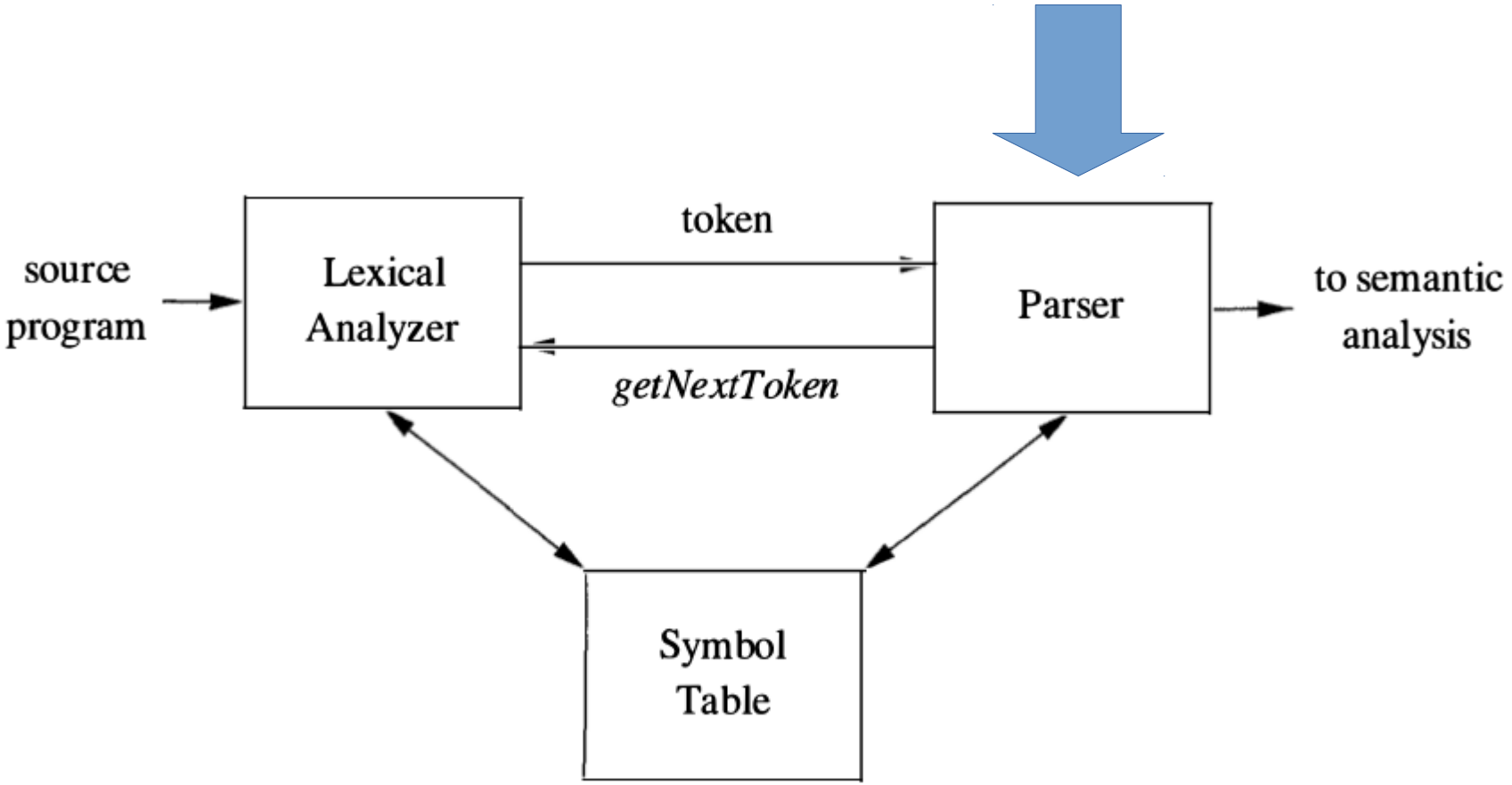
دانشگاه صنعتی شاهرود
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

درس اصول طراحی کامپایلر

مبحث Parsing (تجزیه)

مدرس:
علیرضا تجری

مرور مباحث گذشته



تعاریف اولیه

- استخراج / اشتقاق / بسط
- $E \rightarrow E + E \mid E * E \mid -E \mid (E) \mid id$
- غیرپایانه E فرم جمله‌ای E- را استخراج (مشتق) می‌کند / بسط می‌دهد
- $E \Rightarrow -E$
- نحوه استخراج (id)-
- $E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(id)$
- استخراج در صفر یا چند مرحله
- $E \overset{*}{\Rightarrow} -(id)$
- استخراج در یک یا چند مرحله
- $E \overset{+}{\Rightarrow} -(id)$

تعاریف اولیه

- برای هر رشته α داریم

- $\alpha \overset{*}{\Rightarrow} \alpha$
- if $\alpha \overset{*}{\Rightarrow} \beta$, $\beta \overset{*}{\Rightarrow} \gamma$ then $\alpha \overset{*}{\Rightarrow} \gamma$

