Формални Езици и Езикови Процесори
ТУ, кат. КС, летен семестър 2012

Лекция 11

Тема:

Синтактичен Анализ
(метод Рекурсивно Спускане)

Съдържание:

- Формални методи.
- Низходяща /Top-down/ СА стратегия.
- Разширени КСГ.
- Метод на рекурсивното спускане.
- Демо програми.

Основи на СА

- Формални методи за СА.

  Основна задача: Да се изгради дървото на разбора /syntax (parse) tree/.

Налице са две алтернативни стратегии/подходи:

Top-Down Низходяща: от корена към короната/листата.

Bottom-Up Възходяща: от короната/листата към корена.

Низходящи СА методи

- Метод груба сила
- Метод на рекурсивното спускане
- LL(k) метод
  - s-граматики
  - q-граматики
  - LL(1) граматики

Възходящи СА методи

- Локализиране на основата чрез:
  - Bound Context BC(m,n) граматики
  - Отношения на просто предшествие
  - Отношения на операторно предшествие
  - LR(k) граматики (LR(1), SLR(1), LALR(1))

РекСпуск парсер

- РекСпуск парсер представлява низходящите СА методи/стратегии.
- Необходими условия за прилагане на този метод:

  a/ КСГ без лява рекурсия в правилата;

  b/ инструменталният ПЕ да допуска рекурсия.

Практически подход

- Опростената обосновка на РекСпуск се свежда до следните две правила:
  - На всеки терминален символ съответства разпознаващ оператор.
  - На всеки нетерминален символ X съответства обръщение към ПП /функция, метод/, която разпознава (приема или отхвърля) низове, породени от нетерминала X.

Теоретичен подход

- Формалната обосновка се базира на:
  - Концепцията Транслираща Функция;
  - Концепцията Разширена Контекстно Свободна Граматика КСГ /Extended Context Free Grammar/.

Теоретичен подход

Транслиращата функция има за аргументи елементи на граматиката:

- терминални символи,
- нетерминални символи,
- низове от терминали/нетерминали (alternation, concatenation, closure операции),
- продукции,
- граматика с всички правила.

Теоретичен подход

Връщаната от Транслиращата функция стойност представлява първичен текст на конвенционален ПЕ за парсер, който разпознава низове, съответстващи на аргумента на Транслиращата функция.

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - Терминални символи

    a е терминал, *a* принадлежи на Σ

F(a) ≡ if (ch==‘a’) getnewch();
else error();

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - НеТерминални символи

  X е нетерминал, X принадлежи на N

  F(X) ≡ X( );

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - Низове от терминали/нетерминали

  $\alpha 1 \mid \alpha 2$ (alternation operator)

```
F(α1|α2) ≡
  if (ch in FIRST(α1) { F(α1); }
  else   if (ch in FIRST(α2) { F(α2); }
         else error();
```

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - Низове от терминали/нетерминали

  $\alpha 1.\alpha 2$ (concatenation operator)

  $F(\alpha 1.\alpha 2) \equiv F(\alpha 1); F(\alpha 2);$

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - Низове от терминали/нетерминали

  α^* (closure operator)

$F(\alpha^*) \equiv$ while (ch in FIRST(α))

```
{
   F(α);  getnewch();
}
```

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - Низове от терминали/нетерминали

  $\alpha+$ (closure operators)

```
F(α+) ≡ F(α); getnewch();          |   do  {
         while (ch in FIRST(α))    |         F(α);
         {                         |         getnewch();
            F(α); getnewch();      |       }
          }                        |   while (ch in FIRST(α));
```

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:

  Продукции на граматика $X \rightarrow \alpha$

$F(X \rightarrow \alpha) \equiv$ void X(void)

{

F(α);

}

Транслираща Функция F(.)

- Аргументи са компоненти на граматика:
  - Граматика с всички правила Gr = r1; r2; ... rn

```
F(Gr) ≡   F(r1)
          F(r2)
          . . .
          F(rn)
          void main(void)
           {
               getnewch();  S( );
           }
```

Разширени КСГ

- РКСГ се въвеждат с цел избягване на рекурсия (лява или дясна) в КСГ:
  - Определение: РКСГ са КСГ, при които лявата/дясната рекурсия в продукциите се заменя с итерация.
  - Пример: Граматика КСГ на АИ:

Expr $\rightarrow$ Term | Expr + Term	Expr $\rightarrow$ Term { + Term }*
Term $\rightarrow$ Factor | Term * Factor	Term $\rightarrow$ Factor {*Factor }*
Factor $\rightarrow$ a | (Expr)	Factor $\rightarrow$ a | (Expr)

