اصول طراحی کامپایلر

درس‌نامه‌ی ۱۰

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۸
ویراست دوم: ۱۳۹۳

فهرست مطالب

۱۰ برخورد با خطاهای نحوی در تجزیه‌گر ۱

۱۰-۱ خطاهای نحوی ۱

۱۰-۱-۱ برخورد با خطا و روش‌های تجزیه ۱

۱۰-۱-۲ گزارش خطا ۱

۱۰-۲ استراتژی‌های گذر از خطا ۲

۱۰-۲-۱ استراتژی حالت وحشت ۲

۱۰-۲-۲ استراتژی سطح عبارت ۲

۱۰-۲-۳ استراتژی قواعد خطا ۳

۱۰-۲-۴ استراتژی تصحیح سراسری ۳

۱۰-۳ خطاهای نحوی در روش LL(1) ۳

۱۰-۳-۱ گذر از خطا در روش LL(1): استراتژی حالت وحشت ۳

انتخاب عناصر مجموعه‌ی SYNCH(A) ۴

۱۰-۳-۲ گذر از خطا در روش LL(1): استراتژی سطح عبارت ۵

۱۰-۳-۳ گذر از خطا در روش LL(1): استراتژی قواعد خطا ۶

۱۰-۴ خطاهای نحوی در روش‌های تقدم ۶

۱۰-۴-۱ گذر از خطا در روش‌های تقدم: حالت عدم وجود رابطه ۶

۱۰-۴-۲ گذر از خطا در روش‌های تقدم: حالت عدم تطابق با سمت راست قواعد . . . ۷

۱۰-۵ خطاهای نحوی در روش‌های LR(1) ۸
۱۰-۵-۱ گذر از خطا در روش‌های LR: استراتژی حالت وحشت ۸
۱۰-۵-۲ گذر از خطا در روش‌های LR: استراتژی سطح عبارت ۹

❊ تمرین ۱۱

۱۰ برخورد با خطاهای نحوی در تجزیه‌گر

SYNTAX ERROR HANDLING IN PARSER

۱۰-۱ خطاهای نحوی

اغلب، کشف خطا و ترمیم آن در مرحله‌ی تحلیل نحوی انجام می‌شود؛ زیرا:

- طبیعت بسیاری از خطاها نحوی است.
- ویژگی‌های موجود در روش‌های تجزیه می‌تواند خطاهای نحوی را با دقت بالایی کشف کند.

انتظارات موجود از ماژول برخورد با خطا در تجزیه‌گر را می‌توان به صورت زیر فهرست کرد:

- گزارش دقیق و صریح وجود خطا
- گذر سریع از روی خطا به منظور کشف خطاهای بعدی
- عدم لطمه زدن به سرعت پردازش برنامه‌های صحیح
- * حدس زدن منظور برنامه‌نویس در محل خطا

۱۰-۱-۱ برخورد با خطا و روش‌های تجزیه

روش‌های LL و LR در سریع‌ترین زمان ممکن یک خطا را کشف می‌کنند. در واقع، این روش‌ها دارای **خصوصیت viable-prefix** هستند:

> هرگاه پیشوندی از ورودی، پیشوند هیچ رشته‌ی معتبری در زبان نباشد، خطا کشف می‌شود.

۱۰-۱-۲ گزارش خطا

گزارش خطا شامل دو جزء است:

- گزارش محل خطا در برنامه‌ی منبع
- گزارش نوع خطا

۱۰-۲ استراتژی‌های گذر از خطا

استراتژی حالت وحشت	Panic mode strategy
استراتژی سطح عبارت	Phrase level strategy
استراتژی قواعد خطا	Error production strategy
استراتژی تصحیح سراسری	Global correction strategy

معمولاً برای برخورد با خطا از ترکیبی از استراتژی‌های فوق استفاده می‌شود.

۱۰-۲-۱ استراتژی حالت وحشت (Panic mode)

تجزیه‌گر در هنگام کشف خطا، نمادهای ورودی را تا رسیدن به یک توکن مناسب دور می‌ریزد.

