اصول طراحی کامپایلر

درس ۷

تحلیل نحوی (۲)

تجزیه‌ی بالا به پایین ـ روش نزولی بازگشتی

Syntax Analysis (2)

Top-Down Parsing – Recursive Descent Method

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

http://courses.fouladi.ir/compiler

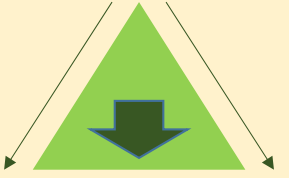
اصول طراحی کامپایلر

تحلیل نحوی
تجزیه‌ی بالا به پایین
روش نزولی بازگشتی

۱

مقدمه

تجزیه‌ی بالا به پایین

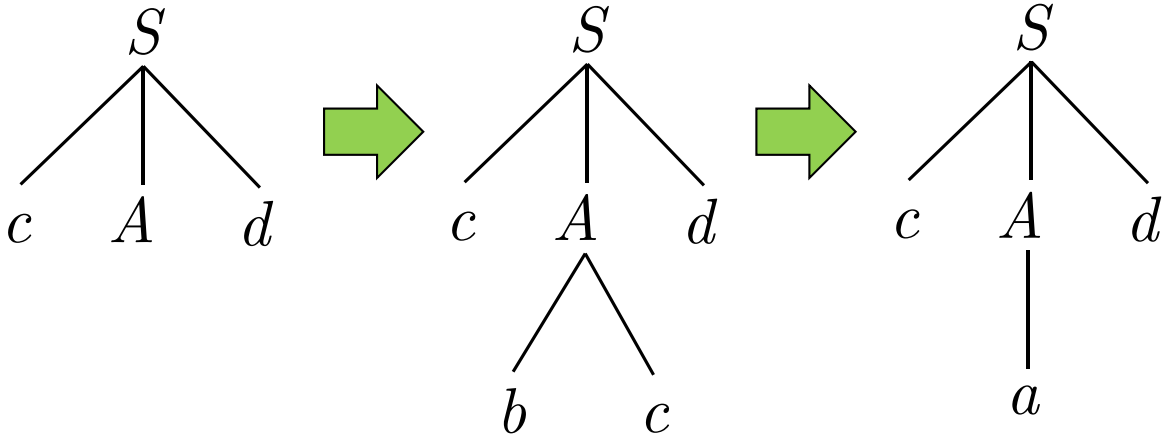
روش‌های تجزیه *Parsing Methods*	
روش‌های بالا به پایین *Top-Down Parsing Methods*	روش‌های پایین به بالا *Bottom-Up Parsing Methods*
ساخت درخت تجزیه از بالا به پایین (از ریشه به سمت برگ‌ها)	ساخت درخت تجزیه از پایین به بالا (از برگ‌ها به سمت ریشه)
	
از **اشتقاق‌های چپ‌ترین** در جهت مستقیم استفاده می‌شود.	

