اصول طراحی کامپایلر

درس‌نامه‌ی ۸

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۸
ویراست دوم: ۱۳۹۳

فهرست مطالب

۸ تجزیه‌ی پایین به بالا: روش‌های تقدم ۱

۸-۱ مقدمه ۱

۸-۲ روش تقدم عملگر ۳

۸-۲-۱ گرامر عملگر ۳

۸-۲-۲ روابط تقدم میان پایانه‌ها ۴

توابع لازم برای یافتن روابط تقدم ۴

قواعد یافتن روابط تقدم ۵

۸-۲-۳ جدول تجزیه‌ی تقدم عملگر ۵

۸-۲-۴ روال تجزیه‌ی تقدم عملگر ۶

عمل کاهش در روال تجزیه‌ی تقدم عملگر ۶

۸-۲-۵ وقوع خطا در تجزیه‌ی تقدم عملگر ۷

۸-۲-۶ روش تقدم عملگر: مزایا و معایب ۷

گرامر تقدم عملگر ۸

۸-۲-۷ توابع تقدم ۸

۸-۲-۸ محاسبه‌ی روابط تقدم برای گرامر عبارت‌های ریاضی ۸

۸-۳ روش تقدم ساده ۹

۸-۳-۱ روابط تقدم میان نمادهای گرامر ۹

توابع لازم برای یافتن روابط تقدم ۹
قواعد یافتن روابط تقدم ۱۰
۸-۳-۲ جدول تجزیه‌ی تقدم ساده ۱۱
۸-۳-۳ روال تجزیه‌ی تقدم ساده ۱۱
عمل کاهش در روال تجزیه‌ی تقدم ساده ۱۱
۸-۳-۴ گرامرهای بازگشتی و روش تجزیه‌ی تقدم ساده ۱۲
مشکل بازگشتی از چپ و روش تجزیه‌ی تقدم ساده ۱۲
مشکل بازگشتی از راست و روش تجزیه‌ی تقدم ساده ۱۳
۸-۳-۵ ویژگی‌های روش تجزیه‌ی تقدم ساده ۱۳

❊ تمرین **۱۳**

۸ تجزیه‌ی پایین به بالا: روش‌های تقدم

BOTTOM-UP PARSING: PRECEDENCE METHOD

۸-۱ مقدمه

یکی از ساده‌ترین روش‌های تجزیه‌ی پایین به بالا، روش‌های تقدم می‌باشد که در دو نوع عمده‌ی «تقدم عملگر» و «تقدم ساده» دسته‌بندی می‌شود.

برای توضیح منطق روش‌های تقدم، گرامر زیر را در نظر می‌گیریم:

$$\begin{aligned} E &\rightarrow E + F \mid F \\ F &\rightarrow (E) \mid \mathbf{id} \end{aligned}$$

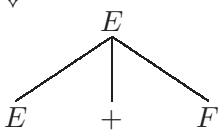
برای اشتقاق رشته‌ی $\mathbf{id} + (\mathbf{id})$ با روش اشتقاق راست‌ترین، خواهیم داشت:

$$\underline{E} \Rightarrow E + \underline{F} \Rightarrow E + (\underline{E}) \Rightarrow E + (\underline{F}) \Rightarrow \underline{E} + (\mathbf{id}) \Rightarrow \underline{F} + (\mathbf{id}) \Rightarrow \mathbf{id} + (\mathbf{id})$$

که مراحل ایجاد درخت اشتقاق و فرم جمله‌ای در هر گام به شکل زیر خواهد بود:

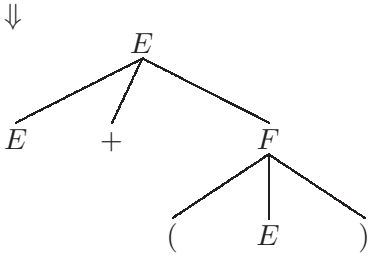
E $\qquad\qquad$ $\lessdot\ E\ \gtrdot$

$\Downarrow$



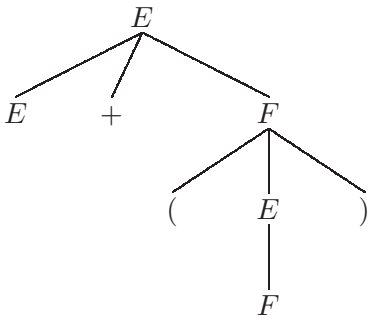
$\lessdot\ E \doteq + \doteq F\ \gtrdot$

$\Downarrow$



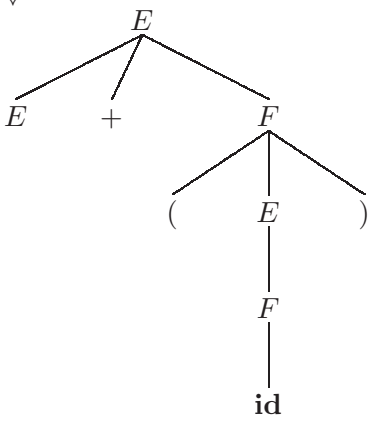
$\lessdot\ E \doteq + \lessdot (\doteq E \doteq) \gtrdot\ \gtrdot$