توکن مناسب می‌تواند

- توکن پایان‌دهنده‌ی یک ساختار (مانند ; یا end و ...)
- یا توکن آغازکننده‌ی یک ساختار (مانند کلمات کلیدی) باشد.

ویژگی‌های این استراتژی را می‌توان به صورت زیر برشمرد:

- (مزیت) ساده‌ترین استراتژی برای پیاده‌سازی
- (مزیت) می‌تواند توسط اکثر روش‌های تجزیه به کار گرفته شود.
- (مزیت) ضمانت می‌شود که تجزیه‌گر درون یک حلقه‌ی بی‌پایان قرار نگیرد (زیرا طول باقیمانده‌ی ورودی به مرور کم می‌شود).
- (عیب) ممکن است حجم زیادی از ورودی حذف شود.
- در مواردی که وقوع چندین خطا در یک دستور به‌ندرت اتفاق می‌افتد، این روش بسیار مناسب است.

۱۰-۲-۲ استراتژی سطح عبارت (Phrase level)

تجزیه‌گر یک تصحیح محلی بر روی باقیمانده‌ی ورودی انجام می‌دهد.

تصحیح محلی مانند تعویض پیشوندی از باقیمانده‌ی ورودی با رشته‌ای که به تجزیه‌گر امکان ادامه‌ی کار بدهد، مانند:

- تعویض , با ;
- حذف یک ; اضافی
- درج یک ; جاافتاده

در انتخاب تعویض‌ها باید مراقب باشیم که در حلقه‌ی نامتناهی قرار نگیریم.
ضعف عمده‌ی این روش، در برخورد با مواردی است که در آنها خطای اصلی قبل از نقطه‌ی تشخیص آن رخ داده باشد.

۱۰-۲-۳ استراتژی قواعد خطا (Error production)

قواعد گرامری خطاهای متداول به گرامر اصلی اضافه می‌شود.

اگر ایده‌های خوبی از خطاهای رایج داشته باشیم، می‌توان گرامر زبان را به گونه‌ای گسترش داد که قواعد تولید دارای ساختارهای خطا را نیز شامل شود.
سپس از این گرامر توسعه‌یافته با تولیدات خطا برای ساخت تجزیه‌گر استفاده می‌کنیم.
اگر یک قاعده‌ی خطا توسط تجزیه‌گر به کار رفت، مانند کشف خطا با آن برخورد می‌شود و می‌توان روال برخورد با خطای متناظر با آن را فراخوانی کرد و ساختار دارای خطا را گزارش داد.

۱۰-۲-۴ استراتژی تصحیح سراسری (Global correction)

مطلوب این است که کامپایلر هنگام پردازش رشته‌های ناصحیح از ورودی تغییرات اندکی را اعمال کند.

رشته‌ی ورودی ناصحیح x و گرامر G در اختیار ماست. باید یک درخت تجزیه برای رشته‌ی صحیح متناظر y پیدا شود به طوری که تعداد درج‌ها، حذف‌ها و تغییرهای توکن‌های مورد نیاز برای تبدیل x به y کمترین تعداد ممکن باشد.

یک مشکل این است که رشته‌ی صحیح y ممکن است مورد نظر برنامه‌نویس نبوده باشد.

(الگوریتم‌های Minimum Edit Distance) این روش‌ها هزینه‌ی زمانی و فضایی زیادی در پیاده‌سازی دارند و لذا تنها جنبه تئوری دارند. از این روش می‌توان به عنوان معیاری برای ارزیابی تکنیک‌های گذر از خطا استفاده نمود. به علاوه، از آن می‌توان برای یافتن رشته‌های بهینه برای جایگزینی در استراتژی تصحیح خطای سطح عبارت استفاده کرد.

۱۰-۳ خطاهای نحوی در روش LL(1)

منشأ خطاهای نحوی در روش $\mathrm{LL}(1)$ دو مورد است:

- خالی بودن خانه‌ی $M[A, a]$ در جدول تجزیه (A ناپایانه‌ی بالای پشته، a توکن جاری)
- عدم تطابق پایانه‌ی بالای پشته با توکن جاری

۱۰-۳-۱ گذر از خطا در روش LL(1): استراتژی حالت وحشت

تجزیه‌گر در هنگام کشف خطا، نمادهای ورودی را تا رسیدن به یک توکن مناسب دور می‌ریزد.

