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

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 $A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$ da usare per riscrivere A.
- Per ogni simbolo X trovato nel corpo della produzione scelta:

  Match(X)

  - 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.

    V();

Algoritmo di parsing ricorsivo

```
var w : string                          // w è la stringa da riconoscere con $ in fondo
var i : int                             // i è l'indice del prossimo simbolo di w da leggere

procedure match(a : symbol)
  if w[i] = a then i ← i + 1 else error

procedure parse(v : string)             // v è la stringa da riconoscere
  w ← v$
  i ← 0
  S()                                   // S è il simbolo iniziale della grammatica
  match($)                              // controlla di aver letto tutta la stringa

procedure A()                           // A → α_1 | ··· | α_n sono le produzioni per A
  if w[i] ∈ GUIDA(A → α_1) then
  ⋮
  else if w[i] ∈ GUIDA(A → α_k) then
    for X ∈ α_k do
      if X è un terminale then match(X) else X()
  ⋮
  else error                            // 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

$$S \to aSb \mid \varepsilon$$

Insiemi guida

- $\textsc{guida}(A \to aSb) = \{a\}$
- $\textsc{guida}(A \to \varepsilon) = \{b, \$\}$

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