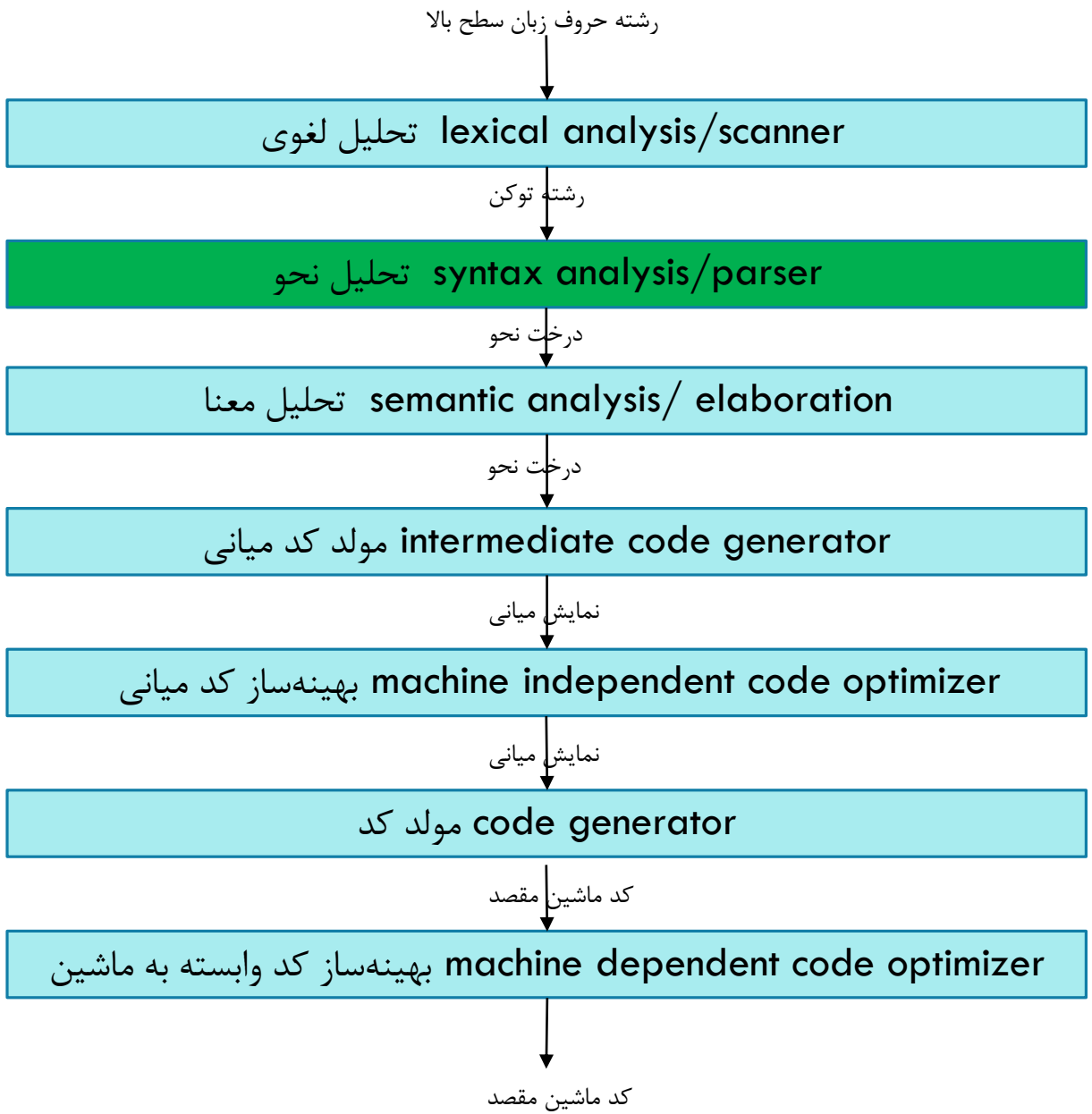
کامپایلر
تجزیه گر
تجزیه بالا به پائین

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان


رشته حروف زبان سطح بالا
lexical analysis/scanner تحلیل لغوی
رشته توکن
syntax analysis/parser تحلیل نحو
درخت نحو
semantic analysis/ elaboration تحلیل معنا
درخت نحو
intermediate code generator مولد کد میانی
نمایش میانی
machine independent code optimizer بهینه‌ساز کد میانی
نمایش میانی
code generator مولد کد
کد ماشین مقصد
machine dependent code optimizer بهینه‌ساز کد وابسته به ماشین
کد ماشین مقصد