$\Downarrow$



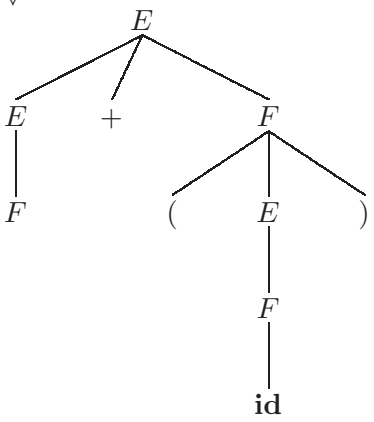
$\lessdot E \doteq + \lessdot (\lessdot F \gtrdot) \gtrdot \gtrdot$

$\Downarrow$



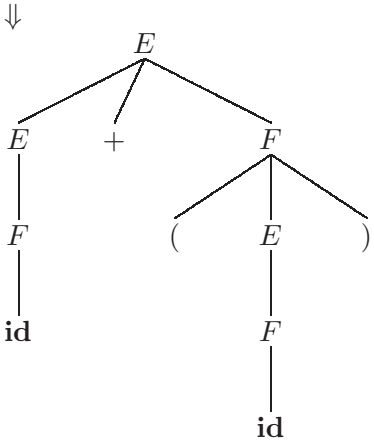
$\lessdot E \doteq + \lessdot (\lessdot \lessdot \mathbf{id} \gtrdot \gtrdot) \gtrdot \gtrdot$

$\Downarrow$



$\lessdot \lessdot F \gtrdot + \lessdot (\lessdot \lessdot \mathbf{id} \gtrdot \gtrdot) \gtrdot \gtrdot$

$\Downarrow$



$\lessdot \lessdot \lessdot \mathbf{id} \gtrdot \gtrdot + \lessdot (\lessdot \lessdot \mathbf{id} \gtrdot \gtrdot) \gtrdot \gtrdot$

هر درخت اشتقاق جزیی، بیانگر یک فرم جمله‌ای است. فرم جمله‌ای در قالب سلسله‌مراتبی خود (درختی) نمایش داده شده است، به طوری که آغاز هر زیردرخت با $\lessdot$ و پایان هر زیردرخت با $\gtrdot$ نمایش داده شده است و بین عناصر هم سطح در یک زیردرخت نماد $\doteq$ قرارگرفته است.

با مقایسه‌ی این نمایش با فرم جمله‌ای متناظر در فرایند اشتقاق درمی‌یابیم که با پویش رشته از سمت چپ به راست، دستگیره، همیشه سمت چپ‌ترین عبارتی است که بین دو علامت $\lessdot$ و $\gtrdot$ قرارگرفته است و درون آن دیگر علامت $\lessdot$ و $\gtrdot$ وجود ندارد. در اشتقاق زیر، زیر دستگیره در هر گام خط کشیده شده است:

$$E \Rightarrow \underline{E + F} \Rightarrow E + \underline{(E)} \Rightarrow E + (\underline{F}) \Rightarrow E + (\underline{\mathbf{id}}) \Rightarrow \underline{F} + (\mathbf{id}) \Rightarrow \underline{\mathbf{id}} + (\mathbf{id})$$

در جدول زیر، ستون‌ها به ترتیب، فرم جمله‌ای جاری در اشتقاق، نمایش درختی فرم جمله‌ای با نمادهای تعریف شده، توکن‌های خوانده شده از سمت چپ به راست و دستگیره‌ی جاری که زیر آن خط کشیده شده است، را نشان می‌دهند:

$\Rightarrow \underline{\mathbf{id}} + (\mathbf{id})$	$\lessdot \lessdot \underline{\lessdot \mathbf{id} \gtrdot} \gtrdot + \lessdot (\lessdot \lessdot \mathbf{id} \gtrdot \gtrdot) \gtrdot \gtrdot$	$\mathbf{id}$	$\lessdot \lessdot \underline{\lessdot \mathbf{id}}$
$\Rightarrow \underline{F} + (\mathbf{id})$	$\lessdot \underline{\lessdot F \gtrdot} + \lessdot (\lessdot \lessdot \mathbf{id} \gtrdot \gtrdot) \gtrdot \gtrdot$	$\mathbf{id}$	$\lessdot \underline{\lessdot F}$
$\Rightarrow E + (\underline{\mathbf{id}})$	$\lessdot E \doteq + \lessdot (\lessdot \underline{\lessdot \mathbf{id} \gtrdot} \gtrdot) \gtrdot \gtrdot$	$\mathbf{id} + (\mathbf{id}$	$\lessdot E \doteq + \lessdot (\underline{\lessdot \mathbf{id}}$
$\Rightarrow E + (\underline{F})$	$\lessdot E \doteq + \lessdot (\underline{\lessdot F \gtrdot}) \gtrdot \gtrdot$	$\mathbf{id} + (\mathbf{id})$	$\lessdot E \doteq + \lessdot (\underline{\lessdot F}$
$\Rightarrow E + (\underline{E})$	$\lessdot E \doteq + \underline{\lessdot (\doteq E \doteq) \gtrdot} \gtrdot$	$\mathbf{id} + (\mathbf{id})$	$\lessdot E \doteq + \underline{\lessdot (\doteq E \doteq)}$
$\Rightarrow \underline{E + F}$	$\underline{\lessdot E \doteq + \doteq F \gtrdot}$	$\mathbf{id} + (\mathbf{id})$	$\underline{\lessdot E \doteq + \doteq F}$
E	$\lessdot E \gtrdot$	$\mathbf{id} + (\mathbf{id})$	$\lessdot E$