- اگر $S \overset{*}{\Rightarrow} \alpha$ آنگاه α فرم‌جمله‌ای گرامر G است.
  - غیرپایانه S سمبل شروع گرامر G است.
- یک جمله در G: فرم‌جمله‌ای بدون غیرپایانه (ω)
- زبان تولید شده توسط گرامر ـ L(G): مجموعه جمله‌های آن گرامر

- $\omega \in L(G) \Leftrightarrow S \overset{*}{\Rightarrow} \omega$

- گرامرهای معادل: دو گرامری که می‌توانند یک زبان را تولید کنند

تعاریف اولیه

- استخراج (id+id)-

- $E \rightarrow E + E \mid E * E \mid -E \mid (E) \mid id$
- $E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(id+E) \Rightarrow -(id+id)$
- $E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(E+id) \Rightarrow -(id+id)$

- در هر مرحله، کدام غیرپایانه جایگزین شد؟
- بسط سمت‌چپ‌ترین leftmost derivation

$$\alpha \underset{lm}{\Rightarrow} \beta$$

  - هر بار سمت‌چپ‌ترین غیرپایانه بسط داده می‌شود

- بسط سمت‌راست‌ترین rightmost derivation

$$\alpha \underset{rm}{\Rightarrow} \beta$$

  - هر بار سمت‌راست‌ترین غیرپایانه بسط داده می‌شود

تعاریف اولیه

- نمونه بسط سمت‌چپ‌ترین (با اعمال قاعده $A \rightarrow \delta$)
- $\omega A \gamma \underset{lm}{\Rightarrow} \omega \delta \gamma$

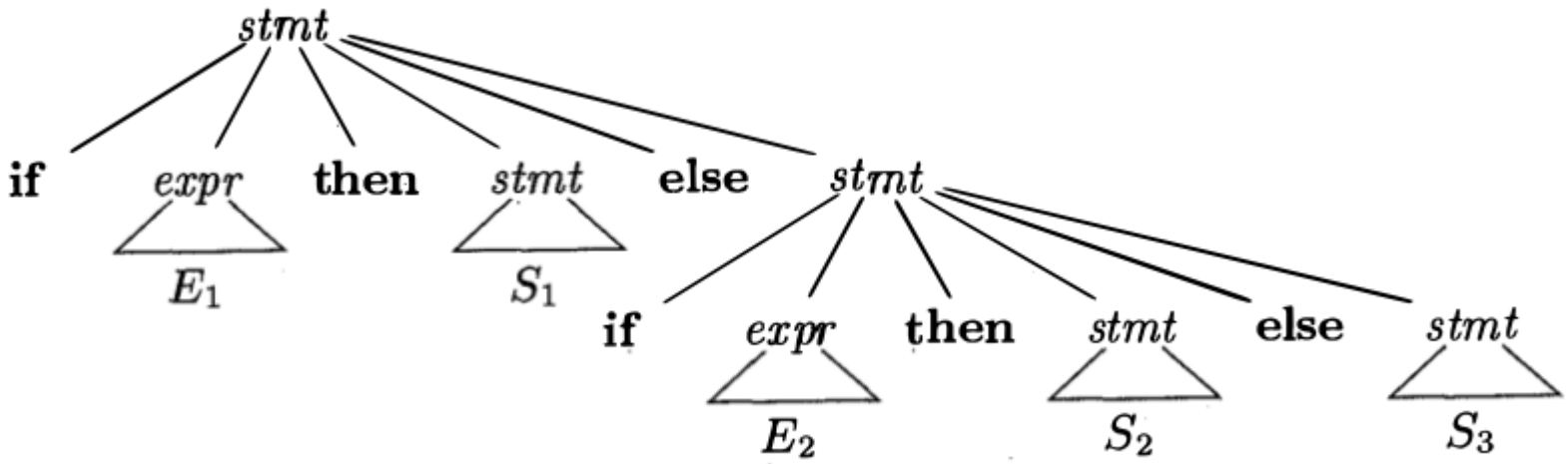
- با توجه به فرآیند استخراج یک جمله، می‌توان درخت تجزیه رشته را رسم کرد.

- ابهام و گرامر مبهم:
  - گرامری مبهم است که برای یک جمله، بیش از یک استخراج سمت‌چپ‌ترین و یا بیش از یک استخراج سمت‌راست‌ترین داشته باشد.
- $E \rightarrow E + E \mid E * E \mid -E \mid (E) \mid id$
- $id + id * id$