مقدمه

مرحله دوم در پیشا

کار با نتیجه حاصل از پویشگر

هر واژه با دسته نحوی

اشتقاق ساختار نحوی برنامه

اهمیت سرعت

نیاز

- سازوکار صوری جهت مشخص‌سازی جمله معتبر
- روش سیستماتیک معین‌کردن عضویت در زبان

تجزیهٔ نزول-بازگشتی

امکان تجزیهٔ راحت بعضی از دستورها با استفاده از الگوریتمی با نام «نزول-بازگشتی»

- چرا نزول و نه کاهش

تبدیل هر تولید دستور به یک «بند» از تابعی بازگشتی

تجزیهٔ نزول بازگشتی– *مثال ۱*

$S \rightarrow \text{if } E \text{ then } S \text{ else } S$

$S \rightarrow \text{begin } S\ L$

$S \rightarrow \text{print } E$

$L \rightarrow \text{end}$

$L \rightarrow ;\ S\ L$

$E \rightarrow \text{num} = \text{num}$

مثال-

تجزیه‌گر نزول-بازگشتی جهت دستور روبرو

- «یک» تابع به ازای هر متغیر
- «یک» بند به ازای هر قانون تولید

نیاز به تعریف مناسب توابع خطا و گرفتن تکه

تجزیهٔ نزول بازگشتی- مثال ۱- *ادامه*

S → if E then S else S

S → begin S L

S → print E

L → end

L → ; S L

E → num = num

```
enum token {IF, THEN, ELSE, BEGIN, END, PRINT, SEMI, NUM, EQ};
extern enum token getToken(void);

enum token tok;
void advance() {tok=getToken();}
void eat(enum token t) {if (tok==t) advance(); else error();}

void S(void) {switch(tok) {
        case IF:    eat(IF); E(); eat(THEN); S();
                                  eat(ELSE); S(); break;
        case BEGIN: eat(BEGIN); S(); L(); break;
        case PRINT: eat(PRINT); E(); break;
        default:    error();
       }}
void L(void) {switch(tok) {
        case END:   eat(END); break;
        case SEMI:  eat(SEMI); S(); L(); break;
        default:    error();
       }}
void E(void) {  eat(NUM); eat(EQ); eat(NUM); }
```

تجزیه‌گرهای بالابه‌پائین نزول بازگشتی

مثال- $A \rightarrow \beta_1 \mid \beta_2 \mid \beta_3$

- دستور پیش‌بین
- پس دانستن آغاز، امکان یافتن سمت راست مناسب
-

```
/* یافتن A */
اگر (تکه‌فعلی ∈ آغاز($\beta_1$))
    یافتن (بسط) $\beta_1$ و بازگشت درست
وگرنه (تکه‌فعلی ∈ آغاز($\beta_2$))
    یافتن (بسط) $\beta_2$ و بازگشت درست
وگرنه (تکه‌فعلی ∈ آغاز($\beta_3$))
    یافتن (بسط) $\beta_3$ و بازگشت درست
ورنه {
    گزارش خطا بر اساس A و تکه‌فعلی
}
```

تجزیهٔ نزول بازگشتی- مثال ۲

$S \rightarrow E\,\$$

$E \rightarrow E + T$

$E \rightarrow E - T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow T / F$

$T \rightarrow F$

$F \rightarrow id$

$F \rightarrow num$

$F \rightarrow (E)$

```
void S(void) {  E(); eat(EOF); }
void E(void) {switch (tok) {
          case ?: E(); eat(PLUS); T(); break;
          case ?: E(); eat(MINUS); T(); break;
          case ?: T(); break;
          default: error();
         }}
void T(void) {switch (tok) {
          case ?: T(); eat(TIMES); F(); break;
          case ?: T(); eat(DIV); F(); break;
          case ?: F(); break;
          default: error();
         }}
```