توکن مناسب: توکنی که در مجموعه‌ی $\textsc{Synch}(A)$ قرار دارد. (A ناپایانه‌ی بالای پشته است.)

- در صورت خالی بودن خانه‌ی $M[A, a]$ در جدول تجزیه:
  حذف توکن جاری از ورودی

- در صورت قرار داشتن علامت synch در خانه‌ی $M[A, a]$:
حذف ناپایانه‌ی بالای پشته A **و ادامه‌ی تجزیه** (اگر A تنها ناپایانه‌ی موجود در پشته نباشد)
حذف توکن جاری از ورودی و ادامه‌ی تجزیه (اگر A تنها ناپایانه‌ی موجود در پشته باشد)
- در صورت عدم تطابق پایانه‌ی بالای پشته با توکن جاری:
حذف پایانه‌ی بالای پشته و ادامه‌ی تجزیه

انتخاب عناصر مجموعه‌ی Synch(A)

SYNCH(A) می‌تواند از عناصر زیر تشکیل شود:

- $Follow(A)$:
اگر A بالای پشته باشد، عمل حذف می‌تواند آن‌قدر انجام شود تا به نماد بعد از ساختار A برسیم.
- توکن‌های آغازگر ساختارهای مختلف (مانند کلمات کلیدی)
- $First(A)$:
اگر A بالای پشته باشد، عمل حذف می‌تواند با کنارگذاشتن توکن‌های اضافی به ابتدای ساختار A برسد.

در خانه‌های خالی جدول تجزیه زیر عناصر SYNCH(A) نماد synch را قرار می‌دهیم.

هدف از مجموعه‌ی SYNCH این است که حتی‌الامکان زیاد از ورودی توکن حذف نشود.

مثال

این مثال، گذر از خطا در روش LL(1) با استراتژی حالت وحشت را نشان می‌دهد. برای گرامر

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

مجموعه‌های SYNCH از روی $Follow$ی ناپایانه‌ها محاسبه شده است:

	id	+	*	(	)	$
E	TE'			TE'	synch	synch
E'		$+TE'$			ϵ	ϵ
T	FT'	synch		FT'	synch	synch
T'		ϵ	$*FT'$		ϵ	ϵ
F	**id**	synch	synch	(E)	synch	synch

حال با استفاده از این جدول تجزیه، می‌توانیم یک رشته‌ی دارای خطا را به شکل زیر تجزیه کنیم:

Stack	Input	Output
$\$E$	) **id** $*+$**id**\$	error, skip)
$\$E$	**id** $*+$**id**\$	$E \rightarrow TE'$
$\$E'T$	**id** $*+$**id**\$	
$\$E'T'F$	**id** $*+$**id**\$	
$\$E'T'$**id**	**id** $*+$**id**\$	
$\$E'T'$	$*+$ **id**\$	
$\$E'T'F*$	$*+$ **id**\$	
$\$E'T'F$	$+$**id**\$	error, M[F,+] = synch
$\$E'T'$	$+$**id**\$	
$\$E'$	$+$**id**\$	
$\$E'T+$	$+$**id**\$	
$\$E'T$	**id**\$	
$\$E'T'F$	**id**\$	
$\$E'T'$**id**	**id**\$	
$\$E'T'$	\$	
$\$E'$	\$	
\$	\$	finish

۲-۳-۱۰ گذر از خطا در روش LL(1): استراتژی سطح عبارت

تجزیه‌گر یک تصحیح محلی بر روی باقیمانده‌ی ورودی انجام می‌دهد.

خانه‌های خالی در جدول تجزیه با اشاره‌گرهایی به روال اداره‌کننده‌ی خطا پر می‌شود. روال‌های اداره‌کننده‌ی خطا می‌توانند:

- ورودی را تغییر دهند،
- از ورودی توکن حذف کنند،
- در ورودی توکن درج کنند،
- پیغام‌های خطای مناسب چاپ کنند،
- از بالای پشته حذف کنند.