رفع ابهام else پادرهوا dangling else

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{other}
\end{array}
$$

if E_1 **then** S_1 **else if** E_2 **then** S_2 **else** S_3



رفع ابهام else پادرهوا dangling else

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{other}
\end{array}
$$

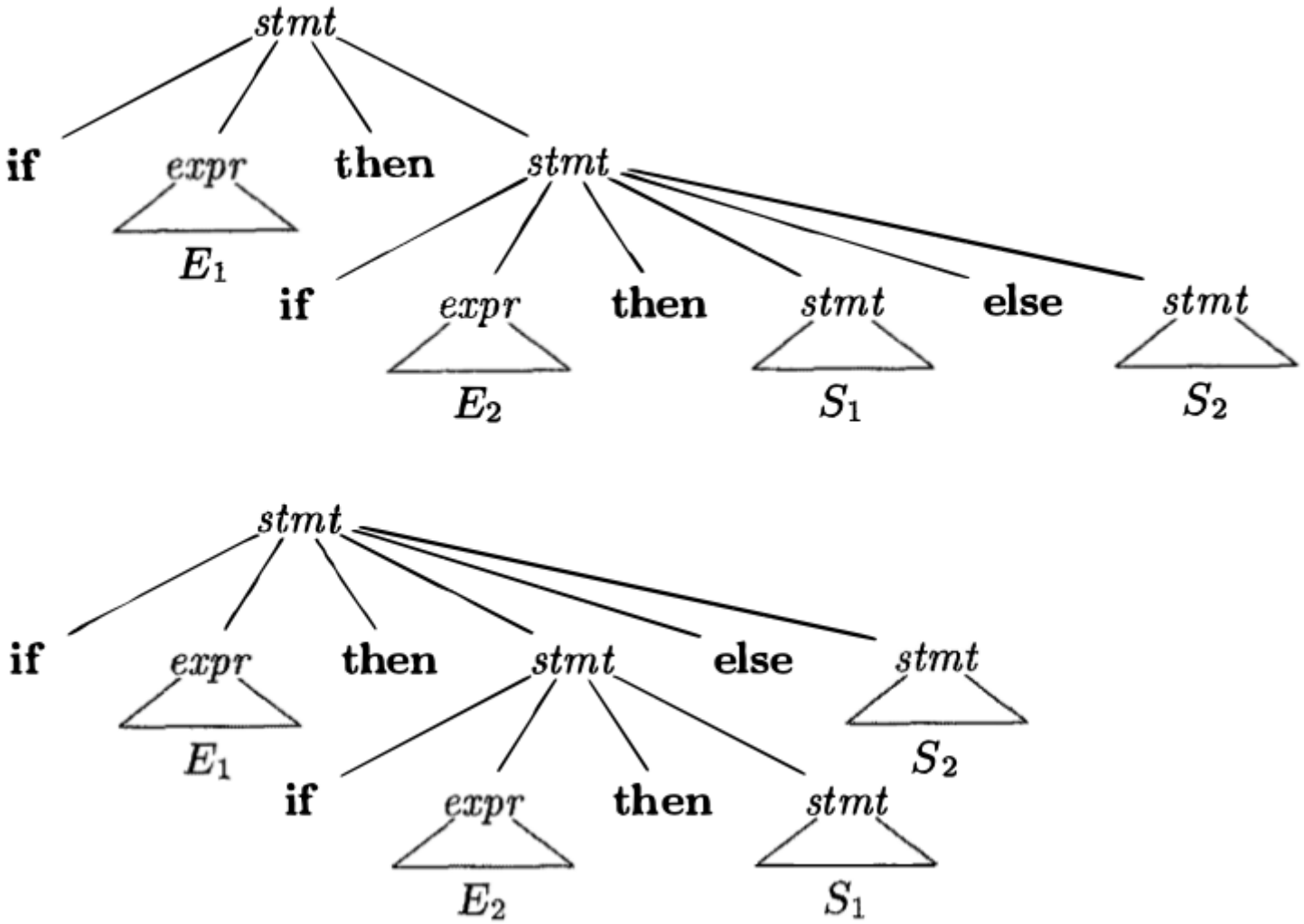
$$\textbf{if } E_1 \textbf{ then if } E_2 \textbf{ then } S_1 \textbf{ else } S_2$$

در این جمله، else برای کدام if است؟

```
if  E1 then
      if E2 then
            S1
      else
            S2
```

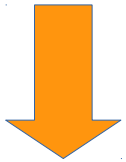
```
if  E1 then
      if E2 then
            S1
else
      S2
```

رفع ابهام else پادرهوا dangling else



رفع ابهام else پادرهوا dangling else

- قاعده کلی: هر else برای نزدیک‌ترین if بدون elseدر نظر گرفته شود.