مناقشه

- تابع E راهی برای اینکه کدام بند را بکار گیرد ندارد

تجزیهٔ نزول بازگشتی- مثال ۲ -*ادامه*

$S \rightarrow E\ \$$

$E \rightarrow E + T$

$E \rightarrow E - T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow T / F$

$T \rightarrow F$

$F \rightarrow id$

$F \rightarrow num$

$F \rightarrow (E)$

```
void S(void) {  E(); eat(EOF); }
void E(void) {switch (tok) {
        case ?: E(); eat(PLUS); T(); break;
        case ?: E(); eat(MINUS); T(); break;
        case ?: T(); break;
        default: error();
}}
void T(void) {switch (tok) {
        case ?: T(); eat(TIMES); F(); break;
        case ?: T(); eat(DIV); F(); break;
        case ?: F(); break;
        default: error();
}}
```

دو عبارت 4+(1*2-3) و (1*2-3)

- در عبارت نخست، باید استفاده از $E \rightarrow E + T$
- در مورد اخیر، استفاده از قاعدهٔ $E \rightarrow T$

تجزیهٔ نزول-بازگشتی (پیش‌بین)

- صرفا اجراپذیر بر دسته خاصی از دستورها
- *علامت حرف نخست* هر زیر عبارت دارای اطلاع کافی
- جهت انتخاب قاعده

- جهت فهم بهتر مسئله و حل آن
  - ابتداء نیاز به تعریف مفهوم مجموعه‌های «آغاز»
  - سپس ایجاد تجزیه‌گر نزول-بازگشتی بی‌مناقشه

امکان پیاده‌سازی ابزار تجزیه‌ساز

- گاهی اوقات زیادی و یا ناممکن
- امکان پیاده‌سازی تجزیه‌گرهای با دست با استفاده از تجزیهٔ پیش‌بین

مجموعه‌های آغاز و مجموعه پیرو

$S \rightarrow E\ \$$

$E \rightarrow E + T$

$E \rightarrow E - T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow T / F$

$T \rightarrow F$

$F \rightarrow id$

$F \rightarrow num$

$F \rightarrow (E)$

γ رشته‌ای تشکیل‌شده از متغیرها و حروف الفباء

آغاز رشته

- فهرست تمامی حروف الفبا که در آغاز هر رشتهٔ مشتق شده از γ
- مثال- $FIRST(T * F)$

.

مجموعه‌های آغاز و مجموعه پیرو

$S \rightarrow E\ \$$

$E \rightarrow E + T$

$E \rightarrow E - T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow T / F$

$T \rightarrow F$

$F \rightarrow id$

$F \rightarrow num$

$F \rightarrow (E)$

γ رشته‌ای تشکیل‌شده از متغیرها و حروف الفباء

آغاز رشته

- فهرست تمامی حروف الفبا که در آغاز هر رشتهٔ مشتق شده از γ
- مثال- $FIRST(T * F) = \{id, num, (\}$

پیرو

- حرف الفبائی که به دنبال متغیری در جریان اشتقاق خواهد آمد.

مجموعه‌های آغاز

مثال- دستوری شامل دو قاعدهٔ تولید $X \rightarrow \gamma_1$ و $X \rightarrow \gamma_2$

- هر دو دارای متغیر یکسان در سمت چپ قاعده
- سمت راست هر دو قاعده دارای «آغازهای» **مشترک**
  - عدم‌امکان تجزیهٔ دستور مذکور با استفاده از تجزیهٔ پیش‌بین
  - حرف I هم در آغاز γ_1 و هم در آغاز γ_2
- تابع X در تجزیهٔ نزول-کاهشی
  - با دیدن I نمی‌داند چه عملی را انجام دهد.