در روش‌های تقدم، با استفاده از نمادهایی که مرز دستگیره را مشخص می‌کنند، و قرار دادن این نمادها به همراه فرم جمله‌ای جاری در پشته‌ی تجزیه‌گر، مرحله‌ی شناسایی دستگیره ـ که مهم‌ترین مرحله در هر گام تجزیه‌ی پایین به بالا است ـ به صورت سرراست انجام می‌شود. چگونگی قرار دادن این نمادها با توجه به منطق استفاده از آنها در هر یک از روش‌ها تشریح می‌شود.

۸-۲ روش تقدم عملگر

۸-۲-۱ گرامر عملگر

گرامر عملگر (operator grammar)، گرامری است که

۱) قاعده‌ی ϵ ندارد.

۲) در سمت راست هیچ یک از قواعد آن **دو ناپایانه‌ی مجاور** وجود نداشته باشد.

شرط لازم برای استفاده از روش تجزیه‌ی تقدم عملگر این است که گرامر مورد استفاده، یک گرامر عملگر باشد.

مثال

گرامر زیر، یک گرامر عملگر نیست:

$$\begin{aligned} E &\rightarrow EAE \mid (E) \mid \mathbf{id} \\ A &\rightarrow + \mid - \mid * \mid / \end{aligned}$$

اما گرامر زیر معادل با گرامر بالاست و یک گرامر عملگر است:

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

در به دست آوردن این گرامر جدید، از قاعده‌ی جایگزینی استفاده کرده‌ایم. ملاحظه می‌کنیم که در هیچ قاعده‌ای دو ناپایانه‌ی مجاور دیده نمی‌شود.

۸-۲-۲ روابط تقدم میان پایانه‌ها

بین هر دو پایانه‌ی گرامر، a و b، چهار رابطه‌ی تقدم ممکن است:

a مقدم بر b	$a \gtrdot b$	(a باید قبل از b کاهش یابد)
a موخر از b	$a \lessdot b$	(a باید بعد از b کاهش یابد)
a هم‌تقدم با b	$a \doteq b$	(a و b هر دو در یک دستگیره قرار دارند)
a بدون رابطه با b	$a \square b$	

در مورد روابط تقدم، توجه به نکات زیر لازم است:

- این روابط متقارن نیستند. همچنین برگردان تقارنی نیز نیستند (یعنی $a \gtrdot b$ به معنی $b \lessdot a$ نیست).
- ممکن است بین دو پایانه هیچ رابطه‌ی تقدمی موجود نباشد.
- ممکن است بین دو پایانه چند رابطه‌ی تقدم موجود باشد.

توابع لازم برای یافتن روابط تقدم

- $FirstTerm(A)$ نخستین پایانه‌ی فرم‌های جمله‌ای حاصل از ناپایانه‌ی A:

$$FirstTerm(A) = \{a : A \Rightarrow^+ a\alpha \vee A \Rightarrow^+ Ba\alpha\}$$

- $LastTerm(A)$ آخرین پایانه‌ی فرم‌های جمله‌ای حاصل از ناپایانه‌ی A:

$$LastTerm(A) = \{a : A \Rightarrow^+ \alpha a \vee A \Rightarrow^+ \alpha aB\}$$

برای محاسبه‌ی تابع $FirstTerm(A)$ می‌توانیم از گراف اشتقاق‌های چپ‌ترین گرامر با شروع از ناپایانه‌ی A و برای محاسبه‌ی تابع $LastTerm(A)$ می‌توانیم از گراف اشتقاق‌های راست‌ترین گرامر با شروع از ناپایانه‌ی A استفاده کنیم.