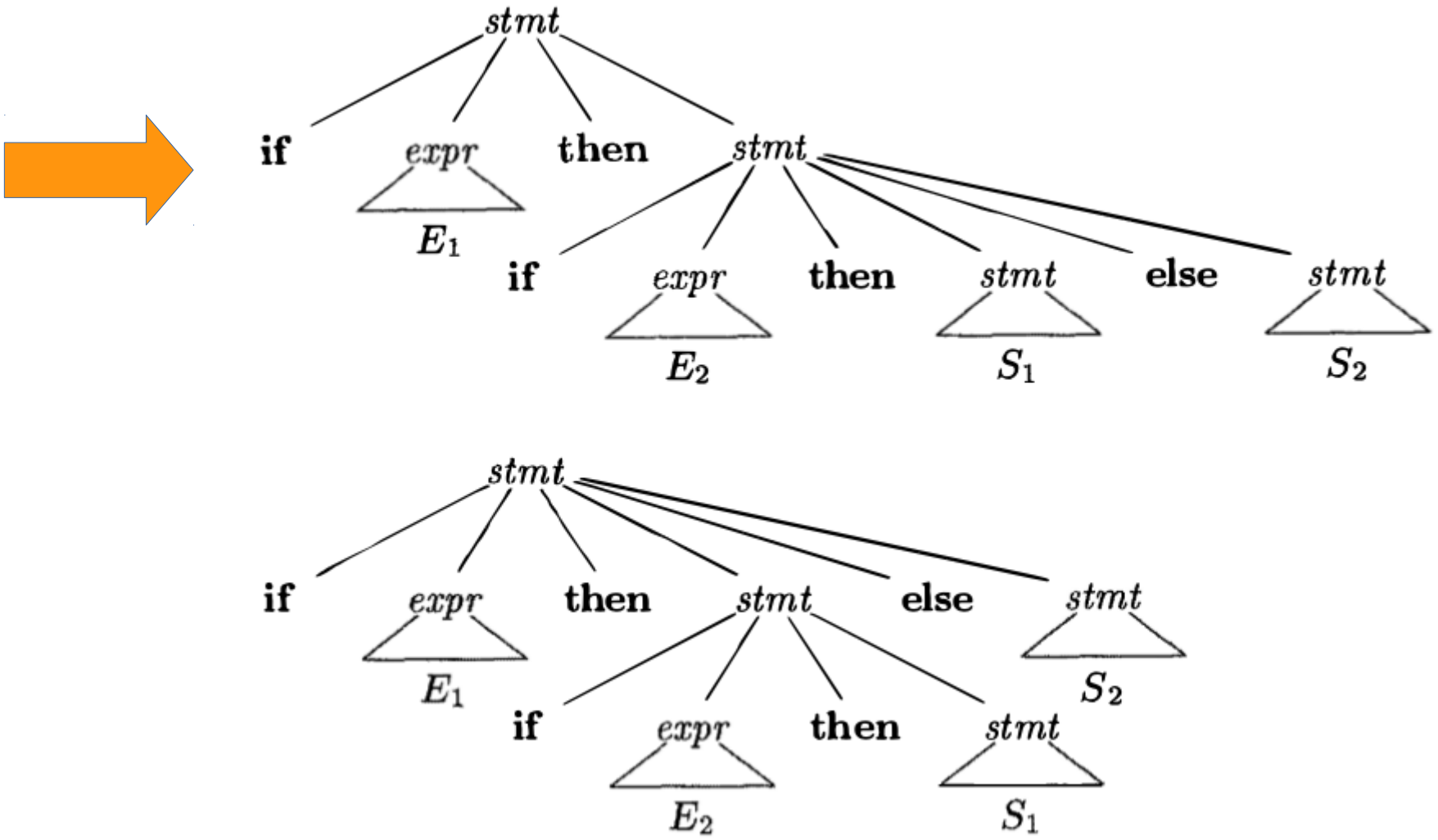


```
if  E1 then
      if E2 then
            S1
      else
            S2
```

```
if  E1 then
      if E2 then
            S1
else
      S2
```

رفع ابهام else پادرهوا dangling else

- قاعده کلی: هر else برای نزدیک‌ترین if بدون else در نظر گرفته شود.



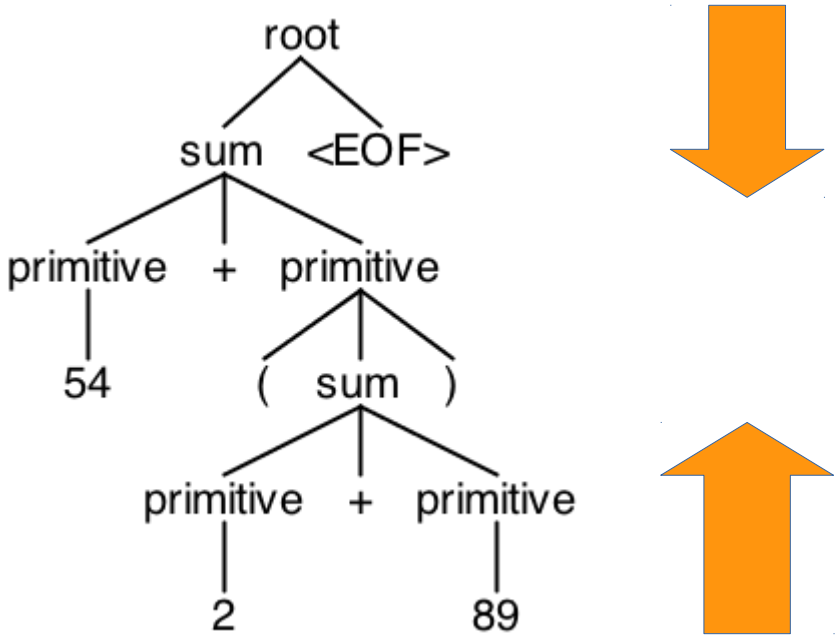
رفع ابهام else پادرهوا dangling else

- ایده: stmt بین then و else باید حتما دارای else باشد.