روش ساده‌تر

- کامپایلر پیش از اجرا بداند که اولین نویسه رشتهٔ حاصل از اعمال قانون تولید چیست.
- مقایسه با نویسه یا تکهٔ فعلی
- انتخاب خردورزانهٔ قانون تولیدی بعدی

مجموعه آغاز - مثالی جهت نمایش چرائی استفاده

مثال- رشته ورودی cad جهت دستور روبرو

- پس از شروع و خواندن نویسه c، نویسهٔ بعدی a
  - چشم‌پوشی از $A \rightarrow bc$
  - چون اولین نویسه این قانون b است و نه a
  - استفاده مستقیم از قانون $A \rightarrow a$
  - چون اولین نویسهٔ این قانون a است

$$S \rightarrow cAd$$
$$A \rightarrow bc \mid a$$

در صورت اطلاع تجزیه‌گر از اولین نویسه قابل حصول از هر قانون

- منجر به اعمال خردمندانه قانون درست جهت دریافت درخت نحو صحیح

محاسبهٔ مجموعه آغاز

First(X)

محاسبه آسان

- اگر γ =XYZ
- به نظر می‌توان Y و Z را نادیده گرفت و صرفا توجه به X کافی است
  - به مثال روبرو حساب آغاز Z
  - ولی افتاد مشکل‌ها!
- نیاز به بررسی همه متغیرهائی تولیدکنندهٔ رشتهٔ تهی
  - موسوم به رشته‌های پوچ‌پذیر
  - بررسی چه چیزی در پی متغیر پوچ‌پذیر می‌آید (پیرو متغیر پوچ‌پذیر)

$$Z \rightarrow d$$
$$Z \rightarrow X\ Y\ Z$$
$$Y \rightarrow \epsilon$$
$$Y \rightarrow c$$
$$X \rightarrow Y$$
$$X \rightarrow a$$

محاسبهٔ مجموعه آغاز- *ادامه*

با داشتن رشته‌ای شامل متغیرها و حروف

- پوچ‌پذیری متغیری برقرار است اگر متغیر رشته تهی تولید کند
- آغاز رشته مجموعه حروفی است که در ابتدای رشته‌های حروف مشتق از ان بدست می‌ایند
- پیرو متغیری مجموعه حروفی است که بلافاصله در پی متغیر مذکور می‌آیند.
  - $t \in FOLLOW(X)$
    - اگر اشتقاقی به صورت Xt باشد
    - می‌تواند به صورت $XYZt$ *باشد اگر* Y و Z هر دو منتج به تهی شوند

محاسبهٔ مجموعه آغاز- الگوریتم

First(X)

چگونگی محاسبه مجموعه آغاز

۱- اگر x حرف الفباء، آن‌گاه $First(x) = \{ 'x' \}$

۲- اگر $x \to \epsilon$ جزو قواعد تولید، آن‌گاه افزودن ϵ به مجموعه آغاز

۳- اگر $X \to Y_1\ Y_2\ Y_3 \dots Y_n$ جزو قواعد تولید،

- الف- $First(X) = First(Y_1)$
- ب- اگر $\epsilon \in First(Y_1)$، آن‌گاه $First(X) = \{ First(Y_1) - \epsilon \}\ U\ \{ First(Y_2) \}$
- ج- اگر به ازای ۱ تا i $\epsilon \in First(Y_i)$، آن‌گاه $First(Y_{i+1}) \in First(X)$
- د- اگر به ازای تمامی Y_iها و $i = 1:n$ داشته باشیم $\epsilon \in First(Yi)$، آن‌گاه $\epsilon \in First(X)$

یافتن مجموعه آغاز – مثال

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid id$

مجموعه‌های آغاز

$First(E) = First(T) = \{(, id\}$
$First(E') = \{+, \epsilon\}$
$First(T) = First(F) = \{(, id\}$
$First(T') = \{*, \epsilon\}$
$First(F) = \{(, id\}$

$S \rightarrow ACB \mid Cbb \mid Ba$
$A \rightarrow da \mid BC$
$B \rightarrow g \mid \epsilon$
$C \rightarrow h \mid \epsilon$

مجموعه‌های آغاز

$First(S) = First(A) \cup First(B) \cup First(C) = \{d, g, h, \epsilon, b, a\}$
$First(A) = \{d\} \cup First(B) = \{d, g\} \cup First(C) = \{d, g, h, \epsilon\}$
$First(B) = \{g, \epsilon\}$
$First(C) = \{h, \epsilon\}$

