | ... | z | A | ... | Z | 0 | ... | 9)*
VALINT = (+|-)? (0 | ... | 9)+
TIPOVOID = void
TIPOINT = int
VIRG = ,
PVIRG = ;
TONDAAP = (
TONDACH = )
GRAFAP =
SEPARATORE = ( |\n|\r|\t)+
```

N.B. la classe SEPARATORE è una classe di token da non restituire (non va considerata nella specifica della grammatica).

Specifica della grammatica non contestuale:

```
start → TIPOVOID IDENT TONDAAP listapar TONDACH GRAFAP seqistr GRAFCH
listapar → ε | dichpar restolistapar
dichpar → TIPOINT IDENT
restolistapar → ε | VIRG dichpar restolistapar
seqistr → ε | istruzione seqistr
istruzione → dichiarazione | assegnazione
dichiarazione → TIPOINT IDENT PVIRG
assegnazione → IDENT UGUALE latodestroassegnazione PVIRG
latodestroassegnazione → IDENT | VALINT
```

Specifica JavaCC:

SKIP:

```
{ (" " | "\n" | "\r" | "\t")+ }
```

TOKEN:

```
{
  <IDENT: ("a" | ... | "z" | "A" | ... | "Z") ("a" | ... | "z" | "A" | ... | "Z" | "0" | ... | "9")*>
  | <VALINT: ("+" | "-")? ("0" | ... | "9")+>
  | <TIPOVOID: "void">
  | <TIPOINT: "int">
  | <VIRG: ", ">
  | <PVIRG: "; ">
  | <TONDAAP: "(">
  | <TONDACH: ")">
  | <GRAFAP: "{">
  | <GRAFCH: "}">
}
```

```
void start() : {}
{ <TIPOVOID> <IDENT> <TONDAAP> listapar() <TONDACH> <GRAFAP> seqistr() <GRAFCH> }
```

```
void listapar() : {}
{ dichpar() restolistapar() | {} }
```

```
void dichpar() : {}
{ <TIPOINT> <IDENT> }
```

```
void restolistapar() : {}
{ <VIRG> dichpar() restolistapar() | {} }
```

```
void seqistr() : {}
{ istruzione() seqistr() | {} }
```

```

void istruzione() : {}
{ dichiarazione() | assegnazione() }

void dichiarazione() : {}
{ <TIPOINT> <IDENT> <PVIRG> }

void assegnazione() : {}
{ <IDENT> <UGUALE> latodestroassegnazione() <PVIRG> }

void latodestroassegnazione() : {}
{ <IDENT> | <VALINT> }

```

Esercizio 10 Estendere il linguaggio dell'esercizio precedente come segue:

1. estendere il lessico ammettendo la parola chiave **String** e le costanti di tipo **String**, cioè sequenze (anche vuote) di caratteri tra doppi apici;
2. estendere la sintassi ammettendo anche il tipo **String** nelle dichiarazioni di parametri e di variabile, nonché le costanti di tipo **String** nel lato destro delle istruzioni di assegnazione.

Un esempio di stringa appartenente a questo linguaggio è il seguente:

```

void metodo1 (int x, String s) {
    int y;
    String p;
    y=1;
    p="ciao";
}

```

Altro esempio:

```

void metodo2 (int x, String s, String p) {
    String y;
    String z;
    z = p;
    y = "ciao ciao";
    z = 324;
}

```

Esercizio 11 Estendere il linguaggio dell'esercizio precedente come segue:

1. estendere il lessico ammettendo le parole chiave **if** e **else** e il simbolo **==**;
2. estendere la sintassi ammettendo anche l'istruzione **if-else** (cioè la presenza del ramo **else** è obbligatoria) con condizione dell'istruzione del tipo *variabile == variabile*.

Un esempio di stringa appartenente a questo linguaggio è il seguente:

```

void metodo1 (int x, String s) {
    int y;
    String p;
    y = 0;
    if (x==y)
        y=1;
    else p="ciao";
}

```

Altro esempio:

```

void metodo2 (int x, int z, String s) {
    int y;
    String p;
    y = 0;
    if (x==y) {
        y=1;
        if (z==x)
            p="ciao";
        else p = "ciao ciao";
    }
    else {
        p="ciao ciao ciao";
        y=123;
    }
}

```

Esercizio 12 Estendere il linguaggio dell'esercizio precedente ammettendo espressioni con l'operatore + nei lati destri delle assegnazioni, e la presenza dell'operatore booleano && nelle condizioni dell'istruzione if.

Un esempio di stringa appartenente a questo linguaggio è il seguente:

```

void metodo1 (int x, String s) {
    int y;
    int z;
    String p;
    z = x;
    y = x + 10;
    if (x==y && z==1)
        y=1;
    else p="ciao";
}

```

Altro esempio:

```

void metodo2 (int x, int z, String s) {
    int y;
    String p;

```

```

y = x + 10;
if (x==y && z==1) {
    y = z + 1 + x;
    if (z==x)
        p="ciao";
    else p = "ciao ciao";
}
else {
    p="ciao ciao ciao";
    y=1+x+123;
}
}

```

Esercizio 13 Estendere il linguaggio dell'esercizio precedente ammettendo la definizione di più metodi nella stessa stringa.

Un esempio di stringa appartenente a questo linguaggio è il seguente:

```

void metodo1 (int x, String s) {
    int y;
    int z;
    String p;
    z = x;
    y = x + 10;
    if (x==y && z==1)
        y=1;
    else p="ciao";
}
void metodo2 (int x, int z, String s) {
    int y;
    String p;
    y = x + 10;
    if (x==y && z==1) {
        y = z + 1 + x;
        if (z==x)
            p="ciao";
        else p = "ciao ciao";
    }
    else {
        p="ciao ciao ciao";
        y=1+x+123;
    }
}
}

```