$$
\begin{array}{rcl}
stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{other}
\end{array}
$$

$$
\begin{array}{rcl}
stmt & \rightarrow & matched\_stmt \\
 & | & open\_stmt \\
matched\_stmt & \rightarrow & \textbf{if } expr \textbf{ then } matched\_stmt \textbf{ else } matched\_stmt \\
 & | & \textbf{other} \\
open\_stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } matched\_stmt \textbf{ else } open\_stmt
\end{array}
$$

دسته‌بندی کلی روش‌های تجزیه



- روش‌های تجزیه بالا به پایین
  - Top-Down
  - ساخت درخت تجزیه از ریشه به سمت برگ‌ها
- روش‌های تجزیه پایین به بالا
  - Bottom-Up
  - ساخت درخت تجزیه از برگ‌ها به سمت ریشه

در همه روش‌ها، ورودی از چپ به راست بررسی شده و با توجه به آن درخت تجزیه ساخته می‌شود.
هر بار یک توکن

دسته‌بندی کلی روش‌های تجزیه

- به طور کلی، پیچیدگی الگوریتم تجزیه یک رشته با استفاده از گرامرهای مستقل از متن:
  - $O(n^3)$
- برای زبان‌های برنامه‌نویسی:
  - معمولا $O(n)$
  - پارامتر n: تعداد توکن‌های رشته
  - یک‌بار پیمایش توکن‌ها از ابتدا تا انتها

روش‌های بالا به پایین را می‌توان به صورت دستی پیاده‌سازی کرد.
معمولا روش‌های پایین به بالا توسط ابزارهای تولید خودکار کامپایلر پیاده‌سازی می‌شوند.

فرآیند تجزیه در روش‌های بالا به پایین

1) گره ریشه را ایجاد کنید.

2) در گره N که با غیرپایانه A نشان داده شده است و فرزندی ندارد، یکی از قوانین مربوط به A را انتخاب کنید و فرزندان گره N را با توجه به آن قانون ایجاد کنید.

- انتخاب قانون با توجه به توکن ورودی کنونی (پیش‌بینی lookahead)

3) گره بعدی که باید گسترش یابد را پیدا کنید.

- اولین گره گسترش نیافته سمت چپ

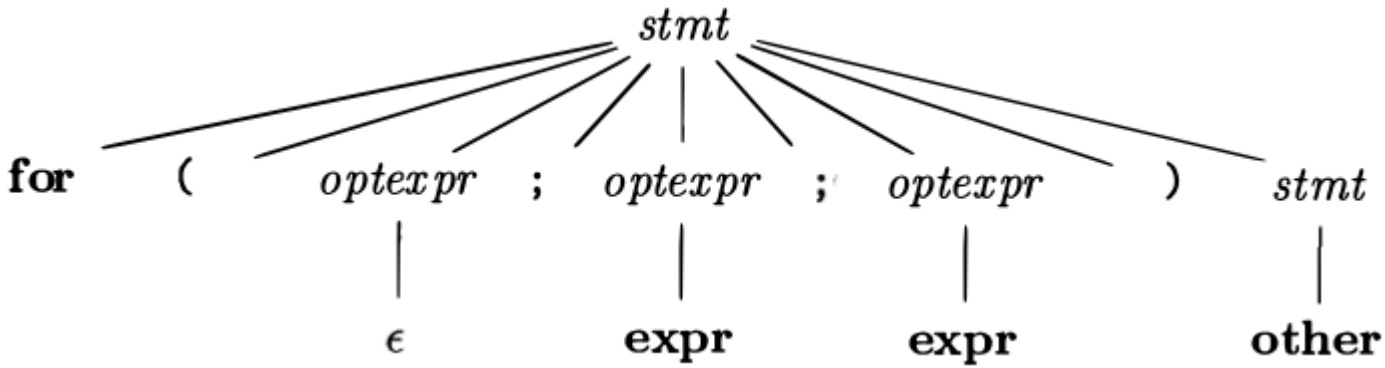
در حین تجزیه، lookahead از ابتدای رشته حرکت می‌کند و به انتهای رشته می‌رسد.