مجموعه پیرو - مثالی جهت برهانی بر چرائی استفاده

مثال- رشته ورودی ab جهت دستور روبرو

$$S \rightarrow aAb$$
$$A \rightarrow c \mid \epsilon$$

- اعمال قانون $S \rightarrow aAb$ برای خواندن a
- نویسه ورودی بعدی b و متغیر بعدی جهت بسط A
  - A دارای قانون منجر به b نیست
  - دارای $A \rightarrow \epsilon$ و با اعمال آن رسیدن به رشته ورودی
  - ولی تجزیه‌گر زمانی می‌توانند چنین کاری انجام دهد که بداند پس از A، b رخ می‌دهد.
  - با اعمال یافتن پیرو

در صورت اطلاع تجزیه‌گر از حرف الفباء بعدی (پیرو)، امکان حذف متغیر

با یافتن مجموعه‌های آغاز و پیرو برای دستور، تجزیه‌گر امکان اعمال قوانین مناسب را خواهد یافت

یافتن مجموعه پیرو

Follow(X)

- مجموعه حروف الفباء که بدون فاصله در سمت راست متغیر X در قالب جمله‌ای ظاهر می‌شوند

چگونگی محاسبه مجموعه پیرو

۱- پیرو متغیر شروع $، $Follow(S) = \{\$\}$

۲- اگر $A \to mBn$، آن‌گاه $Follow(B) = First(n) - \{\epsilon\}$

۳- اگر $A \to mB$، آن‌گاه $Follow(B)$ دارای $Follow(A)$ است.

۴- اگر $A \to mBn$ و $\epsilon \in First(n)$، آن‌گاه

- $Follow(B)$ شامل $Follow(A) \cup \{First(n) - \epsilon\}$

یافتن مجموعه پیرو – مثال

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid id$

مجموعه‌های آغاز

$First(E) = First(T) = \{(, id\}$
$First(E') = \{+, \epsilon\}$
$First(T) = First(F) = \{(, id\}$
$First(T') = \{*, \epsilon\}$
$First(F) = \{(, id\}$

$Follow(E) = \{\$,)\}$
$Follow(E') = Follow(E) = \{\$,)\}$
$Follow(T) = \{First(E') - \epsilon\} \cup Follow(E') \cup Follow(E) = \{+, \$,)\}$
$Follow(T') = Follow(T) = \{+, \$,)\}$
$Follow(F) = \{First(T') - \epsilon\} \cup Follow(T') \cup Follow(T) = \{*, +, \$,)\}$

یافتن مجموعه پیرو – مثال

$S \rightarrow ACB|Cbb|Ba$
$A \rightarrow da|BC$
$B \rightarrow g|\epsilon$
$C \rightarrow h \mid \epsilon$

مجموعه‌های آغاز

$First(S) = First(A) \cup First(B) \cup First(C) = \{d, g, h, \epsilon, b, a\}$
$First(A) = \{d\} \cup First(B) = \{d, g\} \cup First(C) = \{d, g, h, \epsilon\}$
$First(B) = \{g, \epsilon\}$
$First(C) = \{h, \epsilon\}$

مجموعه‌های پیرو

$Follow(S) = \{\$\}$
$Follow(A) = \{First(C) - \epsilon\} \cup \{First(B) - \epsilon\} \cup Follow(S) = \{h, g, \$\}$
$Follow(B) = Follow(s) \cup \{a\} \cup \{First(c) - \epsilon\} \cup Follow(A) = \{\$, a, h, g\}$
$Follow(C) = \{First(B) - \epsilon\} \cup Follow(S) \cup \{b\} \cup Follow(A) = \{g, \$, b, h\}$

محاسبهٔ کلی مجموعه آغاز و پیرو و متغیرهای پوچ‌پذیر

مقداردهی اولیه تهی برای تمامی مجموعه‌های آغاز و پیرو و قراردادن پوچ‌پذیری برابر غلط تمامی متغیرها