تجزیه‌ی بالا به پایین با عقب‌گرد

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

گرامر

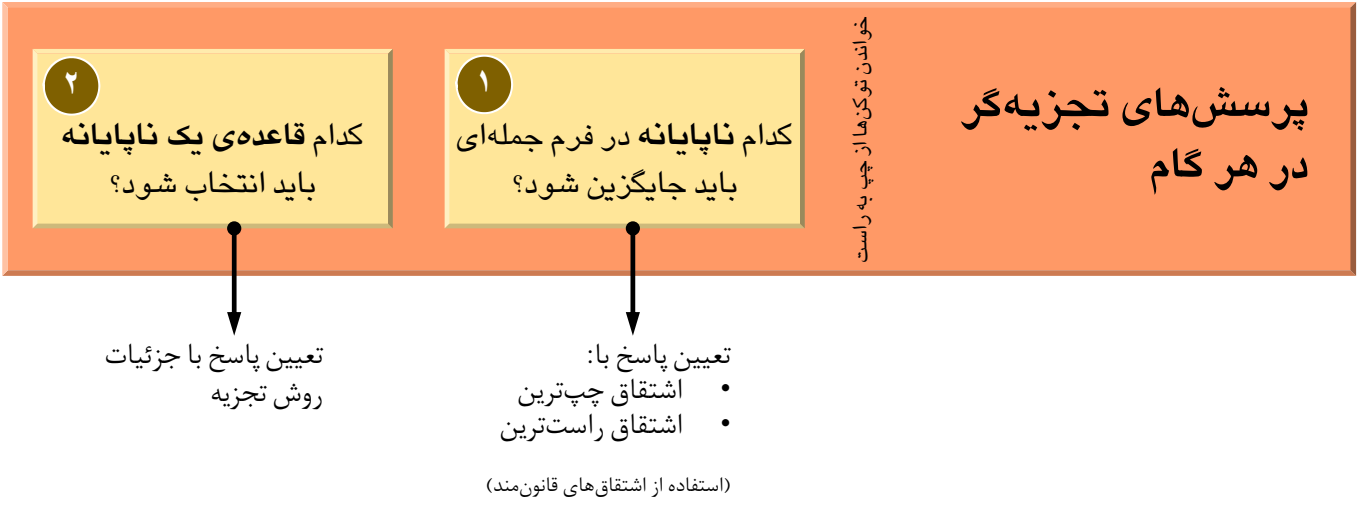
cad رشته‌ی ورودی



Error!! backtrack

پرسش‌های کلیدی تجزیه‌گر در هر گام

گزینه‌های موجود در یک گام اشتقاق



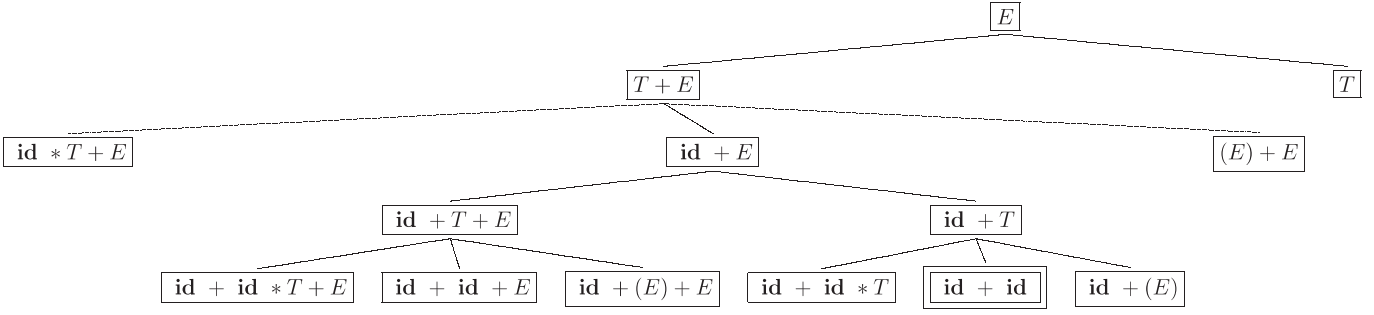
تجزیه‌ی بالا به پایین با عقب‌گرد

مثال

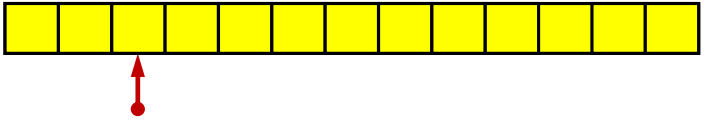
گرامر:

$$E \rightarrow T + E \mid T$$
$$T \rightarrow \mathbf{id} * T \mid \mathbf{id} \mid (E)$$

درخت جستجو برای تجزیه‌ی رشته‌ی $\mathbf{id} + \mathbf{id}$:



توکن جاری (Lookahead)



توکن جاری: توکنی در دنباله‌ی توکن‌ها که اشاره‌گر بر روی آن قرار دارد.

«کاراکتر انتهای رشته» و «قاعده‌ی افزوده»

END-OF-STRING (EOS) & AUGMENTED PRODUCTION

ورودی تجزیه‌گر (دنباله‌ی توکن‌ها) همیشه به **کاراکتر انتهای رشته** ختم می‌شوند

$$\$$$

$$abcd\$$$

برای همین، فرض می‌شود هر گرامر داده شده، یک **قاعده‌ی افزوده** دارد، به صورت:

$$S' \longrightarrow S\$$$

گرامر افزوده
Augmented Grammar

نماد شروع گرامر افزوده

نماد شروع گرامر اصلی

اصول طراحی کامپایلر

تحلیل نحوی
تجزیه‌ی بالا به پایین
روش نزولی بازگشتی

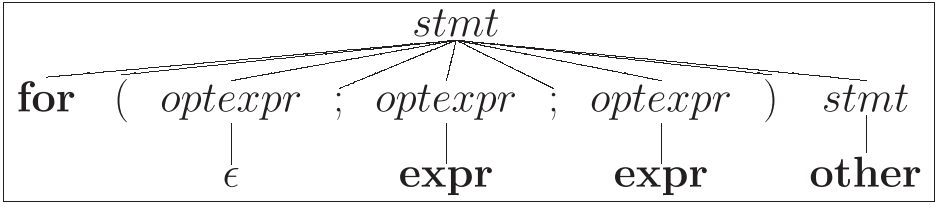
۲

تجزیه‌ی نزولی بازگشتی

تجزیه‌ی بالا به پایین

مثال: تجزیه‌ی بالا به پایین با پویش توکن‌های ورودی از سمت چپ به راست (۱ از ۲)

