دانشکده‌ی مهندسی کامپیوتر

طراحی کامپایلر سارا آذرنوش ۹۸۱۷۰۶۶۸

تمرین سری سوم

۱

برای تعیین وضعیت Symbol Table و Scope Stack در هنگام اجرای خطوط ۷ و ۱۱، باید جریان اجرای برنامه را مدل کنیم. در اینجا فرض می‌کنیم اجرای برنامه از خط ۱ شروع می‌شود.
ابتدا، یک جدول نمادها Table Symbol و یک پشته محدوده Stack Scope ایجاد می‌کنیم.
Symbol Table:

Scope	Type	Name
A	integer	i
A	integer	j
A	procedure	B
B	real	a
B	procedure	C
C	integer	k
C	real	b
A	procedure	D
D	integer	l
D	procedure	E
E	real	d

Scope Stack :
Global, A

حالت اجرا: ۱. خط ۷: برای اجرای این خط، ابتدا باید متغیرهای مورد نیاز و محدوده آنها در جدول نمادها و پشته محدوده چک شود. بررسی نشان می‌دهد که متغیر a در محدوده B و متغیر c در محدوده C وجود دارند. بنابراین، اجرای این خط بدون مشکل ادامه می‌یابد.
Scope Stack:
Global, A, B, C

- Scope ‘C’ : ‘k’ , ’c’
- Scope ‘B’ : ‘a’ , ‘b’
- Scope ‘A’: ‘i’, ‘j’

۲. خط ۱۱: برای اجرای این خط، نیاز به تغییر مقدار متغیر l در محدوده D است. به ترتیب، باید متغیرها و محدوده‌ها در جدول نمادها و پشته محدوده چک شوند. متغیرهای i و j در محدوده A و متغیر l در محدوده D وجود دارند. بنابراین، اجرای این خط بدون مشکل ادامه می‌یابد.

بنابراین، وضعیت Symbol Table و Scope Stack بعد از اجرای خطوط ۷ و ۱۱ به شرح زیر است:

Symbol Table:

Scope	Type	Name
A	integer	i
A	integer	j
A	procedure	B
B	real	a
B	procedure	C
C	integer	k
C	real	b
A	procedure	D
D	integer	l
D	procedure	E
E	real	d

۲

در قطعه کد زیر، چهار خطا رخ می‌دهد:

۱. خطا ۱: خطا در خط ۸ - نوع خطا: خطای دسترسی به متغیر در این خط، متغیر j در محدوده B تعریف شده است، اما در دستور k =: i + j، متغیر i در محدوده A تعریف شده ولی متغیر j در محدوده B تعریف شده است. بنابراین، دسترسی به متغیر j از محدوده نادرست است. یک خطای dynamic است.

۲. خطا ۲: خطا در خط ۱۰ - نوع خطا: تکرار نام متغیر در خط ۱۰، متغیر l به عنوان ورودی تابع C تعریف شده است، اما در خط ۱۰ نیز متغیری با نام l تعریف شده است. این تکرار نام متغیر باعث ایجاد خطای تعریف متغیر می‌شود. یک خطای static است.

۳. خطا ۳: خطا در خط ۱۳ - نوع خطا: خطای دسترسی به آرایه در خط ۱۳، متغیر j به عنوان اندیس آرایه a استفاده شده است، اما متغیر j هنوز مقداردهی نشده است. این باعث ایجاد خطای دسترسی به آرایه می‌شود. یک خطای static است ولی اگر آن را یک runtime check در نظر بگیریم، dynamic خواهد بود

۴. خطا ۴: خطا در خط ۱۴ - نوع خطا: خطای تقسیم بر صفر در خط ۱۴، دستور k =: ۳/i وجود دارد. اما مقدار متغیر i برابر با صفر است. بنابراین، تقسیم بر صفر انجام می‌شود که باعث خطای تقسیم بر صفر می‌شود. یک خطای dynamic است.