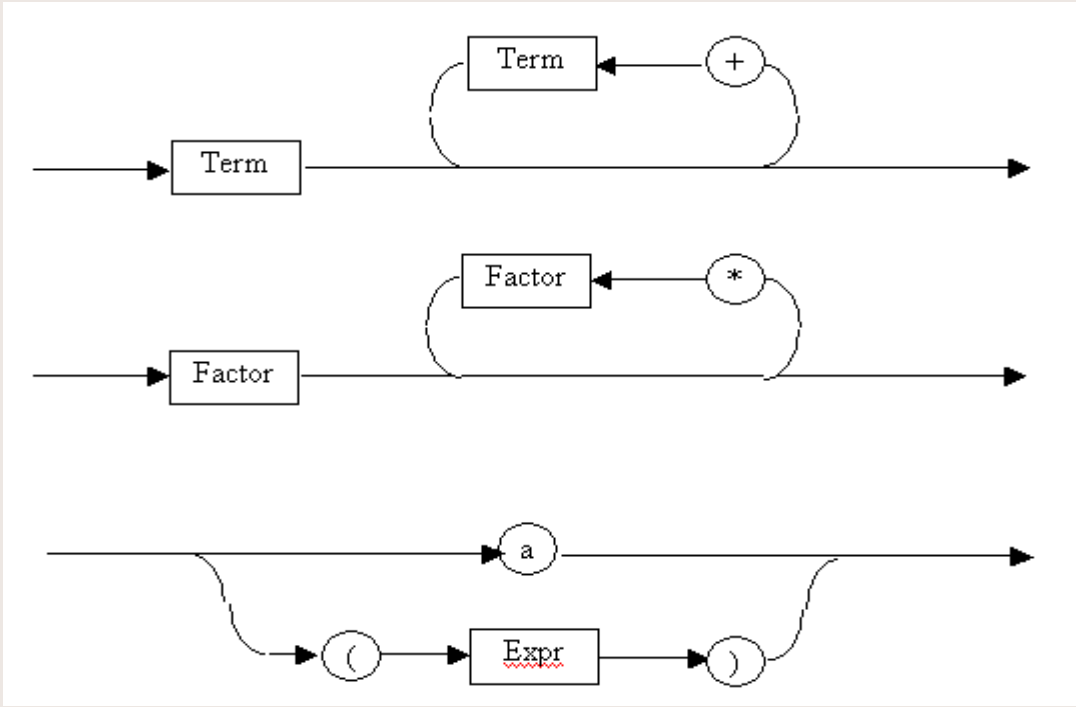
РКСГ синтактични диаграми

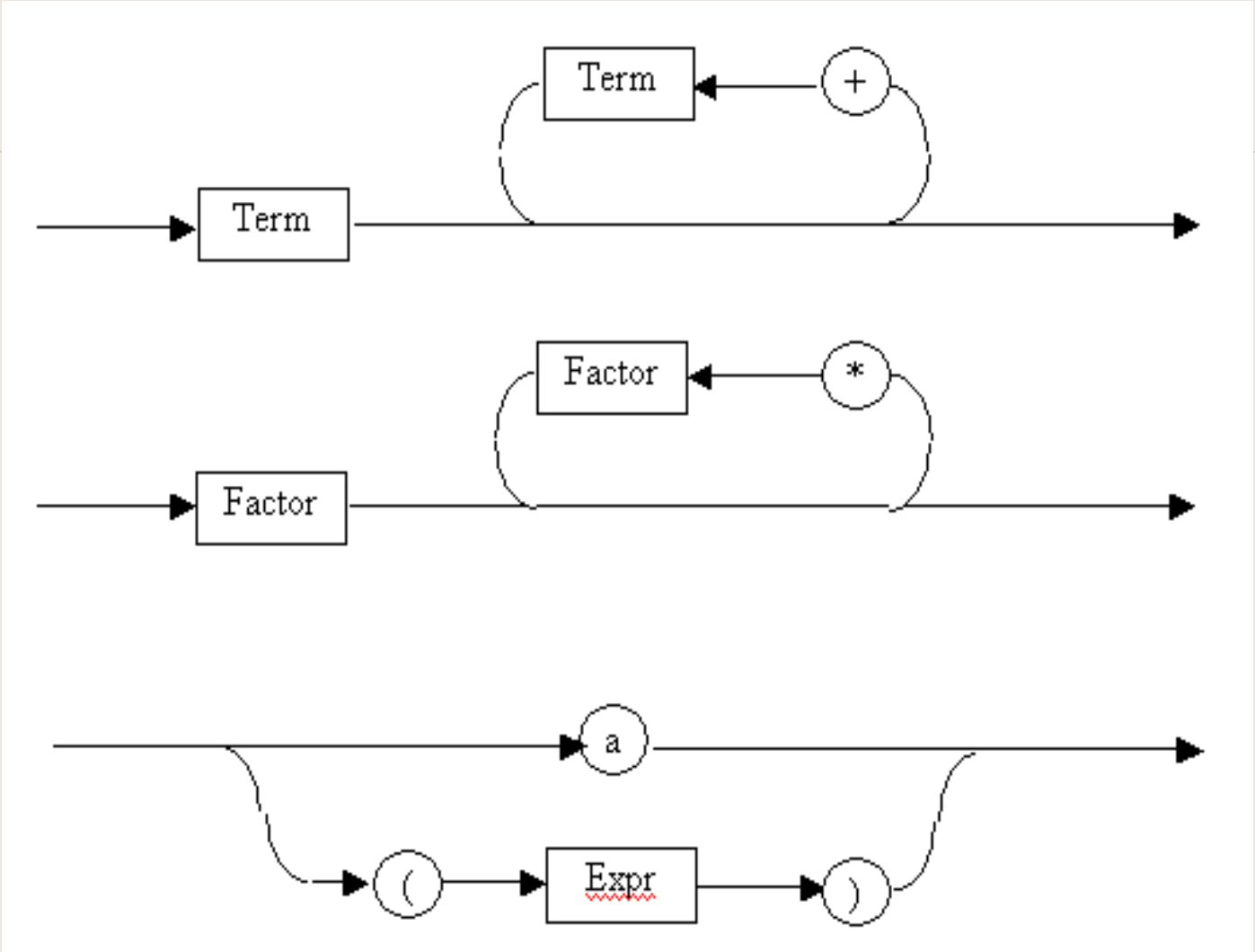
РКСГ се описват със синтактични диаграми.

Expr

Term

Factor




Term
+
Term
Factor
*
Factor
a
(
Expr
)

Реализация - РекСпуск парсер

КСГ: $A \rightarrow x \mid (B)$

$B \rightarrow A\ C$

$C \rightarrow + A\ C \mid \varepsilon$

Разширена КСГ: $A \rightarrow x \mid (B)$

$B \rightarrow A\ C$

$C \rightarrow \{ + A \}^*$

Реализация - РекСпуск парсер

Разширена КСГ: $A \rightarrow x \mid (B)$

$B \rightarrow A\ C$

$C \rightarrow \{ + A \}^*$

- За всеки нетерминал се строи синтактична диаграма.
- За всеки нетерминал /според синтактичната диаграма/ се съставя ПП функция, разпознаваща низовете, породени от нетерминала.
- Скелетът на парсера – виж сл. слайд

Реализация - РекСпуск парсер

Разширена КСГ: $A \rightarrow x \mid (B)$

$B \rightarrow A\ C$

$C \rightarrow \{ + A \}^*$

Скелет на парсера:

```
void a() { … }
void b() { … }
void c() { … }
void main()
{
  getnewch(); a();
}
```