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



for (; **expr** ; **expr**) **other**

تجزیه‌ی بالا به پایین

مثال: تجزیه‌ی بالا به پایین با پویش توکن‌های ورودی از سمت چپ به راست (۲ از ۲)

stmt	→	**expr** ;
	|	**if** (**expr**) *stmt*
	|	**for** (*optexpr* ; *optexpr* ; *optexpr*) *stmt*
	|	**other**
optexpr	→	ϵ
	|	**expr**

1

PARSE TREE: *stmt*

INPUT: **for (; expr ; expr) other**

2

PARSE TREE: *stmt* → **for** (*optexpr* ; *optexpr* ; *optexpr*) *stmt*

INPUT: **for (; expr ; expr) other**

3

PARSE TREE: *stmt* → **for** (*optexpr* ; *optexpr* ; *optexpr*) *stmt*

INPUT: **for (; expr ; expr) other**

تجزیه‌ی پیش‌بینی‌کننده

PREDICTIVE PARSING

یک روش تجزیه، **پیش‌بینی‌کننده** است،
هرگاه تجزیه‌گر برای گسترش یک ناپایانه بتواند
تنها با نگاه کردن به توکن جاری، قاعده‌ی صحیح برای آن ناپایانه را انتخاب کند.

تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

RECURSIVE PREDICTIVE PARSING

در روش تجزیه‌ی **پیش‌بینی‌کننده‌ی بازگشتی**،
برای هر ناپایانه‌ی گرامر، روالی وجود دارد که
تنها با نگاه کردن به توکن جاری، قاعده‌ی صحیح برای آن ناپایانه را انتخاب کند.

این روال‌ها به صورت بازگشتی و تودرتو یکدیگر را فراخوانی می‌کنند
تا فرآیند تجزیه تمام شود.

تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

مثال ۱ : گرامری ساده بدون قاعده‌ی تهی (۱ از ۴)

$$
\begin{aligned}
type &\rightarrow simple \\
&\mid \textbf{array}\ [simple]\ \textbf{of}\ type \\
simple &\rightarrow \textbf{integer} \\
&\mid \textbf{char} \\
&\mid \textbf{num dotdot num}
\end{aligned}
$$

```
void type()
{
 if (lookahead is in {"integer", "char", "num"}) simple();
 else if (lookahead == "array")
 {
 match("array"); match("["); simple(); match("]"); match("of"); type()
 }
 else error();
}
```

تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

مثال ۱ : گرامری ساده بدون قاعده‌ی تهی (۲ از ۴)

$$
\begin{aligned}
type &\rightarrow simple \\
&\mid \textbf{array}\ [simple]\ \textbf{of}\ type \\
simple &\rightarrow \textbf{integer} \\
&\mid \textbf{char} \\
&\mid \textbf{num dotdot num}
\end{aligned}
$$

```
void simple()
{
 if (lookahead == "integer") match("integer");
 else if (lookahead == "char") match("char");
 else if (lookahead == "num")
 {
 match("num"); match("dotdot"); match("num");
 }
 else error();
}
```

تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

مثال ۱ : گرامری ساده بدون قاعده‌ی تهی (۳ از ۴)

$$
\begin{aligned}
type &\rightarrow simple \\
&\mid \textbf{array}\ [simple]\ \textbf{of}\ type \\
simple &\rightarrow \textbf{integer} \\
&\mid \textbf{char} \\
&\mid \textbf{num dotdot num}
\end{aligned}
$$

```
void match(token t)
{
 if (lookahead == t) lookahead = nextToken();
 else error();
}
```

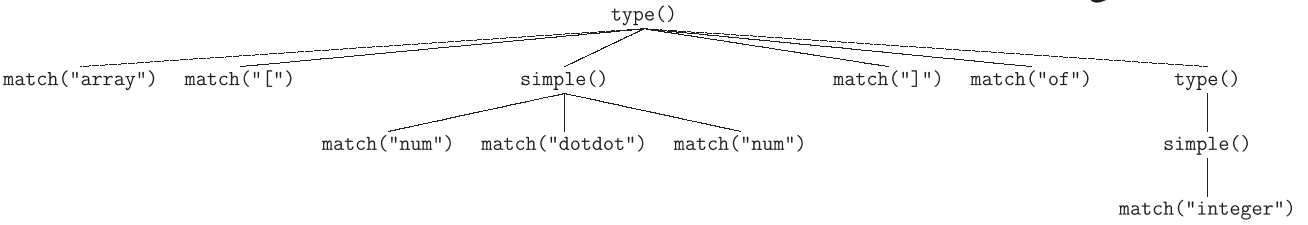
تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

مثال ۱ : گرامری ساده بدون قاعده‌ی تهی (۴ از ۴)