مثال

برای گرامر

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid \mathbf{id} \end{aligned}$$

توابع فوق الذکر را محاسبه می‌کنیم:

$$\begin{aligned} FirstTerm(E) &= \{+, *, (, \mathbf{id}\} & LastTerm(E) &= \{+, *,), \mathbf{id}\} \\ FirstTerm(T) &= \{*, (, \mathbf{id}\} & LastTerm(T) &= \{*,), \mathbf{id}\} \\ FirstTerm(F) &= \{(, \mathbf{id}\} & LastTerm(F) &= \{), \mathbf{id}\} \end{aligned}$$

قواعد یافتن روابط تقدم

با توجه به قواعد گرامر و ملاحظه‌ی سمت راست آنها، می‌توان از جدول زیر برای یافتن قواعد تقدم استفاده نمود:

		بین a و b حداکثر یک ناپایانه باشد
$a \doteq b$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots ab \cdots \vee U \rightarrow \cdots aWb \cdots)$
$a \lessdot FirstTerm(W)$		a قبل از یک ناپایانه W
$a \lessdot b$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots aW \cdots), b \in FirstTerm(W)$
$LastTerm(W) \gtrdot b$		b بعد از یک ناپایانه W
$a \gtrdot b$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots Wb \cdots), a \in LastTerm(W)$
		عدم وجود رابطه‌ی تقدم بین a و b
$a \square b$	$\Leftrightarrow$	$\neg(a \lessdot b \vee a \doteq b \vee a \gtrdot b)$

برای یافتن رابطه‌ی \$ و سایر پایانه‌ها از قاعده‌ی افزوده‌ی $S' \rightarrow \$S\$$ استفاده می‌شود.

مثال

برای گرامر

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

با استفاده از توابع محاسبه شده در مثال قبل، خواهیم داشت:

$$\begin{aligned} \$ &\doteq \$ \\ (&\doteq) \\ \$ &\lessdot FirstTerm(E) \\ + &\lessdot FirstTerm(T) \\ * &\lessdot FirstTerm(F) \\ (&\lessdot FirstTerm(E) \\ LastTerm(E) &\gtrdot \$ \\ LastTerm(E) &\gtrdot + \\ LastTerm(T) &\gtrdot * \\ LastTerm(E) &\gtrdot) \end{aligned}$$

۸-۲-۳ جدول تجزیه‌ی تقدم عملگر

جدول تجزیه‌ی تقدم عملگر، یک جدول مربعی با ابعاد $(|T| + ۱)(|T| + ۱)$ است.
این جدول تجزیه شامل روابط تقدم میان ناپایانه‌هاست:

$$P : (T \cup \{\$\}) \times (T \cup \{\$\}) \rightarrow \{\lessdot, \doteq, \gtrdot, \square\}$$

مثال

با استفاده از روابط محاسبه شده در مثال قبل برای گرامر

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

جدول تجزیه‌ی تقدم عملگر این گرامر به صورت زیر به دست می‌آید:

	$+$	$*$	$($	$)$	**id**	$\$$
$+$	$\gtrdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
$*$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
$($	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	$\lessdot$	
$)$	$\gtrdot$	$\gtrdot$		$\gtrdot$		$\gtrdot$
id	$\gtrdot$	$\gtrdot$		$\gtrdot$		$\gtrdot$
$\$$	$\lessdot$	$\lessdot$	$\lessdot$		$\lessdot$	$\doteq$

۸-۲-۴ روال تجزیه‌ی تقدم عملگر

Operator Precedence Parsing Procedure

$push(\$)$
repeat
  $a \leftarrow top\text{-}terminal(Stack)$
  $b \leftarrow lookahead$

if $P[a,b] = \lessdot$ **then** $push(\lessdot); shift(b)$
if $P[a,b] = \doteq$ **then** $push(\doteq); shift(b)$
if $P[a,b] = \gtrdot$ **then** $reduce()$
if $P[a,b] = \square$ **then** $error()$
if $a = b = \$$ **then** $accept()$

عمل کاهش در روال تجزیه‌ی تقدم عملگر

عمل reduce():

۱) یافتن دستگیره‌ی بالای پشته،

۲) حذف دستگیره،

۳) قرار دادن ناپایانه‌ی نوعی N در بالای پشته (می‌توان به طورکلی از قرار دادن ناپایانه در پشته، صرف نظر کرد.)[۱]

یافتن دستگیره:
آن‌قدر در پشته پایین می‌رویم تا به اولین نماد $\lessdot$ برسیم.
رشته‌ی میان نماد $\lessdot$ و بالای پشته (و ناپایانه‌ی زیر $\lessdot$ در صورت وجود) دستگیره است.