Реализация - РекСпуск парсер

Разширена КСГ: $A \rightarrow x \mid (B)$

$B \rightarrow A\ C$

$C \rightarrow \{ + A \}^*$

Три синтактични диаграми приведени към една.

Скелет на парсера:

```
void a() { … }
void main()
{
  getnewch(); a();
}
```

Реализация - РекСпуск парсер

опростена КСГ на АИ:

- Expr → Term | Expr + Term
- Term → Factor | Term * Factor
- Factor → a | (Expr)

Expr → Term { + Term }*
Term → Factor { * Factor }*
Factor → a | (Expr)

Три синтактични диаграми.

Скелет/структура на парсера:

```
void expr(); void term(); void factor();
void main() { getnewch(); expr(); }
void expr() {…}
void term() {…}
void factor() {…}
```

Обработка на грешки

Откриване на грешките: специални маркери /позиции/ в синтактичните диаграми.

ОПЗ и парсери по метод на рекурсивното спускане

- Семантично натоварен парсер

Демо програми:

- recda1.cpp, recda2.cpp,
- recdex1.cpp, recdex1m.cpp, recdex5.cpp,
- recdConstList1.cpp, recdConstList2.cpp,
- recdIdentList1.cpp, recdIdentList2.cpp.

Благодаря
За
Вниманието