"به نام خدا"

نظریه زبان ها و ماشین ها - تمرین کوتاه کلاسی ۹ — مصطفی نصلی ثمری — ۹۸۲۲۸۰۲ — ۱۴۰۰/۲/۱۳

۱. اثبات مبهم بودن گرامر if - statement و طریقه رفع ابهام آن

اگر قوانین این گرامر را بنویسیم داریم:

statement → if BE then st

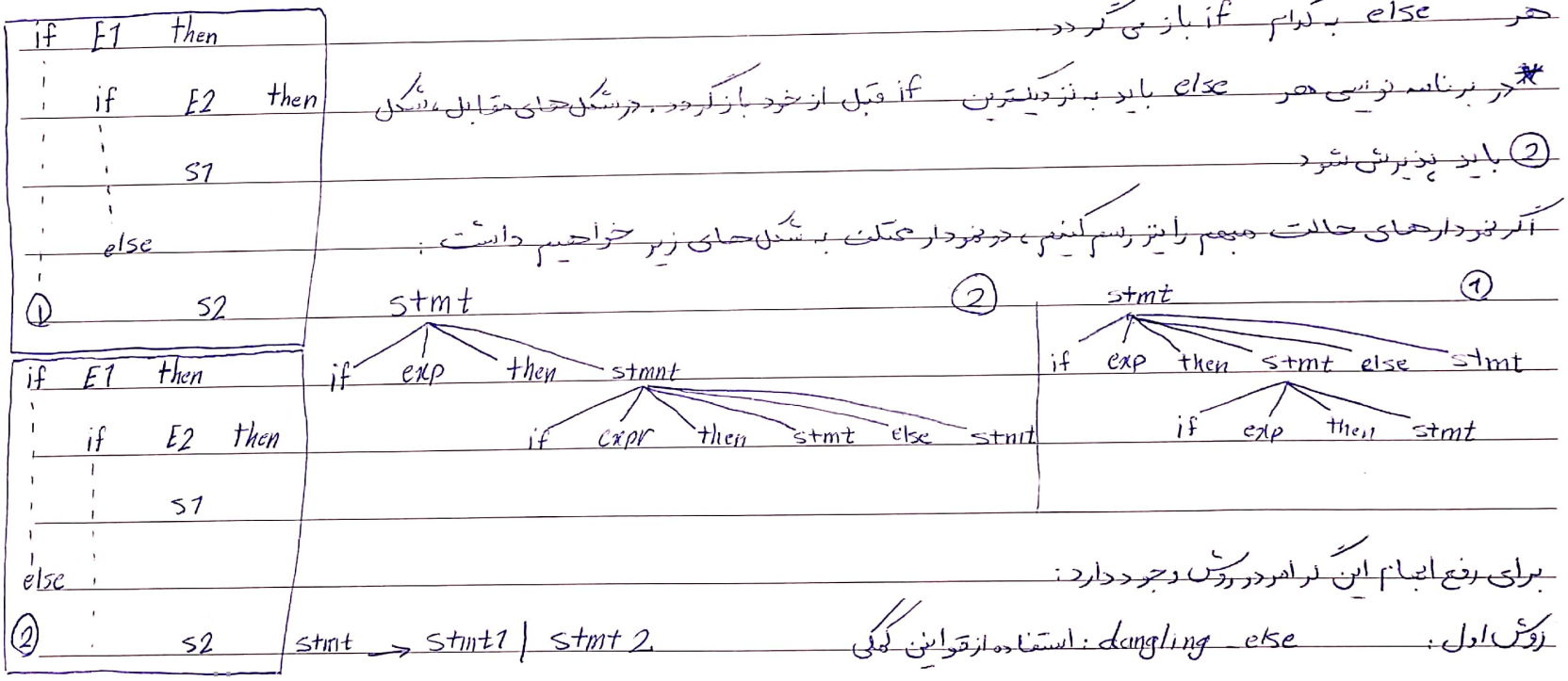
| if BE then Statement else statement

BE → be st → Statement

این گرامر در هنگامی که چندین شرط تو در تو و به صورت پیاپی بیایند مبهم می شود؛ بدین صورت که معلوم نمی شود هر else به کدام if باز می گردد.

*در برنامه نویسی هر else باید به نزدیکترین if قبل از خود باز گردد. در شکل های مقابل به شکل ② باید پذیرش شود

اگر نمودارهای حالت مبهم را به ترسیم کنیم، درخت مختلف به شکل های زیر خواهیم داشت:



برای رفع ابهام این گرامر دو روش وجود دارد:

روش اول: dangling_else: استفاده از قوانین کلی

stmt → stmt1 | stmt2

stmt1 → if exp then stmt1 else stmt1 | other

stmt2 → if exp then stmt1 else stmt2 | if exp then stmt

Subject:
Year. Month. Day.