مثال

با استفاده از جدول تجزیه‌ی به دست آمده در مثال قبلی، رشته‌ی $\mathbf{id} + \mathbf{id} * \mathbf{id}$ را با روش تجزیه‌ی تقدم عملگر، تجزیه می‌کنیم:

Stack	Input	Action
$	$\mathbf{id} + \mathbf{id} * \mathbf{id}$$	shift
$$\underline{\lessdot \mathbf{id}}$	$+ \mathbf{id} * \mathbf{id}$$	reduce $F \to \mathbf{id}$
$$N$	$+ \mathbf{id} * \mathbf{id}$$	shift
$$N \lessdot +$	$\mathbf{id} * \mathbf{id}$$	shift
$$N \lessdot + \underline{\lessdot \mathbf{id}}$	$*\mathbf{id}$$	reduce $F \to \mathbf{id}$
$$N \lessdot + N$	$*\mathbf{id}$$	shift
$$N \lessdot + N \lessdot *$	$\mathbf{id}$$	shift
$$N \lessdot + N \lessdot * \underline{\lessdot \mathbf{id}}$	$	reduce $F \to \mathbf{id}$
$$N \lessdot + \underline{N \lessdot * N}$	$	reduce $T \to T * F$
$$\underline{N \lessdot + N}$	$	reduce $E \to E + T$
$$N$	$	accept

۸-۲-۵ وقوع خطا در تجزیه‌ی تقدم عملگر

در تجزیه‌ی تقدم عملگر، در دو صورت امکان وقوع خطا وجود دارد:

- عدم وجود رابطه میان a (پایانه‌ی بالای پشته) و b (توکن جاری)
- عدم تطبیق دستگیره‌ی واقع در بالای پشته با سمت راست هیچ یک از قواعد

۸-۲-۶ روش تقدم عملگر: مزایا و معایب

- **مزیت:** پیاده‌سازی ساده
- **معایب:**
  ۱) محدودیت زیاد
  ۲) غیر قابل استفاده برای گرامرهای شامل عملگرهایی یکسان با دو تقدم مختلف (مانند منهای دوتایی و تکی)

[۱] تشخیص قاعده‌ی استفاده شده در هر مرحله، بر مبنای این روش، فقط بر اساس ترتیب پایانه‌های مشاهده شده در دستگیره است که در بخش بالایی پشته قابل دسترس می‌باشد. برای همین، در عمل ناپایانه‌ها نقشی بازی نمی‌کنند. در نتیجه در این روش امکان کارکردن با گرامرهایی که یک ترتیب از پایانه‌ها را در بیش از یک قاعده داشته باشند، وجود ندارد (به دلیل تداخل کاهش ـ کاهش)؛ برای مثال نمی‌توانیم دو قاعده به صورت $E \to E + T$ و $F \to F + H$ داشته باشیم.

۳) امکان یافت نشدن برخی از خطاهای نحوی (گاهی اوقات منشأ خطاهای نحوی به درستی شناسایی نمی‌شود.)
۴) حتی گاهی نگران این هستیم که زبان پذیرفته شده توسط تجزیه‌گر با زبان تولید شده توسط گرامر مساوی نباشد! (به دلیل عدم توجه کافی این روش به ناپایانه‌ها)

گرامر تقدم عملگر

گرامر تقدم عملگر، گرامری است که جدول تجزیه‌ی تقدم عملگر آن تداخلی نداشته باشد.

۸-۲-۷ توابع تقدم

توابع تقدم دو تابع f و g به صورت

$$f, g : T \cup \{\$\} \rightarrow \{۰, ۱, ۲, \ldots\}$$

هستند که برای آنها داشته باشیم:

$a \gtrdot b$	اگر و فقط اگر	$f(a) > g(b)$
$a \doteq b$	اگر و فقط اگر	$f(a) = g(b)$
$a \lessdot b$	اگر و فقط اگر	$f(a) < g(b)$

اشکال اساسی وارد به توابع تقدم این است که اگر بین a و b رابطه‌ی تقدم موجود نباشد، یعنی $a \square b$، در این صورت تشخیص آن ممکن نیست.

۸-۲-۸ محاسبه‌ی روابط تقدم برای گرامر عبارت‌های ریاضی

$$E \rightarrow E\theta_۱E \mid E\theta_۲E \mid \ldots \mid E\theta_mE \mid \alpha E \mid E\alpha \mid (E) \mid \mathbf{id}$$

بر اساس روابط تقدم و شرکت‌پذیری عملگرهای $\theta_۱$، ...، θ_m، می‌توان معادل غیر مبهم این گرامر را به دست آورد. از طریقی دیگر، روابط تقدم بر اساس قواعد زیر نیز قابل حصول است:

$\text{precedence}(\theta_1) > \text{precedence}(\theta_2) \Rightarrow \theta_1 \gtrdot \theta_2, \theta_2 \lessdot \theta_1$
$\text{precedence}(\theta_1) = \text{precedence}(\theta_2)$ or $\theta_1 = \theta_2$
- if θ_1 is left-associative: $\theta_1 \gtrdot \theta_2, \theta_2 \gtrdot \theta_1$
- if θ_1 is right-associative: $\theta_1 \lessdot \theta_2, \theta_2 \lessdot \theta_1$

if α is a unary operator:
- prefix, $\text{precedence}(\alpha) > \text{precedence}(\theta_i)$: $\theta_i \lessdot \alpha$ for all θ_i
- postfix, $\text{precedence}(\alpha) > \text{precedence}(\theta_i)$: $\theta_i \lessdot \alpha$ for all θ_i