بایستی اطمینان پیدا کرد که امکان بروز حلقه‌ی نامتناهی در اثر تصحیح خطا وجود ندارد. (برای مثال اطمینان از کاهش یافتن تدریجی طول باقیمانده‌ی ورودی)

۱۰-۳-۳ گذر از خطا در روش LL(1): استراتژی قواعد خطا

قواعد گرامری خطاهای متداول به گرامر اصلی اضافه می‌شود.

مثال

گرامر زیر را در نظر می‌گیریم:

$$
\begin{array}{lcl}
S & \rightarrow & S\$ \\
S & \rightarrow & \textbf{if e then } S \\
S & \rightarrow & \textbf{if e then } S \textbf{ else } S \\
S & \rightarrow & \textbf{a } := \textbf{ e}; S \\
S & \rightarrow & \textbf{while e do } S
\end{array}
$$

با اضافه کردن قواعد خطاهای متداول، به گرامر زیر می‌رسیم:

$$
\begin{array}{lcll}
S & \rightarrow & S\$ & (0) \\
S & \rightarrow & \textbf{if e then } S & (1) \\
S & \rightarrow & \textbf{if e then } S \textbf{ else } S & (2) \\
S & \rightarrow & \textbf{a } := \textbf{ e}; S & (3) \\
S & \rightarrow & \textbf{while e do } S & (4) \\
S & \rightarrow & \textbf{a } := \textbf{ e}S & (-3) \\
S & \rightarrow & \textbf{while e } S & (-4)
\end{array}
$$

- قاعده‌ی (۳−): متداول بودن خطای جا افتادن ;
- قاعده‌ی (۴−): متداول بودن خطای جا افتادن **do**

۱۰-۴ خطاهای نحوی در روش‌های تقدم

منشأ خطاهای نحوی در روش‌های تقدم دو مورد است:

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

۱۰-۴-۱ گذر از خطا در روش‌های تقدم: حالت عدم وجود رابطه

در حالت عدم وجود رابطه میان توکن جاری و ناپایانه‌ی بالای پشته از استراتژی **سطح عبارت** استفاده می‌شود.

تجزیه‌گر یک تصحیح محلی بر روی باقیمانده‌ی ورودی انجام می‌دهد.

در خانه‌های خالی جدول تجزیه، اشاره‌گرهایی به زیرروال‌های اداره‌کننده‌ی خطا قرار داده می‌شود. اگر در عمل تجزیه به یک خانه‌ی خالی رجوع شد، زیرروال متناظر فراخوانی می‌شود و عبور از خطا را به طور مناسب انجام می‌دهد.

مثال

برای نمایش گذر از خطا در روش تقدم عملگر در حالت عدم وجود رابطه، گرامر زیر را در نظرمی‌گیریم

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

که جدول تجزیه‌ی زیر برای آن به دست می‌آید. خانه‌های خالی را با اشاره‌گرهایی به روال‌های خطا پر می‌کنیم:

	+	(	)	**id**	$
+	$\gtrdot$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\doteq$	$\lessdot$	e_3
)	$\gtrdot$	e_2	$\gtrdot$	e_2	$\gtrdot$
id	$\gtrdot$	e_2	$\gtrdot$	e_2	$\gtrdot$
$	$\lessdot$	$\lessdot$	e_1	$\lessdot$	$\doteq$

معنای روال‌های خطای متناظر با اشاره‌گرها می‌تواند به صورت زیر باشد:

e_1	عبارت با پرانتز بسته شروع شده است
e_2	پس از **id** یا پرانتز بسته، **id** یا پرانتز باز آمده است (گم شدن عملگر بین آنها)
e_3	عبارت با پرانتز باز خاتمه یافته است

که متناظر با هر یک، کنش مناسبی انجام شده و پیام خطایی به کاربر نمایش داده می‌شود:

e_1	توکن (را حذف کن	پیام خطا: unbalanced parenthesis
e_2	توکن + را به ورودی اضافه کن	پیام خطا: missing operator
e_3	توکن) را از بالای پشته حذف کن	پیام خطا: missing parenthesis

۱۰-۴-۲ گذر از خطا در روش‌های تقدم: حالت عدم تطابق با سمت راست قواعد

در حالت عدم تطابق دستگیره با سمت راست هیچ یک از قواعد تولید، نیز از استراتژی **سطح عبارت** استفاده می‌شود.