روش دوم: جهت‌های A و B

A : جهت نشده
B : جهت شده

Stmt → A | B

A → if BE then A else A | if BE then ST else ST

B → if BE then ST | if BE then A else B

ST → Stmt

BE → be

2- شرح الگوریتم cyk

این الگوریتم، یک الگوریتم برنامه ریزی پویا برای به دست آوردن تجزیه گرامری جملات با استفاده از گرامر مستقل از متن است، همچنین این الگوریتم از تجزیه از پائین به بالا استفاده می‌کند. نام این الگوریتم از حرف اول سه طراح آن گرفته شده است.

J. Cocke
D. Younger
T. Kasami

این الگوریتم در صورتی کار می‌کند که گرامر در فرم نرمال چامسکی باشد.

فرم نرمال چامسکی، همه قواعد گرامر آن به یکی از دو صورت یا ترکیبی از هر دو باشد:

$$\begin{cases} A \to BC \\ A \to a \end{cases} \qquad A, B, C \in V \ , \ a \in T$$

شرح الگوریتم:

در ابتدا یک جدول برنامه ریزی پویا با سایز $n \times n$ تشکیل می‌دهیم.

سپس با یک حلقه‌ی به تعداد n، برای هر متغیر A بررسی می‌کنیم که اگر $A \to b$ یک قانون باشد $b = w_i$ برای هر i، ما آن را به صورت جفت (i, i) در جدول قرار می‌دهیم.

در مرحله بعد، با یک حلقه‌ی از 2 تا n است، و یک حلقه با کم کردن حلقه‌ی l که از 1 تا $n-l+1$ است، حلقه k را به صورت شروع از i تا $i+l-2$ می‌پیماییم و برای هر قانون $A \to BC$ بررسی می‌کنیم که اگر جفت (i, k) شامل B و جفت $(k+1, j)$ شامل C بودند، جفت (i, j) را در جدول قرار می‌دهیم.

در هر حلقه آخر اگر S از a_1 تا a_n وجود داشت رشته را می‌پذیریم و در غیر این صورت آن را نفی می‌پذیریم.

با توجه به توضیحات بالا پیچیدگی زمانی این الگوریتم $O(n^3)$ و پیچیدگی فضایی آن $O(n^2)$ می‌باشد.

مثال: برای عبارت baaba داریم،

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

	b	a	a	b	a
b	{B}	{S,A}	∅	∅	{S,A,C}
a		{A,C}	{B}	{B}	{S,A,C}
a			{A,C}	{S,C}	{B}
b				{B}	{S,A}
a					{A,C}

Subject:

Year. Month. Day.

به عبارت دیگر فرض کنید گرامر $G=\{V,T,S,P\}$ در فرم چامسکی بوده و رشته ای را داریم

$W = a_i \dots a_j$

زیرمجموعه‌ها $V_{ij} = \{A \in V \mid A \Rightarrow W_{ij}\}$

مثال: آیا رشته $W=aabbb$ در زبان گرامر زیر وجود دارد یا خیر

$$\begin{cases} S \to AB \\ A \to BB \mid a \\ B \to AB \mid b \end{cases}$$

$w_{1,1}=a$, $V_{1,1}=\{A\}$ $\quad V_{1,2}=\{A \mid A \to BC;\ B \in V_{1,1},\ C \in V_{2,2}\}$

$w_{2,2}=a$, $V_{2,2}=\{A\}$ مجموعه‌ای که سمت راست آن‌ها حاوی AA باشد

$w_{3,3}=b$, $V_{3,3}=\{B\}$ چون همچین قوانینی نداریم پس $=\emptyset$

$w_{4,4}=b$, $V_{4,4}=\{B\}$ $\quad V_{2,3}=\{A \mid A \to BC;\ B \in V_{2,2},\ C \in V_{3,3}\}$

$w_{5,5}=b$, $V_{5,5}=\{B\}$ مجموعه قوانینی که سمت راست آن‌ها شامل AB باشد برابر این.

$V_{3,4}=\{A\}$ $\quad V_{4,5}=\{A\}$ $\quad V_{23}=\{S,B\}$

$V_{1,3}=\{S,B\}$ $\quad V_{2,4}=\{A\}$ $\quad V_{3,5}=\{S,B\}$

$V_{2,5}=\{S,B\}$ $\quad V_{1,4}=\{A\}$ $\quad V_{1,5}=\{S,B\}$