$\theta_i \lessdot \mathbf{id}, \quad \mathbf{id} \gtrdot \theta_i$ for all θ_i
$\theta_i \lessdot (, \quad (\gtrdot \theta_i$ for all θ_i
$\theta_i \gtrdot), \quad) \lessdot \theta_i$ for all θ_i
$\theta_i \gtrdot \$, \quad \$ \lessdot \theta_i$ for all θ_i
$(\doteq), \quad \$ \lessdot (, \quad \$ \lessdot \mathbf{id}$
$(\lessdot (, \quad \mathbf{id} \gtrdot \$, \quad) \gtrdot \$$
$(\lessdot \mathbf{id}, \quad \mathbf{id} \gtrdot), \quad) \gtrdot)$

۸-۳ روش تقدم ساده

روش تقدم ساده، شبیه روش تقدم عملگر است، با این تفاوت که روابط تقدم، بین همه‌ی نمادهای گرامر (پایانه و ناپایانه) تعریف می‌شود.

گرامر قابل استفاده در روش تقدم ساده گرامری است که

۱) قاعده‌ی ϵ ندارد.

۲) سمت راست هیچ دو قاعده‌ای در آن یکسان نیست (برای جلوگیری از تداخل کاهش ـ کاهش).

شرط لازم برای استفاده از روش تجزیه‌ی تقدم ساده این است که گرامر مورد استفاده، شرایط فوق را داشته باشد.

۸-۳-۱ روابط تقدم میان نمادهای گرامر

بین هر دو نماد گرامر، X و Y، چهار رابطه‌ی تقدم ممکن است:

X مقدم بر Y	$X \gtrdot Y$	(X باید قبل از Y کاهش یابد)
X موخر از Y	$X \lessdot Y$	(X باید بعد از Y کاهش یابد)
X هم‌تقدم با Y	$X \doteq Y$	(X و Y هر دو در یک دستگیره قرار دارند)
X بدون رابطه با Y	$X \square Y$	

در مورد این روابط، توجه به نکات زیر لازم است:

- این روابط متقارن نیستند. همچنین برگردان تقارنی نیز نیستند (یعنی $X \gtrdot Y$ به معنی $Y \lessdot X$ نیست).
- ممکن است بین دو نماد گرامر هیچ رابطه‌ی تقدمی موجود نباشد.
- ممکن است بین دو نماد گرامر چند رابطه‌ی تقدم موجود باشد.

توابع لازم برای یافتن روابط تقدم

- $Head(A)$ نخستین نماد گرامر در فرم‌های جمله‌ای حاصل از ناپایانه‌ی A:

$$Head(A) = \{X : A \Rightarrow^{+} X\alpha\}$$

- $Tail(A)$ آخرین نماد گرامر در فرم‌های جمله‌ای حاصل از ناپایانه‌ی A:

$$Tail(A) = \{X : A \Rightarrow^{+} \alpha X\}$$

مثال

برای گرامر

$$\begin{aligned} S &\rightarrow (SS) \\ S &\rightarrow c \end{aligned}$$

توابع $Head$ و $Tail$ به صورت زیر محاسبه می‌شوند:

$$Head(S) = \{c, (\} \qquad Tail(S) = \{c,)\}$$

قواعد یافتن روابط تقدم

با توجه به قواعد گرامر و ملاحظه‌ی سمت راست آنها، می‌توان از جدول زیر برای یافتن قواعد تقدم استفاده نمود:

		دو نماد گرامر X و Y کنار هم
$X \doteq Y$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots XY \cdots)$
$X \lessdot Head(A)$		یک نماد گرامر X قبل از یک ناپایانه A
$X \lessdot Y$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots XA \cdots), Y \in Head(A)$
$Tail(A) \gtrdot Y$		یک ناپایانه A قبل از یک نماد گرامر Y
$X \gtrdot Y$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots AY \cdots), X \in Tail(A)$
$Tail(A) \gtrdot Head(B)$		دو ناپایانه A و B کنار هم
$X \gtrdot Y$	$\Leftrightarrow$	$\exists U(U \rightarrow \cdots AB \cdots), X \in Tail(A), Y \in Head(B)$
		عدم وجود رابطه‌ی تقدم بین X و Y
$X \square Y$	$\Leftrightarrow$	$\neg(X \lessdot Y \vee X \doteq Y \vee X \gtrdot Y)$

برای یافتن رابطه‌ی \$ و سایر پایانه‌ها از قاعده‌ی افزوده‌ی $S' \rightarrow \$S\$$ استفاده می‌شود.