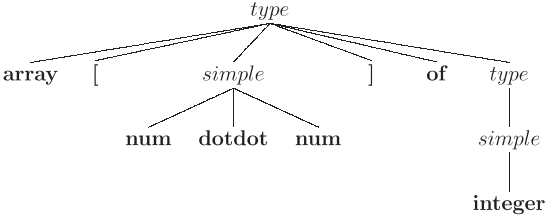
$$
\begin{aligned}
type &\rightarrow simple \\
&\mid \textbf{array}\ [simple]\ \textbf{of}\ type \\
simple &\rightarrow \textbf{integer} \mid \textbf{char} \mid \textbf{num dotdot num}
\end{aligned}
$$

```
array [ num dotdot num ] of integer
```

درخت فراخوانی روال‌ها:



درخت تجزیه:



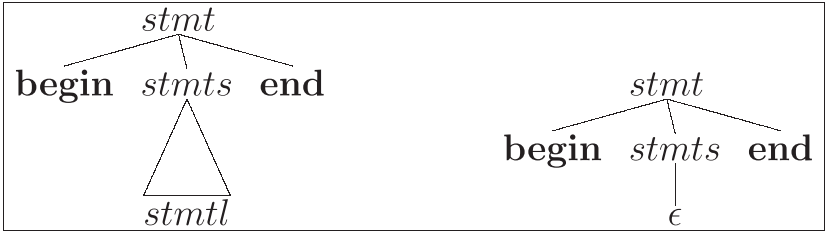
تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

مثال ۲ : استفاده از قاعده‌ی تهی در تجزیه‌ی پیش‌بینی‌کننده‌ی بازگشتی

$$stmt \rightarrow \textbf{begin}\ stmts\ \textbf{end}$$
$$stmts \rightarrow stmtl \mid \epsilon$$

```
void stmts()
{
 if (lookahead == ...) stmtl();
 else if (lookahead == end) return;
 else error();
}
```

begin end



انتخاب قاعده (آلترناتیو) در گام‌های اشتقاق

- Grammar:

$$
\begin{aligned}
S &\rightarrow Ab \mid Bc \\
A &\rightarrow Df \mid CA \\
B &\rightarrow gA \mid e \\
C &\rightarrow dC \mid c \\
D &\rightarrow h \mid i
\end{aligned}
$$

- Sentence: $gchfc$
- The leftmost derivation:

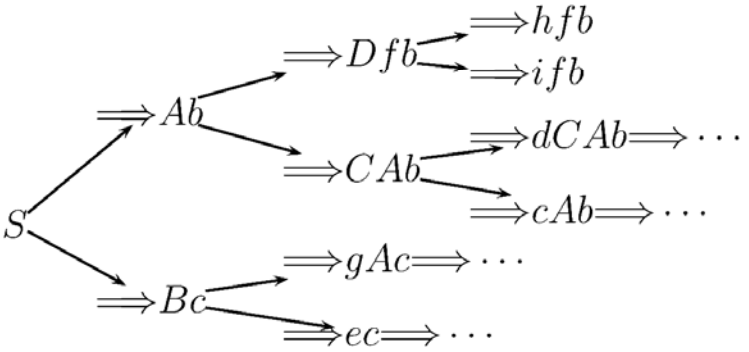
$$
\begin{aligned}
S &\Longrightarrow_{\text{lm}} Bc \Longrightarrow_{\text{lm}} gAc \Longrightarrow_{\text{lm}} gCAc \\
&\Longrightarrow_{\text{lm}} gcAc \Longrightarrow_{\text{lm}} gcDfc \Longrightarrow_{\text{lm}} gchfc
\end{aligned}
$$

انتخاب قاعده (آلترناتیو) در گام‌های اشتقاق

استفاده از نخستین توکن «در فرم‌های جمله‌ای آغاز شده از یک فرم جمله‌ای خاص»/ مثال ۱

- Grammar:

$$
\begin{aligned}
S &\rightarrow Ab \mid Bc \\
A &\rightarrow Df \mid CA \\
B &\rightarrow gA \mid e \\
C &\rightarrow dC \mid c \\
D &\rightarrow h \mid i
\end{aligned}
$$



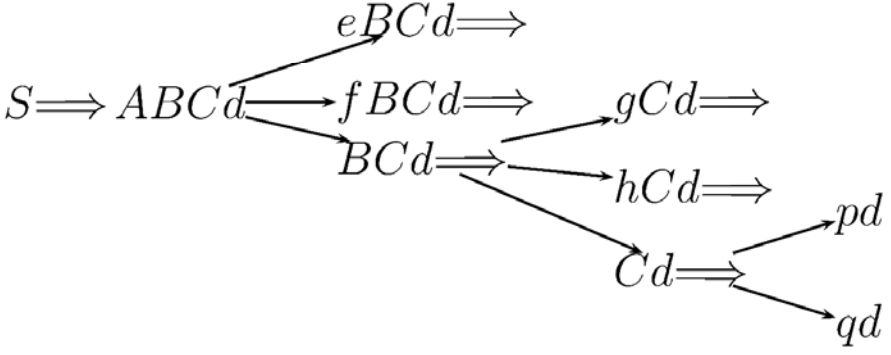
- All possible leftmost derivations:

$\mathsf{First}(Ab) = \{c, d, h, i\}$ $\mathsf{First}(Bc) = \{e, g\}$

انتخاب قاعده (آلترناتیو) در گام‌های اشتقاق

استفاده از نخستین توکن «در فرم‌های جمله‌ای آغاز شده از یک فرم جمله‌ای خاص»/ مثال ۲

$$S \rightarrow ABCd$$
$$A \rightarrow e \mid f \mid \epsilon$$
$$B \rightarrow g \mid h \mid \epsilon$$
$$C \rightarrow p \mid q$$

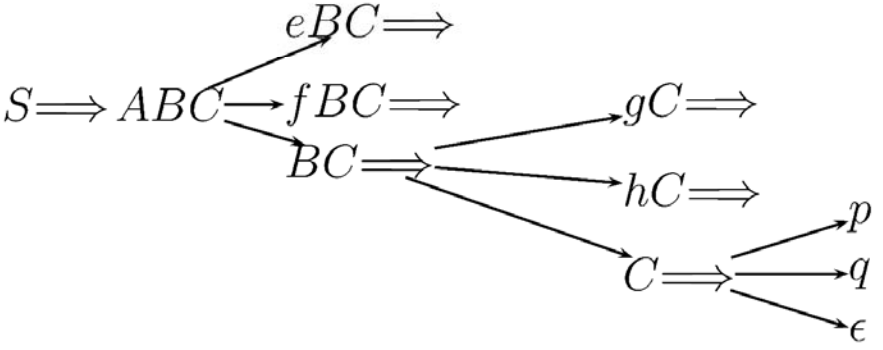


$$\text{First}(ABCd) = \{e, f, g, h, p, q\}$$

انتخاب قاعده (آلترناتیو) در گام‌های اشتقاق

استفاده از نخستین توکن «در فرم‌های جمله‌ای آغاز شده از یک فرم جمله‌ای خاص»/ مثال ۳

$$S \rightarrow ABC$$
$$A \rightarrow e \mid f \mid \epsilon$$
$$B \rightarrow g \mid h \mid \epsilon$$
$$C \rightarrow p \mid q \mid \epsilon$$



$$\textbf{First}(ABC) = \{e, f, g, h, p, q, \epsilon\}$$

تابع سرآغاز (First)

نخستین توکن «در فرم‌های جمله‌ای آغاز شده از یک فرم جمله‌ای خاص»

First هر فرم جمله‌ای
مجموعه‌ی همه‌ی **پایانه‌های** ظاهر شده در ابتدای فرم‌های جمله‌ای آغاز شده از آن فرم‌جمله‌ای است.

و احتمالا رشته‌ی تهی

$$First(\alpha) = \{a : \alpha \Rightarrow^* a\beta\} \cup \{\epsilon : \alpha \Rightarrow^* \epsilon\}$$

تابع سرآغاز (First)

قواعد محاسبه‌ی First

$$First(\epsilon) = \{\epsilon\}$$

$$First(a) = \{a\}$$

$$First(a\beta) = \{a\}$$

$$First(A) = \cup_{i=1}^{k} First(\alpha_i) \quad , A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_k \in P$$

$$First(A\alpha) = \begin{cases} First(A) & , \epsilon \notin First(A) \\ (First(A) - \{\epsilon\}) \cup First(\alpha) & , \epsilon \in First(A) \end{cases}$$

تابع سرآغاز (First)

قواعد محاسبه‌ی First: قاعده‌ی تکمیلی

$$
\begin{aligned}
First(X_1X_2\dots X_m) \;=\; & (First(X_1)-\{\epsilon\}) \\
\cup\; & (First(X_2)-\{\epsilon\}) && ,X_1 \Rightarrow^* \epsilon \\
\cup\; & (First(X_3)-\{\epsilon\}) && ,X_1 \Rightarrow^* \epsilon, X_2 \Rightarrow^* \epsilon \\
& \vdots \\
\cup\; & (First(X_m)) && ,X_1 \Rightarrow^* \epsilon, X_2 \Rightarrow^* \epsilon, \dots, X_{m-1} \Rightarrow^* \epsilon
\end{aligned}
$$

تابع سرآغاز (First)