تجزیه گر پایین‌گرد بازگشتی Recursive Descent Parser

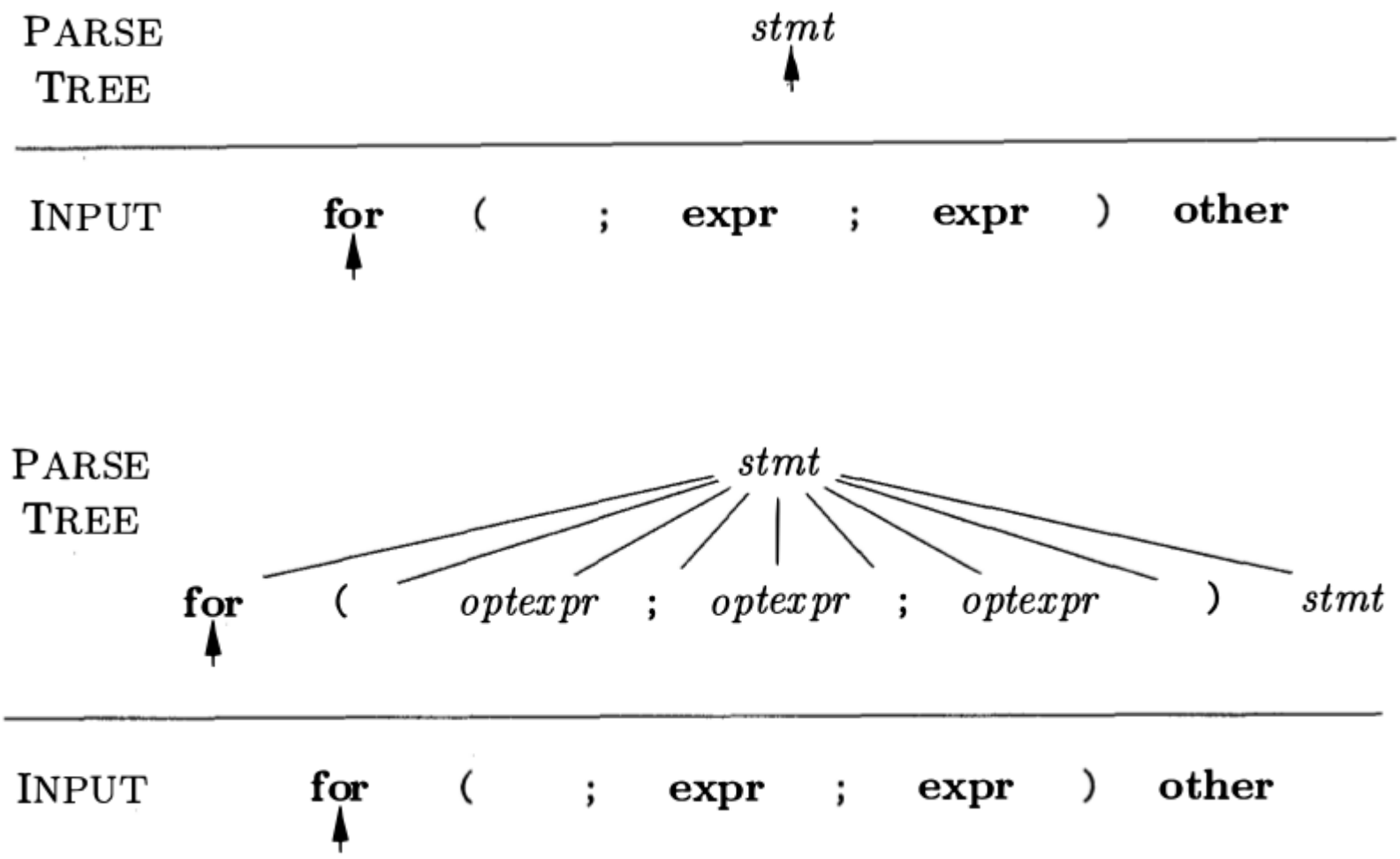
- یک روش تجزیه بالا به پایین
- سعی و خطا

$$
\begin{aligned}
stmt &\rightarrow \mathbf{expr}\ ; \\
&\mid \mathbf{if}\ (\ \mathbf{expr}\ )\ stmt \\
&\mid \mathbf{for}\ (\ optexpr\ ;\ optexpr\ ;\ optexpr\ )\ stmt \\
&\mid \mathbf{other} \\
\\
optexpr &\rightarrow \epsilon \\
&\mid \mathbf{expr}
\end{aligned}
$$