تجزیه‌گر یک تصحیح محلی بر روی باقیمانده‌ی ورودی انجام می‌دهد.

- در این حالت تجزیه‌گر به دنبال قاعده‌ای می‌گردد که سمت راست آن بیشترین شباهت را به دستگیره داشته باشد.
- با توجه به اختلاف دستگیره و سمت راست قاعده‌ی پیدا شده، پیام خطای مناسبی چاپ می‌شود و عمل کاهش انجام شده، تجزیه ادامه می‌یابد.

مثال

برای روشن شدن چگونگی گذر از خطا در روش‌های تقدم در حالت عدم تطابق با سمت راست قواعد، موارد زیر را بررسی می‌کنیم:

- دستگیره: $aEbc$
- قاعده‌ی یافت شده: $E \rightarrow aEc$
- **خطا:** b اضافی در ورودی

- دستگیره: $abEc$
- قاعده‌ی یافت شده: $E \rightarrow abEdc$
- **خطا:** d گم‌شده در ورودی

- دستگیره: abc
- قاعده‌ی یافت شده: $E \rightarrow aEbc$
- **خطا:** ساختار نحوی E مورد انتظار است (مثلاً یک عبارت ریاضی جا افتاده است).
  در این حالت پیام خطا نباید به طور مستقیم به ناپایانه‌ی E اشاره کند، زیرا کاربر در مورد ناپایانه‌های گرامر چیزی نمی‌داند.

۱۰-۵ خطاهای نحوی در روش‌های LR(1)

منشأ خطاهای نحوی در روش‌های LR عبارت است از:

- مراجعه به یک خانه‌ی خالی از بخش ACTION جدول تجزیه

۱۰-۵-۱ گذر از خطا در روش‌های LR: استراتژی حالت وحشت

تجزیه‌گر در هنگام کشف خطا، نمادهای ورودی را تا رسیدن به یک توکن مناسب دور می‌ریزد.

در صورت برخورد با یک خطای نحوی:

- تجزیه‌گر آن‌قدر در پشته پایین می‌رود تا به حالتی مانند s برسد که حداقل برای یک ناپایانه مانند A در خانه‌ی $\text{GOTO}[s, A]$ مقداری مانند n وجود داشته باشد.
- در این صورت A و n را به ترتیب به بالای پشته اضافه می‌کند.
- به علاوه، از توکن‌های ورودی نیز آن‌قدر حذف می‌کند تا به یکی از عناصر $Follow(A)$ برسد.

پیاده‌سازی این روش ساده است، اما ممکن است بخش زیادی از ورودی حذف شود.

۱۰-۵-۲ گذر از خطا در روش‌های LR: استراتژی سطح عبارت

تجزیه‌گر یک تصحیح محلی بر روی باقیمانده‌ی ورودی انجام می‌دهد.

- هیچ بخشی از ورودی بدون بررسی حذف نمی‌شود.
- با استراتژی سطح عبارت، خطا در همان محل وقوع خطا تصحیح می‌شود.

در خانه‌های خالی بخش ACTION جدول تجزیه اشاره‌گرهایی به روال‌های مناسب گذر از خطا قرار داده می‌شود. چنانچه به یکی از این خانه‌ها مراجعه شد، روال متناظر اجرا می‌شود.

مثال

برای توضیح چگونگی گذر از خطا در روش‌های LR با استراتژی سطح عبارت، گرامر و جدول تجزیه‌ی زیر را از فصل قبل در نظر می‌گیریم:

$$E \to E + E \mid E * E \mid (E) \mid \mathbf{id} \quad (1, 2, 3, 4)$$

	ACTION						GOTO
STATE	**id**	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	

خانه‌های خالی حالت‌هایی که عمل کاهش را تنها با یک قاعده‌ی خاص بر روی برخی از نمادهای ورودی فراخوانی می‌کنند، با عمل کاهش با همان قاعده پر می‌شوند. این تغییر باعث می‌شود کشف خطا پس از انجام یک یا چند کاهش به تعویق بیفتد، ولی خطا پیش از انجام یک عمل شیفت کشف شود.