محاسبه‌ی First: مثال

$$
\begin{aligned}
E &\rightarrow TE' \\
E' &\rightarrow +TE' \mid -TE' \mid \epsilon \\
T &\rightarrow FT' \\
T' &\rightarrow *FT' \mid /FT' \mid \epsilon \\
F &\rightarrow \mathbf{id} \mid (E)
\end{aligned}
$$

$$
\begin{aligned}
First(E) &= First(TE') = \{\mathbf{id}, (\} \\
First(E') &= First(+TE') \cup First(-TE') \cup \{\epsilon\} = \{+, -, \epsilon\} \\
First(+TE') &= \{+\}, \quad First(-TE') = \{-\} \\
First(T) &= First(FT') = \{\mathbf{id}, (\} \\
First(T') &= First(*FT') \cup First(/FT') \cup \{\epsilon\} = \{*, /, \epsilon\} \\
First(*FT') &= \{*\}, \quad First(/FT') = \{/\} \\
First(\mathbf{id}) &= \{\mathbf{id}\}, \quad First((E)) = \{(\} \\
First(F) &= First(\mathbf{id}) \cup First((E)) = \{\mathbf{id}, (\}
\end{aligned}
$$

تابع پیرو (Follow)

نخستین توکن «پس از یک ناپایانه‌ی خاص در فرم‌های جمله‌ای گرامر»

Follow هر ناپایانه
مجموعه‌ی همه‌ی **پایانه‌هایی** است که در فرم‌های جمله‌ای گرامر، پس از آن ناپایانه ظاهر می‌شوند.

$$Follow(A) = \{b : S \Rightarrow^* \alpha A b \beta\}$$

تابع پیرو (Follow)

قواعد محاسبه‌ی Follow

برای یافتن $Follow(A)$ از سمت راست قواعد، جاهایی که در آنها ناپایانه‌ی A ظاهر شده است استفاده می‌کنیم.

$\$ \in Follow(S)$

$A \rightarrow \alpha B$, $Follow(B) \supseteq Follow(A)$

$A \rightarrow \alpha B\beta$, $Follow(B) \supseteq (First(\beta) - \{\epsilon\})$

$A \rightarrow \alpha B\beta,\ \epsilon \in First(\beta)$, $Follow(B) \supseteq Follow(A)$

تذکر: ϵ در Followی هیچ ناپایانه‌ای قرار ندارد!

تابع پیرو (Follow)

محاسبه‌ی Follow: مثال

$$
\begin{aligned}
E &\rightarrow TE' \\
E' &\rightarrow +TE' \mid -TE' \mid \epsilon \\
T &\rightarrow FT' \\
T' &\rightarrow *FT' \mid /FT' \mid \epsilon \\
F &\rightarrow \mathbf{id} \mid (E)
\end{aligned}
$$

$$
\begin{aligned}
Follow(E) &= \{), \$\} \\
Follow(E') &= \{), \$\} \\
Follow(T) &= \{+, -,), \$\} \\
Follow(T') &= \{+, -,), \$\} \\
Follow(F) &= \{+, -, *, /,), \$\}
\end{aligned}
$$

تابع انتخاب (Select)

تعیین‌کننده‌ی توکن‌های انتخاب‌کننده‌ی یک قاعده

$Select$ برای قاعده‌ی $A \rightarrow \alpha$ مجموعه‌ی پایانه‌هایی است که آن قاعده را انتخاب می‌کند:

$$Select(A \rightarrow \alpha) = \begin{cases} First(\alpha) & , \epsilon \notin First(\alpha) \\ (First(\alpha) - \{\epsilon\}) \cup Follow(A) & , \epsilon \in First(\alpha) \end{cases}$$

تجزیه‌ی نزولی بازگشتی

روال‌های لازم

روال‌های لازم برای تجزیه‌ی نزولی بازگشتی
Procedures for Recursive Descent Parsing

روال مطابقت توکن‌ها
`Match(t)`

```
void match(token t)
{
    if (lookahead == t)
        lookahead = nextToken();
    else error();
}
```

روال تجزیه برای هر ناپایانه A
`parseA()`

تجزیه‌ی نزولی بازگشتی

روال parseA() برای هر ناپایانه A

اگر قواعد A به صورت زیر باشد:

$$A \longrightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$$

و هر آلترناتیو α به صورت زیر باشد:

$$X_1 X_2 \ldots X_k$$

دنباله‌ای از فراخوانی k روال را ایجاد می‌کنیم:

X توکن t باشد:	X ناپایانه B باشد:
فراخوانی Match(t)	فراخوانی parseB()

تجزیه‌ی نزولی بازگشتی

روال parseA() برای هر ناپایانه A: موردی که هیچ‌یک از سمت‌راست‌های قواعد آن رشته‌ی تهی تولید نمی‌کنند

$A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$

اگر هیچ یک از α_iها رشته‌ی تهی را تولید نکنند،

```
void parseA()
{
    if (lookahead ∈ First(α1))    {X11(); X12(); ... X1k1();}
    else if (lookahead ∈ First(α2))    {X21(); X22(); ... X2k2();}
    ⋮
    else if (lookahead ∈ First(αm))    {Xm1(); Xm2(); ... Xmkm();}
    else    error();
}
```

