Corso: Fondamenti, Linguaggi e Traduttori 2

Paola Giannini

Esercizi e domande in preparazione all'esame

Domande: generale

- Consideriamo un linguaggio, L, **come si specificano**
  1. il lessico di L
  2. la sintassi di L,
- **Cosa** si fa durante
  1. l'Analisi Lessicale
  2. l'Analisi Sintattica,
  3. l'Analisi Semantica (controllo dei tipi)
- Quali sono gli input e gli output delle 3 analisi precedenti?

Domande: scanner

- Quali classi lessicali fanno in generale parte di un linguaggio di programmazione
- Cosa è un Token
- Fare esempi di Token
- Come (con quale classe di linguaggi) si specificano i token dei linguaggi
- Come si può implementare il riconoscimento lessicale

Domande: parsing

- Definizione di grammatica LL(1)
- Per quali ragioni una grammatica può non essere LL(1)?
- Come si può specificare un parser Top-Down

Domande: AST e Symbol Table

- Cosa è e a cosa serve definire un Abstract Syntax Tree per una grammatica di un linguaggio di programmazione.
- Quale è la differenza fra AST e parsing tree per una stringa di un linguaggio di programmazione.

- Cosa è la Symbol Table, quali informazioni contiene e a cosa serve.
- Come può essere implementata una Symbol Table .

Domande: analisi di tipo e visitor

- Cosa si fa con l'Analisi di Tipo.
- A cosa serve il Pattern Visitor. Descriverne la struttura.

Domande

- Consideriamo il seguente linguaggio di espressioni intere e floating point:

```
inum            [0-9]+
fnum            [0-9]+.[0-9]+

Expr -> Expr plus  Expr
      | Expr  times Expr
      | Val
Val  ->  inum
      |  fnum
```

  Assumiamo che le operazioni `plus` e `times` possano avvenire solo se i due operandi sono interi oppure floating point.

- Scrivere
  - una espressione corretta,
  - una con errori lessicali,
  - una senza errori lessicali ma con errori sintattici
  - ed una con solo errori semantici

Esercizi: (1)

- Data la grammatica:

1. $S \rightarrow A\ C\ \$$
2. $C \rightarrow c$
3. $C \rightarrow \epsilon$
4. $A \rightarrow A\ B\ C\ d\ c$
5. $A \rightarrow B\ Q$
6. $B \rightarrow b\ B$
7. $B \rightarrow \epsilon$
8. $Q \rightarrow q$
9. $Q \rightarrow \epsilon$

1. trovare i non-terminali e le produzioni che generano la stringa vuota
2. gli insiemi `First` delle parti destre delle produzioni e `Follow` dei non terminali.
3. gli insiemi `Predict` delle produzioni.

Esercizi: (2-3)

Dire se le seguenti grammatiche sono LL(1) oppure no

1. $S \rightarrow A\ B\ c\ \$$
2. $A \rightarrow a$
3. $A \rightarrow \epsilon$
4. $B \rightarrow b$
5. $B \rightarrow \epsilon$

1. $S \rightarrow A\ b\ \$$
2. $A \rightarrow a$
3. $A \rightarrow B$
4. $A \rightarrow \epsilon$
5. $B \rightarrow b$
6. $B \rightarrow \epsilon$

Esercizi: (4)

Dire come mai la seguente grammatica non è LL(1) e se possibile definirne una equivalente LL(1)

1. $DL \rightarrow DL$; D
2. $DL \rightarrow D$
3. $D \rightarrow T\ IdL$
4. $IdL \rightarrow IdL$, `id`
5. $IdL \rightarrow$ `id`
4. $T \rightarrow$ `int`
4. $T \rightarrow$ `bool`

il punto e virgola e la virgola a destra nelle produzioni e `id`, `int` e `bool` sono simboli terminali

Esercizi: (5)

Consideriamo la grammatica

```
Expr -> Expr plus  Term
      | Term
Term -> Term times Val
      | Val
Val  ->  inum
      |  fnum
```

- Definire una grammatica per lo stesso linguaggio che permetta il parsing top-down e definire i `Predict` delle produzioni.