مثال

برای گرامر

$$\begin{aligned} S' &\rightarrow \$S\$ \\ S &\rightarrow (SS) \\ S &\rightarrow c \end{aligned}$$

با توجه به توابع محاسبه شده در مثال قبل، روابط تقدم زیر را داریم:

$$\begin{aligned} \$ &\doteq S \\ S &\doteq \$ \\ (&\doteq S \\ S &\doteq S \\ S &\doteq) \\ \$ &\lessdot Head(S) \\ (&\lessdot Head(S) \\ S &\lessdot Head(S) \\ Tail(S) &\gtrdot \$ \\ Tail(S) &\gtrdot S \\ Tail(S) &\gtrdot) \\ Tail(S) &\gtrdot Head(S) \end{aligned}$$

۸-۳-۲ جدول تجزیه‌ی تقدم ساده

جدول تجزیه‌ی تقدم ساده، یک جدول مربعی با ابعاد $(|N|+|T|+۱)(|N|+|T|+۱)$ است.
این جدول تجزیه شامل روابط تقدم میان نمادهای گرامر (پایانه و ناپایانه) است:

$$P : (N \cup T \cup \{\$\}) \times (N \cup T \cup \{\$\}) \rightarrow \{\lessdot, \doteq, \gtrdot, \square\}$$

مثال

برای گرامر

$$\begin{aligned} S' &\rightarrow \$S\$ \\ S &\rightarrow (SS) \\ S &\rightarrow c \end{aligned}$$

با توجه به روابط محاسبه شده در مثال قبل، جدول تجزیه‌ی تقدم ساده به صورت زیر به دست می‌آید:

	S	(	)	c	\$
S	$\doteq$	$\lessdot$	$\doteq$	$\lessdot$	$\doteq$
(	$\doteq$	$\lessdot$		$\lessdot$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
c	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
\$	$\doteq$	$\lessdot$		$\lessdot$	

۸-۳-۳ روال تجزیه‌ی تقدم ساده

Simple Precedence Parsing Procedure

$push(\$)$
repeat
  $X \leftarrow top(Stack)$
  $b \leftarrow lookahead$
if $P[X,b] = \lessdot$ **then** $push(\lessdot); push(b)$
if $P[X,b] = \doteq$ **then** $push(\doteq); push(b)$
if $P[X,b] = \gtrdot$ **then** $reduce()$
if $P[X,b] = \square$ **then** $error()$
if $X = b = \$$ **then** $accept()$

عمل کاهش در روال تجزیه‌ی تقدم ساده

عمل reduce():

- یافتن دستگیره‌ی بالای پشته،
- عنصر بالای پشته پس از حذف دستگیره: TOP
- سمت چپ قاعده‌ی مورد استفاده در کاهش: LHS

$$\begin{array}{llll} \textbf{if} & P[TOP, LHS] = \lessdot & \textbf{then} & push(\lessdot); push(LHS) \\ \textbf{if} & P[TOP, LHS] = \doteq & \textbf{then} & push(\doteq); push(LHS) \\ \textbf{if} & P[TOP, LHS] = \square & \textbf{then} & error() \end{array}$$

یافتن دستگیره:
آن‌قدر در پشته پایین می‌رویم تا به اولین نماد $\gtrdot$ برسیم.
رشته‌ی میان نماد $\gtrdot$ و بالای پشته دستگیره است.

مثال

با استفاده از جدول تجزیه‌ی به دست آمده در مثال قبلی، رشته‌ی $(c(cc))$ را با روش تجزیه‌ی تقدم ساده، تجزیه می‌کنیم:

Stack	Input	Action
$\$$	$(c(cc))\$$	shift
$\$ \lessdot ($	$c(cc))\$$	shift
$\$ \lessdot (\underline{\lessdot c}$	$(cc))\$$	reduce $S \to c$, $TOP = (, LHS = S$
$\$ \lessdot (\doteq S$	$(cc))\$$	shift
$\$ \lessdot (\doteq S \lessdot ($	$cc))\$$	shift
$\$ \lessdot (\doteq S \lessdot (\underline{\lessdot c}$	$c))\$$	reduce $S \to c$, $TOP = (, LHS = S$
$\$ \lessdot (\doteq S \lessdot (\doteq S$	$c))\$$	shift
$\$ \lessdot (\doteq S \lessdot (\doteq S \underline{\lessdot c}$	$))\$$	reduce $S \to c$, $TOP = S, LHS = S$
$\$ \lessdot (\doteq S \lessdot (\doteq S \doteq S$	$))\$$	shift
$\$ \lessdot (\doteq S \underline{\lessdot (\doteq S \doteq S)}$	$)\$$	reduce $S \to SS$, $TOP = S, LHS = S$
$\$ \lessdot (\doteq S \doteq S$	$)\$$	shift
$\$ \underline{\lessdot (\doteq S \doteq S \doteq)}$	$\$$	reduce $S \to SS$, $TOP = S, LHS = S$
$\$ \doteq S$	$\$$	accept

۴-۳-۸ گرامرهای بازگشتی و روش تجزیه‌ی تقدم ساده

مشکل بازگشتی از چپ و روش تجزیه‌ی تقدم ساده

اگر قواعد بازگشتی از چپ به صورت

$$\begin{array}{lcl} U & \to & U \cdots \\ V & \to & \cdots XU \cdots \end{array}$$