۳

برای گرامر زیر:

1. $S \rightarrow G\ x$
2. $S \rightarrow y\ G\ z$
3. $S \rightarrow H\ z$
4. $S \rightarrow y\ H\ x$
5. $G \rightarrow w$
6. $H \rightarrow w$

۱. جدول LR:

S	H	G	$	Z	W	y	x	
	3	1			S5	S2		0

			R1	R1	R1		R1	1
		4						2
			R3	R3	R3		R3	3
							S6	4
			R5	R5	R5		R5	5
				S7	S5			6
			R2	R2	R2		R2	7

۲. جدول :LALR

S	H	G	$	Z	W	y	x	
	3	1			S5	S2		0
			R1	R1	R1		R1	1
		4						2
			R3	R3	R3		R3	3
							S6	4
			R5	R5	R5		R5	5
				S7	S5			6
			R2	R2	R2		R2	7

۳. این گرامر از نوع LR نیست زیرا دارای تعاریف مبهم است. برای مثال، ،LR برای ورودی ،"yz" دو مسیر باز است: ، S -> yGz S -> Hz. این باعث می‌شود تجزیه‌گر SLR نتواند رشته "yz" را به درستی تجزیه کند.
همچنین، این گرامر نیز از نوع LALR نیست زیرا دارای تعاریف مبهم است. برای مثال، ،LALR برای ورودی ،"ywx" دو مسیر باز است: ywx -> yGz -> S و S -> yHx. این باعث می‌شود تجزیه‌گر LALR نتواند رشته "ywx" را به درستی تجزیه کند.
بنابراین، این گرامر هیچکدام از نوع LR و LALR نیست و برای بعضی از رشته‌ها ممکن است تجزیه‌گر نتواند آنها را به درستی تجزیه کند.

۴

۱. یک گرامر SLR نیست اگر یک conflict از نوع reduce/shift یا reduce/reduce در حین ساختن parsing table داشته باشیم.

State 5: A -> B.y
State 9: B -> yB.x

در ۵ state اگر lookahead ما x' 'باشد، امکان shift به استیت ۹ ایجاد میشود یا reduce با توجه به A -> B. این منجر به یک conflict از نوع reduce/shift است چون follow(B) شامل x' 'است و این وضعیت را مبهم میسازد.

۲.
follow(S) = {$}
follow(A) = {$}
follow(B) = {y,x}

۳.
state 2 (A -> xx.)
LA: follow(A) = {$}
state 4 (B -> x.)
LA: follow(B) = {y, x}
state 6 (A -> B.y)
LA: follow(A) = {$}
state 7 (B -> yBx.)
LA: follow(B) = {x, y}

۴. باید بررسی کنیم که آیا ادغام حالت ها با core یکسان منجر به conflict در parsing table گرامر مربوطه می شود یا خیر.
استیت های ۴ و ۷:
هر دو B -> x core دارند.

lookahead 4 = {x,y}
lookahead 7 = {x,y}

هیچ conflict ای در عملیات ادغام وجود ندارد.
پس گرامر LALR است.

initial item I_0:

$S' \to .S, \{\$\}$
$S \to .A, \{\$\}$
$A \to .By, \{\$\}$
$A \to .xx, \{\$\}$
$B \to .x, \{y, x\}$
$B \to .yBx, \{y, x\}$

Closure of I_0:

$S' \to .S, \$$
$S \to .A, \$$
$A \to .By, \$$
$A \to .xx, \$$
$B \to .x, y/\$$
$B \to .yBx, y/\$$

goto S:

I_1:
$S' \to S., \$$

goto A:

I_2:
$S \to A., \$$

goto B:

I_3:
$A \to B.y, \$$
$B \to .x, y/\$$
$B \to .yBx, y/\$$

goto x:

I_4:
$A \to x.x, \$$
$B \to x., y/\$$

goto y:

I_5:
$A \to By., \$$

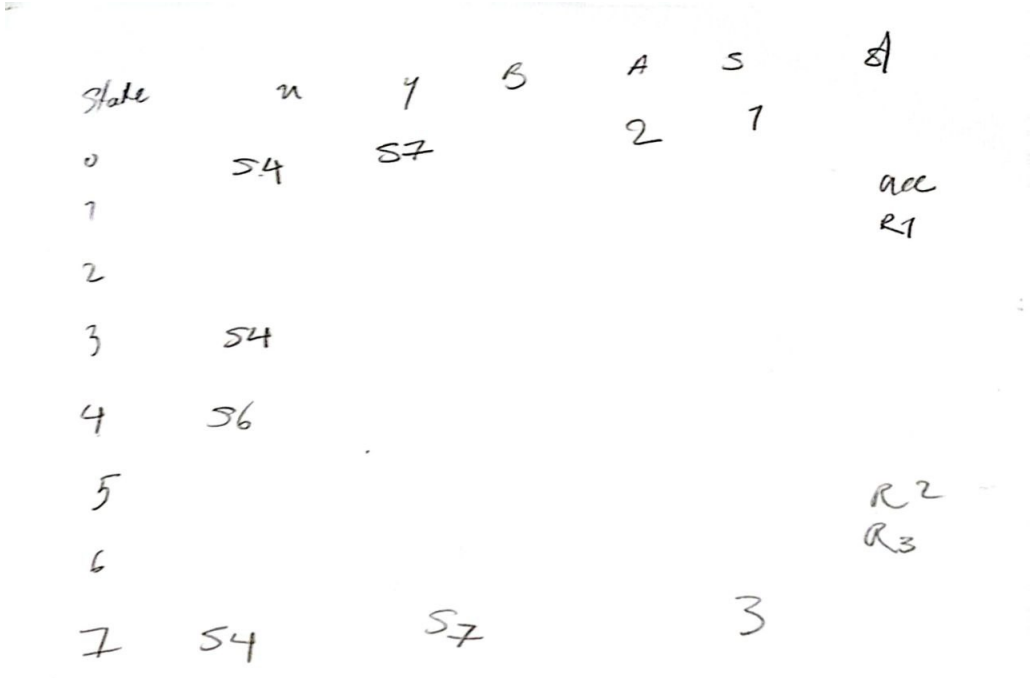
goto xx:

I_6
$A \to xx., \$$

goto yB:

I_7
$B \to yB.x, y/\$$
$B \to .x, y/\$$
$B \to .yBx, y/\$$

.٥



در استیت ۳ یک conlict داریم اگر x بتواند باعث shift به استیت ۴ بشود و reduce کند B -> x.

اگر y بتواند هم باعث shift به استیت ۷ شود و هم reduce کند B -> yBx

conflict از نوع reduce/shift داریم که باعث به وحود آمدن ابهام در parsing table میشوند. بنابراین ، گرامر ما LR نیست

۵

۱. عملیات jp به یک مقصد نامشخص انجام میشود. در linked list مبدا و Lable مربوطه قرار داده میشود.

۲. stmt تبدیل شده است و مقصد L۳ مشخص میشود.

۳. عملیات jp مقصد مشخصی ندارند بنابراین باید در linked list هم مبدا و لیبل مربوطه را قرار دهیم.

۴. عملیات jp مقصد مشخصی ندارد و در linked list مربوط به L۵ که از پیش موجود است. ۳ به مبداها اضافه میشود.

۵. stmt تبدیل شده است و مقصد L۱ مشخص میشود.

۶. stmt تبدیل شده است و مقصد L۵ مشخص میشود.

۷. p به L۳ با مقصد مشخص ۱ طبق linked list انجام میشود

۸. p به L۵ با مقصد مشخص ۵ طبق linked list انجام شده است.

۹. :۸ در جدول jp مقصد مشخص ندارد و در linked list مربوط به L۲ که از قبل مشخص است، ۸ به مبداها اضافه میشود.

۱۰. stmt۴ تبدیل شده و در linked list مقصد L۴ را مشخص میکنیم.

۱۱. jp به L۴ با مقصد مشخص طبق linked list انجام شده به جدول اضافه میشود.

۱۲. p به L۴ با مقصد مشخص ۹ طبق linked list انجام شده و به جدول اضافه میشود.

۱۳. stmt۵ تبدیل میشود و مقصد L۲ مشخص شده است و در linked list و جدول آپدیت می شود.

در اجرا از خط ۰ به خط ۵ جامپ میکنیم و stmt۳ اجرا میشود.
با خواندن خط ۶ به خط ۱ جامپ می کنیم و بنابراین stmt۱ اجرا میشود.
به خط ۲ میرویم و به خط ۱۲ جامپ میکنیم. stmt۵ اجرا شده و در نهایت برنامه پایان می یابد.