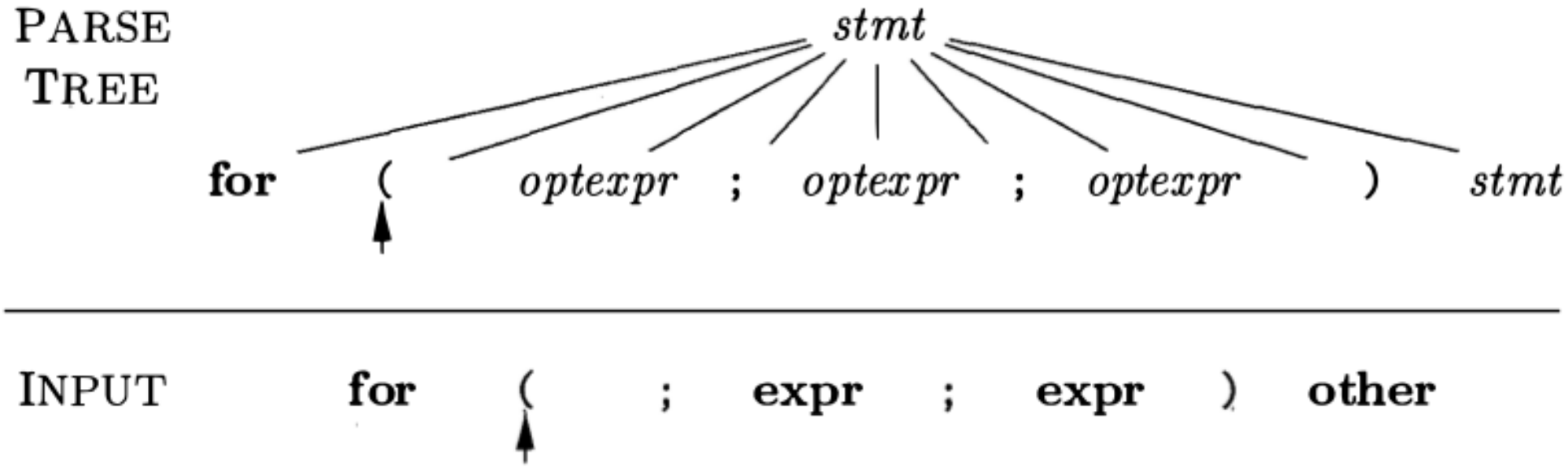
for (; expr ; expr) other



تجزیه گر پایین‌گرد بازگشتی Recursive Descent Parser



تجزیه گر پایین‌گرد بازگشتی Recursive Descent Parser



$$optexpr \rightarrow \epsilon \mid expr$$

تجزیه پیشگویانه Predictive Parsing

- یک روش تجزیه بالا به پایین (بدون سعی و خطا)
- ایجاد یک تابع هر غیرپایانه
  - تصمیم‌گیری در رابطه با انتخاب یک قاعده گرامری (switch – if)
  - استفاده از قاعده انتخاب شده (function call)
- تابع match
  - تطبیق ترمینال درون قاعده گرامری با توکن کنونی (lookahead)

حداقل تعداد توابع = تعداد غیرپایانه‌ها + ۱

تجزیه پیشگویانه Predictive Parsing

- گرامر نمونه

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{expr ;} \\
 & | & \textbf{if (expr)} \ stmt \\
 & | & \textbf{for (} \ optexpr \ \textbf{;} \ optexpr \ \textbf{;} \ optexpr \ \textbf{)} \ stmt \\
 & | & \textbf{other} \\
 & & \\
optexpr & \rightarrow & \epsilon \\
 & | & \textbf{expr}
\end{array}
$$

تجزیه پیشگویانه Predictive Parsing

- تابع match

```
void match(terminal t) {
        if ( lookahead == t ) lookahead = nextTerminal;
        else report("syntax error");
}
```

- تابع مربوط به غیرپایانه optexpr

```
void optexpr() {
        if ( lookahead == expr ) match(expr);
}
```

تجزیه پیشگویانه Predictive Parsing

- تابع مربوط به غیرپایانه stmt

```
void stmt() {
        switch ( lookahead ) {
        case expr:
                match(expr); match(';'); break;
        case if:
                match(if); match('('); match(expr); match(')'); stmt();
                break;
        case for:
                match(for); match('(');
                optexpr(); match(';'); optexpr(); match(';'); optexpr();
                match(')'); stmt(); break;
        case other;
                match(other); break;
        default:
                report("syntax error");
        }
}
```

تجزیه پیشگویانه Predictive Parsing

- تجزیه چگونه انجام می‌شود؟
  - تابع مربوط به سمبل شروع گرامر فراخوانی می‌شود.
  - مقدار lookahead در ابتدا نشان دهنده اولین توکن ورودی است.
- تجزیه رشته زیر

for (; expr ; expr) other

تجزیه پیشگویانه Predictive Parsing

- گرامر نباید چپ‌گردی داشته باشد.
- اگر دو قاعده به صورت زیر داشته باشیم، توکن شروع دو قاعده باید متفاوت باشد.

$$A \rightarrow \alpha$$
$$A \rightarrow \beta$$

دنباله اجرای توابع، درخت تجزیه را ایجاد می‌کند.

چپ‌گردی left recursion

$$A \rightarrow A\ \alpha \mid \beta$$

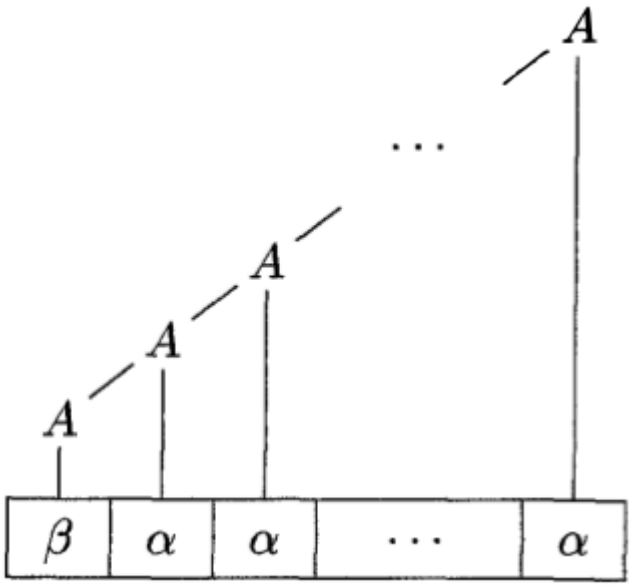
تابع غیرپایانه A؟

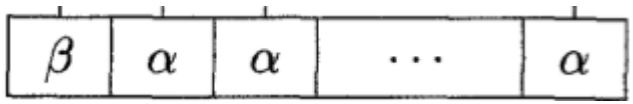
نیاز به یک **گرامر معادل** که چپگرد نباشد.