در گرامر موجود باشد، موجب بروز تداخل میان روابط نماد X و ناپایانه‌ی U می‌شود:

$$X \doteq U, \qquad X \lessdot U$$

برای حل این مشکل، با معرفی ناپایانه‌ی جدید W، مجموعه قواعد فوق را با مجموعه قواعد زیر جایگزین می‌کنیم:

$$\begin{array}{lcl} W & \rightarrow & U \\ U & \rightarrow & U \cdots \\ V & \rightarrow & \cdots X W \cdots \end{array}$$

که در این صورت روابط به صورت زیر در می‌آیند:

$$X \doteq W, \qquad X \lessdot U$$

مشکل بازگشتی از راست و روش تجزیه‌ی تقدم ساده

اگر قواعد بازگشتی از راست به صورت

$$\begin{array}{lcl} U & \rightarrow & \cdots U \\ V & \rightarrow & \cdots U X \cdots \end{array}$$

در گرامر موجود باشد، موجب بروز تداخل میان روابط نماد X و ناپایانه‌ی U می‌شود:

$$U \doteq X, \qquad U \gtrdot X$$

برای حل این مشکل، با معرفی ناپایانه‌ی جدید W، مجموعه قواعد فوق را با مجموعه قواعد زیر جایگزین می‌کنیم:

$$\begin{array}{lcl} W & \rightarrow & U \\ U & \rightarrow & \cdots U \\ V & \rightarrow & \cdots W X \cdots \end{array}$$

که در این صورت روابط به صورت زیر در می‌آیند:

$$W \doteq X, \qquad U \gtrdot X$$

۸-۳-۵ ویژگی‌های روش تجزیه‌ی تقدم ساده

- بهبود یافته‌ی روش تقدم عملگر
- محدودیت کمتری دارد و گرامرهای بیشتری را پشتیبانی می‌کند.
- مجاز بودن ناپایانه‌های مجاور در سمت راست قواعد گرامر

◄ تمرین

۱. گرامر زیر را در نظر بگیرید:

$$\begin{array}{lcl} S & \rightarrow & (L) \mid a \\ L & \rightarrow & L, S \mid S \end{array}$$

برای این گرامر، جدول تجزیه تقدم ـ عملگر را به دست آورید و با استفاده از آن رشته‌های زیر را تجزیه کنید:

$$(a, a) \qquad (a, ((a, a), (a, a)))$$

۲. روابط تقدم عملگر را برای گرامرهای زیر تولید کنید:

آ) $S \to aSbS \mid bSaS \mid \epsilon$

ب)
$$\begin{aligned} bexpr &\to bexpr \ \mathbf{or} \ bterm \mid bterm \\ bterm &\to bterm \ \mathbf{and} \ bfactor \mid bfactor \\ bfactor &\to \mathbf{not} \ bfactor \mid (bexpr) \mid \mathbf{true} \mid \mathbf{false} \end{aligned}$$

۳. یک تجزیه‌گر تقدم ساده برای گرامر زیر بسازید و رشته‌ی $\mathbf{var}(\mathbf{var}, \mathbf{var})$ را با آن تجزیه کنید:

$$\begin{aligned} F &\to \mathbf{var}(P_\circ) \\ P_\circ &\to P \\ P &\to P, P_1 \mid P_1 \\ P_1 &\to \mathbf{var} \mid F \end{aligned}$$

۴. یک تجزیه‌گر تقدم ساده برای گرامر زیر بسازید. چگونه باید تداخل‌های جدول را حذف کنیم تا رفتار تجزیه‌گر قاعده‌ی «تطابق هر **else** با نزدیکترین **then** تطابق‌نیافته» را دنبال کند؟

$$\begin{aligned} stmt &\to \mathbf{if\ expr\ then}\ stmt \\ &\mid \mathbf{if\ expr\ then}\ stmt\ \mathbf{else}\ stmt \\ &\mid \mathbf{other} \end{aligned}$$

۵. الگوریتم‌های مناسبی برای محاسبه‌ی توابع زیر ارایه دهید که در آنها A یک ناپایانه است.

- $FirstTerm(A)$
- $LastTerm(A)$
- $Head(A)$
- $Tail(A)$

۶. نشان دهید هر گرامر مستقل از متن می‌تواند به یک گرامر عملگر (operator-grammar) تبدیل شود که قواعد تولید آن به یکی از صورت‌های زیر است:

$$A \to aBcC,\ A \to aBb,\ A \to aB,\ A \to a$$

و اگر ϵ در زبان وجود داشته باشد، $S \to \epsilon$ هم یک قاعده‌ی تولید باشد.

۷. آیا گرامر زیر، یک گرامر عملگر است؟ چرا؟ در صورت منفی بودن پاسخ، معادل عملگر آن را بنویسید:

$$\begin{aligned} E &\to EAE \mid (E) \mid -E \mid \mathbf{id} \\ A &\to + \mid - \mid * \mid / \end{aligned}$$