

Linguaggi Formali e Traduttori

4.2 Parsing ricorsivo discendente

- Sommario
- Struttura del parser ricorsivo
- Algoritmo di parsing ricorsivo
- Implementazione Java del parser (classe base)
- Esempio: parser per il linguaggio $a^n b^n$
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Problema

- Realizzazione pratica di un parser top-down.

In questa lezione

- Studiamo una tecnica basata sulla **ricorsione** per la realizzazione pratica di un parser top-down.

Struttura del parser ricorsivo

Idea

Usare la pila del linguaggio di programmazione per “ricordare” il suffisso della forma sentenziale sinistra da riconoscere.

Elementi chiave

- Il parser ha una procedura per ogni variabile della grammatica.
- La procedura A nel parser riconosce le stringhe generate da A nella grammatica.
- La procedura A usa il simbolo corrente e gli insiemi guida, per scegliere la produzione $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ da usare per riscrivere A .
- Per ogni simbolo X trovato nel corpo della produzione scelta:
 - Se X è un simbolo terminale, il metodo controlla che il simbolo corrente sia proprio X . In tal caso, fa avanzare il lexer al simbolo successivo. In caso contrario, il metodo segnala un errore di sintassi.
 - Se X è una variabile, il metodo invoca la procedura X .

Algoritmo di parsing ricorsivo

```
var  $w$  : string  
var  $i$  : int
```

// w è la stringa da riconoscere con \$ in fondo
// i è l'indice del prossimo simbolo di w da leggere

```
procedure match( $a$  : symbol)  
  if  $w[i] = a$  then  $i \leftarrow i + 1$  else error
```

```
procedure parse( $v$  : string)  
   $w \leftarrow v\$$   
   $i \leftarrow 0$   
   $S()$   
  match( $\$$ )
```

// v è la stringa da riconoscere

// S è il simbolo iniziale della grammatica
// controlla di aver letto tutta la stringa

```
procedure  $A()$   
  if  $w[i] \in \text{GUIDA}(A \rightarrow \alpha_1)$  then  
    :  
  else if  $w[i] \in \text{GUIDA}(A \rightarrow \alpha_k)$  then  
    for  $X \in \alpha_k$  do  
      if  $X$  è un terminale then match( $X$ ) else  $X()$   
    :  
  else error
```

// $A \rightarrow \alpha_1 \mid \dots \mid \alpha_n$ sono le produzioni per A

// $w[i]$ non è nell'insieme guida di nessuna produzione per A

Implementazione Java del parser (classe base)

```
public abstract class Parser {  
    private String w;           // stringa da riconoscere  
    private int i;              // indice del prossimo simbolo  
  
    protected char peek()      // legge il simbolo corrente  
    { return w.charAt(i); }  
  
    protected void match(char a) // controlla il simbolo corrente  
    { if (peek() == a) i++; else throw error(); }  
  
    public void parse(String v) { // avvia il parsing di v  
        w = v + "$";  
        i = 0;  
        S();  
        match('$');  
    }  
  
    protected abstract void S(); // simbolo iniziale della grammatica  
  
    protected SyntaxError error() { ... } // emette errore e interrompe  
}
```

- Per semplicità assumiamo che i simboli siano caratteri.

Esempio: parser per il linguaggio $a^n b^n$

Grammatica

$$S \rightarrow aSb \mid \varepsilon$$

Insiemi guida

- $\text{GUIDA}(A \rightarrow aSb) = \{a\}$
- $\text{GUIDA}(A \rightarrow \varepsilon) = \{b, \$\}$

Codice del parser

```
public class AnBn extends Parser {  
    protected void S() {  
        switch (peek()) {  
  
            case 'a': // S → aSb  
                match('a');  
                S();  
                match('b');  
                break;  
  
            case 'b': // S → ε  
            case '$':  
                break;  
  
            default:  
                throw error();  
        }  
    }  
}
```

Esercizi

Implementazione di parser

Implementare il parser ricorsivo discendente per le seguenti grammatiche:

1. La grammatica delle stringhe della forma wcw^R dove $w \in \{0,1\}^*$:
 - $S \rightarrow c \mid 0S0 \mid 1S1$
2. La grammatica delle stringhe di parentesi quadre bilanciate:
 - $S \rightarrow \varepsilon \mid [S]S$
3. La grammatica delle stringhe della forma $a^n b^n c^m$:
 - $S \rightarrow XC$
 - $X \rightarrow \varepsilon \mid aXb$
 - $C \rightarrow \varepsilon \mid cC$
4. La grammatica delle espressioni aritmetiche in forma prefissa:
 - $E \rightarrow 0 \mid 1 \mid \dots \mid 9 \mid +EE \mid *EE$
5. La grammatica delle **espressioni aritmetiche** in forma infissa.