رفع چپ‌گردی

$$A \rightarrow A\ \alpha \mid \beta$$

- غیرپایانه A چه فرم‌های جمله‌ایی تولید می‌کند؟
- $\beta,\ \beta\alpha,\ \beta\alpha\alpha,\ \beta\alpha\alpha\alpha, \ldots$





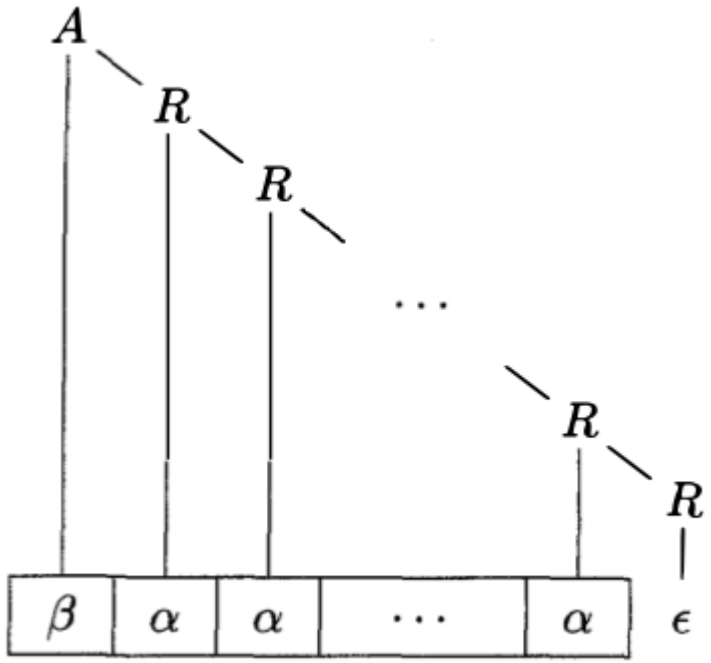
رفع چپ‌گردی

$$A \rightarrow A\ \alpha \mid \beta$$

- غیرپایانه A چه فرم‌های جمله‌ایی تولید می‌کند؟

- $\beta,\ \beta\alpha,\ \beta\alpha\alpha,\ \beta\alpha\alpha\alpha, \ldots$

$$A \rightarrow \beta\ R$$
$$R \rightarrow \alpha\ R \mid \varepsilon$$



حالت کلی چپ‌گردی آشکار

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$$

$$A \to \beta_1 A' \mid \beta_2 A' \mid \cdots \mid \beta_n A'$$
$$A' \to \alpha_1 A' \mid \alpha_2 A' \mid \cdots \mid \alpha_m A' \mid \epsilon$$

چپ‌گردی ضمنی (غیر آشکار)

$$S \rightarrow A\ a \mid b$$
$$A \rightarrow A\ c \mid S\ d \mid \epsilon$$

$$S \Rightarrow Aa \Rightarrow Sda$$

رفع چپ‌گردی ضمنی (غیر آشکار)

$$S \rightarrow A\ a \mid b$$
$$A \rightarrow A\ c \mid S\ d \mid \epsilon$$

- تبدیل چپ‌گردی ضمنی به آشکار
- جایگزینی غیرپایانه با استفاده از قواعد گرامری
- رفع چپ‌گردی آشکار

$$A \rightarrow A\ c \mid A\ a\ d \mid b\ d \mid \epsilon$$

$$S \rightarrow A\ a \mid b$$
$$A \rightarrow b\ d\ A' \mid A'$$
$$A' \rightarrow c\ A' \mid a\ d\ A' \mid \epsilon$$

فاکتورگیری از چپ

- سمت راست دو قاعده گرامری مربوط به یک غیر پایانه با یک فرم جمله‌ای شروع شود.

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{if } expr \textbf{ then } stmt
\end{array}
$$

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$$

$$
\begin{aligned}
A &\rightarrow \alpha A' \\
A' &\rightarrow \beta_1 \mid \beta_2
\end{aligned}
$$

فاکتورگیری از چپ

- سمت راست دو قاعده گرامری مربوط به یک غیر پایانه با یک فرم جمله‌ای شروع شود.

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \cdots \mid \alpha\beta_n \mid \gamma$$

$$\begin{aligned} A &\rightarrow \alpha A' \mid \gamma \\ A' &\rightarrow \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n \end{aligned}$$