سایر عناصر خالی جدول تجزیه با اشاره‌گرهایی به روال‌های گذر از خطا پر می‌شود.

$$E \to E + E \mid E * E \mid (E) \mid \mathbf{id} \quad (1, 2, 3, 4)$$

	Action						Goto
State	**id**	+	*	(	)	$	*E*
0	s3	e_1	e_1	s2	e_2	e_1	1
1	e_3	s4	s5	e_3	e_2	acc	
2	s3	e_1	e_1	s2	e_2	e_1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e_1	e_1	s2	e_2	e_1	7
5	s3	e_1	e_1	s2	e_2	e_1	8
6	e_3	s4	s5	e_3	s9	e_4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	

معنای اشاره‌گرها را مطابق با جدول زیر در نظر می‌گیریم:

e_1	در حالت‌های ۰، ۲، ۴، ۵ در صورت انتظار داشتن **id** یا)، اما یافتن یک عملگر یا انتهای ورودی
e_2	در حالت‌های ۰، ۱، ۲، ۴، ۵ در هنگام مشاهده‌ی یک پرانتز بسته اضافی
e_3	در حالت‌های ۱ و ۶ در صورت انتظار داشتن یک عملگر، اما یافتن **id** یا)
e_4	در حالت ۶ به انتهای ورودی برسیم، در صورت انتظار داشتن یک عملگر یا)

کنش مناسب و پیغام نمایش داده شده به کاربر برای هر یک از اشاره‌گرهای خطا به صورت زیر در نظر گرفته می‌شود:

e_1	قرار دادن **id** و سپس حالت $\text{GOTO}(0; 2; 4; 5, \textbf{id}) = 3$ در پشته	پیام خطا: missing operand
e_2	حذف (از ورودی	پیام خطا: unbalanced right parenthesis
e_3	قرار دادن + و سپس حالت $\text{GOTO}(1; 6, +) = 4$ در پشته	پیام خطا: missing operator
e_4	قرار دادن (و سپس حالت $\text{GOTO}(6, () = 9$ در پشته	پیام خطا: missing right parenthesis

حال می‌توان یک رشته‌ی دارای خطا را به صورت زیر تجزیه کرد:

Stack	Input	Action	Message
0	**id**+)$		
0 **id** 3	+)$		
0 *E* 1	+)$		
0 *E* 1 + 4	)$	error e_2: remove (	unbalanced right parenthesis
0 *E* 1 + 4	$	error e_1: push **id** 3 on stack	missing operand
0 *E* 1 + 4 **id** 3	$		
0 *E* 1 + 4 *E* 7	$		
0 *E* 1	$	finish	

◄ تمرین

۱. با استفاده از ترکیب استراتژی‌های مختلف برخورد با خطا، برای گرامر زیر یک تجزیه‌گر SLR برخورد کننده با خطا بسازید. تداخل‌های جدول تجزیه را نیز به طور معمول رفع کنید.

$$
\begin{array}{rcl}
stmt & \rightarrow & \textbf{if } e \textbf{ then } stmt \\
 & | & \textbf{if } e \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{while } e \textbf{ do } stmt \\
 & | & \textbf{begin } list \textbf{ end} \\
 & | & s \\
list & \rightarrow & list \ ; \ stmt \\
 & | & stmt
\end{array}
$$

۲. بازگشتی از چپ را از گرامر فوق رفع کنید و با استفاده از ترکیب استراتژیهای مختلف برخورد با خطا، برای آن یک تجزیه‌گر LL(1) برخورد کننده با خطا بسازید. تداخل‌های جدول تجزیه را نیز به طور معمول رفع کنید.

۳. رفتار تجزیه‌گرهای فوق را بر روی ورودیهای دارای خطای زیر نشان دهید:

آ) $\textbf{if } e \textbf{ then } s \ ; \ \textbf{if } e \textbf{ then } s \textbf{ end}$

ب) $\textbf{while } e \textbf{ do begin } s \ ; \ \textbf{if } e \textbf{ then } s \ ; \ \textbf{end}$