تجزیه‌ی نزولی بازگشتی

روال parseA() برای هر ناپایانه A: موردی که یکی از سمت‌راست‌های قواعد آن رشته‌ی تهی تولید می‌کند

$A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$

اگر یکی از α_iها به رشته‌ی تهی ختم شود،

```
void parseA()
{
    if (lookahead ∈ First(α1))    {X11(); X12(); ... X1k1();}
    else if (lookahead ∈ First(α2))    {X21(); X22(); ... X2k2();}
     ⋮
    else if (lookahead ∈ First(αm))    {Xm1(); Xm2(); ... Xmkm();}
    else if (lookahead ∈ Follow(A))    return;
    else    error();
}
```

تجزیه‌ی نزولی بازگشتی

مثال : روال parseA()

Coding parse α_i

- Suppose $\alpha_i = aABbC$, where A, B and C are nonterminals
- parse α_i implemented as:

```
match("a");
parseA();
parseB();
match("b");
parseC();
```

- If $\alpha_i = \epsilon$, then parse α_i implemented as:

```
/* empty statement */
```

اصول طراحی کامپایلر

تحلیل نحوی
تجزیه‌ی بالا به پایین
روش نزولی بازگشتی

۳

گرامر LL(1)

گرامر LL(1)

گرامر G یک گرامر LL(1) است
اگر
در هر گام از پویش رشته از سمت **چپ به راست**
بتوان تنها با نگاه کردن به **توکن ورودی جاری**،
قاعده‌ی استفاده شده در **اشتقاق چپ‌ترین** در آن گام را تعیین کرد.

زبان LL(1) زبانی است که حداقل یک گرامر LL(1) داشته باشد.

شرط پیش‌بینی‌پذیر بودن یک تجزیه‌گر بالا به پایین: LL(1) بودن گرامر است.

گرامر LL(1)

شرایط ریاضی

گرامر G، LL(1) است اگر قواعد

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

در گرامر موجود بود، داشته باشیم:

(I) $First(\alpha) \cap First(\beta) = \varnothing$

(II) if $\alpha \Rightarrow^* \epsilon$ then $First(\beta) \cap Follow(A) = \varnothing$

اگر گرامر قاعده‌ی تهی نداشته باشد، برقراری شرط (I) کافی است.

گرامر LL(1)

شرایط ریاضی در حالت کلی

گرامر G، LL(1) است اگر قواعد

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \ \ldots \ \mid \alpha_k$$

در گرامر موجود بود، داشته باشیم:

(I) $\cup_{i \neq j}(First(\alpha_i) \cap First(\alpha_j)) = \varnothing$

(II) if $\exists j(\alpha_j \Rightarrow^* \epsilon)$ then $\forall i(i \neq j \rightarrow \alpha_i \not\Rightarrow^* \epsilon)$

(III) $First(A) \cap Follow(A) = \varnothing$

شرط (II) به این معنی است که حداکثر یکی از آلترناتیوها باید به رشته‌ی تهی منجر شود.

اگر گرامر قاعده‌ی تهی نداشته باشد، برقراری شرط (I) کافی است.

گرامر LL(1)

گرامر غیر LL(1): مثال ۱

نشان دهید گرامر زیر LL(1) نیست.

$$S \longrightarrow aSbS \mid bSaS \mid \epsilon$$

گرامر LL(1)

گرامر غیر LL(1): مثال ۲

نشان دهید گرامر زیر LL(1) نیست.

$$S \rightarrow XC$$
$$X \rightarrow a \mid \epsilon$$
$$C \rightarrow a \mid \epsilon$$

گرامر LL(1)

گرامر غیر LL(1): مثال ۳

نشان دهید گرامر زیر LL(1) نیست.

$$R \rightarrow \mathbf{id}\, S \mid (R)S$$
$$S \rightarrow +RS \mid .RS \mid *S \mid \epsilon$$