آغاز هر پایانه برابر با خود پایانه

برای هر قاعده تولید $X \to Y_1\ Y_2\ Y_3 \dots Y_k$

- برای $i \to ۱$ تا k
  - اگر تمامی Y_i-ها پوچ‌پذیر، آن‌گاه X نیز پوچ‌پذیر
  - اگر تمامی $Y_1\ Y_2\ Y_3 \dots Y_{i-1}$ پوچ‌پذیر، آن‌گاه ***آغاز*** X $\to$ ***آغاز*** X $\cup$ ***آغاز*** Y_i
  - اگر تمامی $Y_{i+1} \dots Y_k$ همگی پوچ‌پذیر، آن‌گاه ***پیرو*** Y_i $\to$ ***پیرو*** Y_i $\cup$ ***پیرو*** X
  - برای $j \to i+1$ تا k
    - اگر تمامی $Y_{i+1} \dots Y_{j-1}$ همگی پوچ‌پذیر، آن‌گاه ***پیرو*** Y_i $\to$ ***پیرو*** Y_i $\cup$ ***آغاز*** Y_j

تکرار تا عدم تغییر مجموعه‌ها

[ببرسبز]

مثال محاسبهٔ کلی مجموعه آغاز و پیرو و متغیرهای پوچ‌پذیر

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$
$Y \rightarrow \epsilon$
$Y \rightarrow c$
$X \rightarrow Y$
$X \rightarrow a$

	تهی	آغاز	پیرو
X	بله	ac	acd
Y	بله	c	acd
Z	نه	acd	

برای هر قاعده تولید $X \rightarrow Y_1\ Y_2\ Y_3 \dots Y_k$

- برای $i \rightarrow ۱$ تا k
  - اگر تمامی Y_i-ها پوچ‌پذیر، آن‌گاه X نیز پوچ‌پذیر
  - اگر تمامی $Y_1\ Y_2\ Y_3 \dots Y_{i-1}$ پوچ‌پذیر، آن‌گاه ***آغاز*** X ← ***آغاز*** X ∪ ***آغاز*** Y_i
  - اگر تمامی $Y_{i+1} \dots Y_k$ همگی پوچ‌پذیر، آن‌گاه ***پیرو*** Y_i ← ***پیرو*** Y_i ∪ ***پیرو*** X
  - برای j $i+1 \rightarrow$ تا k
    - اگر تمامی $Y_{i+1} \dots Y_{j-1}$ همگی پوچ‌پذیر، آن‌گاه ***پیرو*** Y_i ← ***پیرو*** Y_i ∪ آغاز Y_j

مثال محاسبهٔ کلی مجموعه آغاز و پیرو و متغیرهای پوچ‌پذیر

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$
$Y \rightarrow \epsilon$
$Y \rightarrow c$
$X \rightarrow Y$
$X \rightarrow a$

	آغاز	پیرو
X	a,c ϵ	acd
Y	c, ϵ	acd
Z	acd	

چگونگی محاسبه مجموعه آغاز

۱- اگر x حرف الفباء، آن‌گاه $First(x) = \{ 'x' \}$

۲- اگر $x \rightarrow \epsilon$ جزو قواعد تولید، آن‌گاه افزودن ϵ به مجموعه آغاز

۳- اگر $X \rightarrow Y_1\ Y_2\ Y_3 \ldots Y_n$ جزو قواعد تولید،

- الف- $First(X) = First(Y_1)$
- ب- اگر $\epsilon \in First(Y_1)$، آن‌گاه $First(X) = \{ First(Y_1) - \epsilon \}\ U\ \{ First(Y_2) \}$
- ج- اگر به ازای ۱ تا i $\epsilon \in First(Y_i)$، آن‌گاه $First(Y_{i+1}) \in First(X)$
- د- اگر به ازای تمامی Y_iها و $i = 1{:}n$ داشته باشیم $\epsilon \in First(Yi)$، آن‌گاه $\epsilon \in First(X)$

چگونگی محاسبه مجموعه پیرو

۱- پیرو متغیر شروع $، $Follow(S) = \{ \$ \}$

۲- اگر $A \rightarrow mBn$، آن‌گاه $Follow(B) = First(n) - \{\epsilon\}$

۳- اگر $A \rightarrow mB$، آن‌گاه $Follow(B)$ دارای $Follow(A)$ است.

۴- اگر $A \rightarrow mBn$ و $\epsilon \in First(n)$، آن‌گاه

- $Follow(B)$ شامل $Follow(A) \cup \{First(n) - \epsilon\}$

منابع

[ببرسبز]

[اژدرها]

[کوپر]

[فیشر]