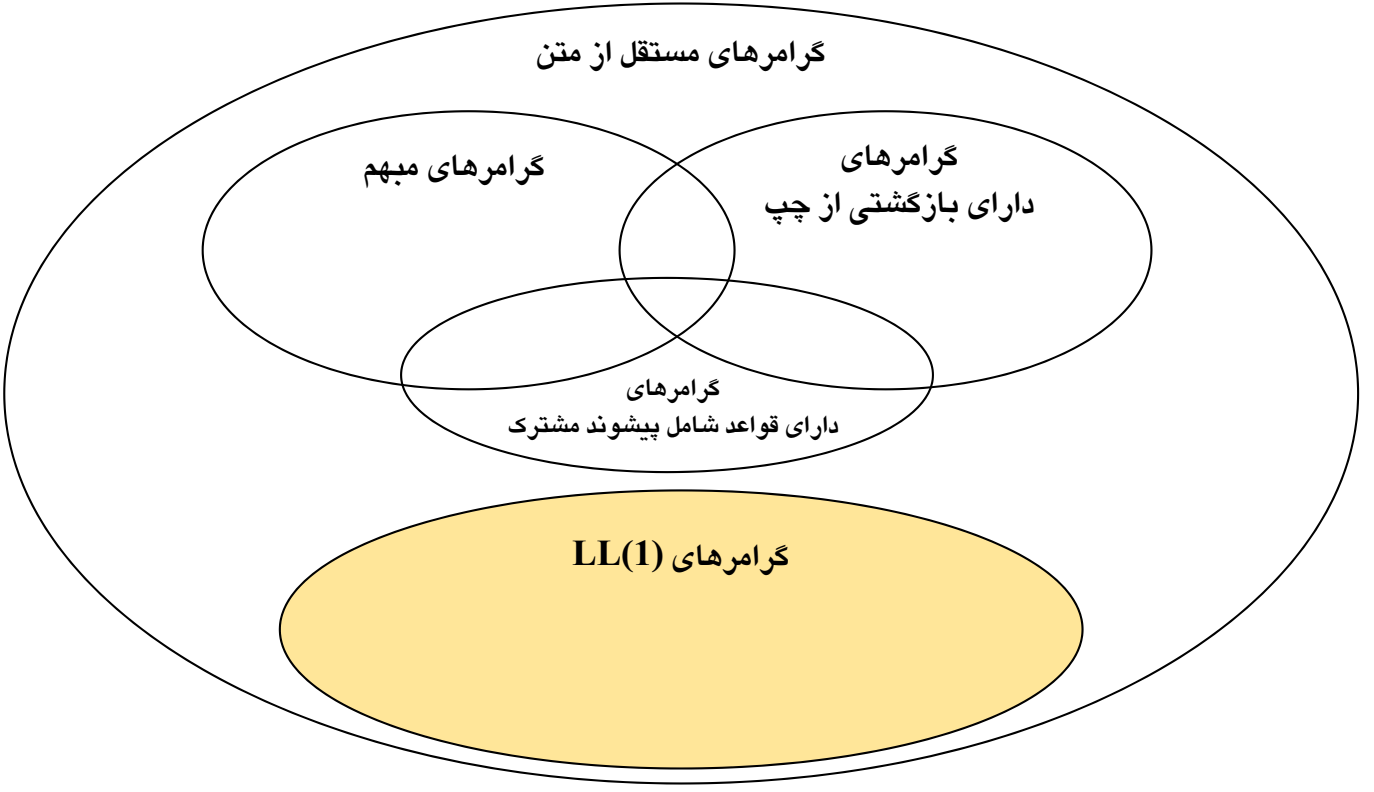
گرامر LL(1)

گرامر غیر LL(1): مثال ۴

نشان دهید گرامر زیر LL(1) نیست.

$$S \rightarrow Bc \mid DB$$
$$B \rightarrow ab \mid cS$$
$$D \rightarrow d \mid \epsilon$$

نسبت گرامرهای LL(1) با انواع گرامرهای مستقل از متن



تجزیه‌ی بالا به پایین

مشکل حلقه‌های نامتناهی

گرامرهای دارای بازگشتی از چپ، باعث می‌شوند
تجزیه‌گر بالا به پایین گرفتار حلقه‌ی نامتناهی شود.

راه حل

رفع بازگشتی از چپ از گرامر

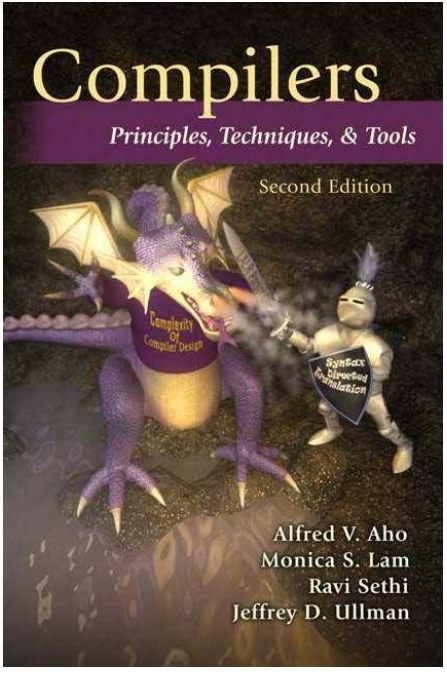
اصول طراحی کامپایلر

تحلیل نحوی
تجزیه‌ی بالا به پایین
روش نزولی بازگشتی

۴

منابع

منبع اصلی



A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman,
Compilers: Principles, Techniques and Tools,
Second Edition, Addison-Wesley, 2007.

Chapter 2 (2.4), Chapter 4 (4.4)