مروری بر مبانی نظریه محاسبه

امیرحسین رجبی کوروش مسلمی

۱ گرامرهای مستقل از متن

گرامرها [1] یک مدل محاسباتی هستند و برخلاف اتوماتاهای متناهی که کلمات یک زبان را با قرارگرفتن در استیت اکسپت پذیرش می‌کنند، کلمات زبانی را به کمک تعدادی قاعده [2] تولید می‌کنند.

تعریف ۱. گرامر مستقل از متن یک چهارتایی $G = (V, \Sigma, S, P)$ است که به ترتیب V مجموعه متناهی متغیرهای گرامر، Σ مجموعه متناهی حروف الفبا، S متغیر شروع گرامر ($S \in V$) و P مجموعه متناهی از قواعد که به فرم $A \to \alpha$ هستند که $A \in V$ و $\alpha \in (V \cup \Sigma)^*$.

مثال: گرامر زیر زبان $L = \{a^n b^n \mid n \in \mathbb{N}\}$ را تولید می‌کند. (زبان $AnBn$)

$$G = (\{S\}, \{a, b\}, S, \{S \to aSb, S \to \Lambda\})$$

$$S \to aSb$$

$$S \to \Lambda$$

یا به طور خلاصه $S \to aSb \mid \Lambda$. واضح است که $\Lambda \in (V \cup \Sigma)^*$ و به معنای قابلیت جایگزینی S با رشته تهی است؛ در نتیجه تولید یا پروداکشن می‌تواند خاتمه یابد. همچنین به حروف Σ ترمینال نیز گفته می‌شود.

برای نشان دادن نحوه تولید یک کلمه به کمک زبان از نماد های $\Rightarrow$، $\Rightarrow^*$ و $\Rightarrow^n$ استفاده می‌شود.

مثال: گرامر G که در بالا توصیف شد را در نظر بگیرید، رشته $aabb$ به صورت زیر از قواعد گرامر مشتق [3] می‌شود:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aa\Lambda bb \Rightarrow aabb$$

یا به طور خلاصه $S \Rightarrow_G^* aabb$ نوشته می‌شود که یعنی با تعدادی مرحله به رشته مذکور می‌رسیم. همچنین با زیرنویس G مشخص می‌کنیم که از قواعد گرامر G استفاده شده‌است.

تعریف ۲. اگر $G = (V, \Sigma, S, P)$ یک گرامر مستقل از متن باشد، منظور از زبان تولید شده توسط G،

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$$

است. زبانی مستقل از متن است اگر گرامر مستقل از متنی وجود داشته باشد که آن را تولید کند.

مثال: زبان حاوی کلمات پالیندروم، زبانی مستقل از متن است. (زبان Pal)

[1] Context Free Grammar
[2] Production Rule
[3] Derivation

گرامر مستقل از متن زیر را داریم: (با فرض $\Sigma = \{a, b\}$)

$$S \rightarrow aSa \mid bSb \mid \Lambda$$

مثال: زبان حاوی کلماتی که تعداد a در آنها با تعداد b برابر است، زبانی مستقل از متن است. (زبان $AEqB$)

گرامر مستقل از متن زیر را داریم: (با فرض $\Sigma = \{a, b\}$)

$$S \rightarrow aSbS \mid bSaS \mid \Lambda$$

اثبات. باید نشان دهیم هر کلمه مثل x که $n_a(x) = n_b(x)$ توسط این قواعد تولید می‌شود و همچنین باید ثابت کنیم هر کلمه‌ای که توسط این قواعد تولید می‌شود چنین ویژگی را دارد. گزاره دوم واضح است. گزاره اول را ثابت می‌کنیم. به استقرای قوی حکم را روی $|x|$ ثابت می‌کنیم. پایه استقرا $x = \Lambda$ است که توسط گرامر تولید می‌شود. اکنون فرض کنید حکم برای همه رشته‌های به طول زوج کوچکتر از $|x|$ برقرار باشد، حکم را برای x نشان می‌دهیم. تابع $f : \Sigma^* \rightarrow \mathbb{Z}$ را با ضابطه $f(x) = n_a(x) - n_b(x)$ تعریف می‌کنیم. همچنین فرض می‌کنیم $x = x_۱x_۲ \dots x_n$ ($x_i \in \Sigma$). بدون کاسته شدن از کلیت مسئله فرض کنید $x_۱ = a$. اولین i که $f(x_۱x_۲ \dots x_i) = ۰$ را در نظر می‌گیریم. در این صورت نتیجه می‌شود $x_i = b$. پس $x_۲ \dots x_{i-۱} \in AEqB$ و همچنین $x_{i+۱} \dots x_n \in AEqB$. طبق فرض استقرا این دو کلمه به کمک قواعد G از متغیر S قابل تولیدند. با توجه به قاعده $S \rightarrow aSbS$ کلمه x نیز توسط گرامر قابل تولید است. $\square$

قضیه ۱. اگر $L_۱$ و $L_۲$ زبان‌هایی مستقل از متن باشند، زبان‌های $L_۱ \cup L_۲$، $L_۱L_۲$ و $L_۱^*$ نیز مستقل از متن هستند.

اثبات. فرض کنید $G_۱ = (V_۱, \Sigma, S_۱, P_۱)$ و $G_۲ = (V_۲, \Sigma, S_۲, P_۲)$ به ترتیب گرامرهای مستقل از متن برای زبان‌های $L_۱$ و $L_۲$ باشند.

اگر $H_۱ = (V_۱ \cup V_۲, \Sigma, S, P_۱ \cup P_۲ \cup \{S \rightarrow S_۱ \mid S_۲\})$، آنگاه $L(H_۱) = L_۱ \cup L_۲$.

اگر $H_۲ = (V_۱ \cup V_۲, \Sigma, S, P_۱ \cup P_۲ \cup \{S \rightarrow S_۱S_۲\})$، آنگاه $L(H_۲) = L_۱L_۲$.

اگر $H_۳ = (V_۱, \Sigma, S, P_۱ \cup P_۲ \cup \{S \rightarrow SS_۱ \mid \Lambda\})$، آنگاه $L(H_۳) = L_۱^*$. $\square$

۱.۱ گرامرهای منظم و خطی راست و چپ

تعریف ۳. گرامر مستقل از متن $G = (V, \Sigma, S, P)$ را گرامر خطی راست [4] می‌گوییم اگر همه قواعد آن به فرم $A \rightarrow \sigma B$ یا $A \rightarrow \Lambda$ باشد که در آن $A, B \in V$ و $\sigma \in \Sigma$.

تعریف ۴. گرامر مستقل از متن $G = (V, \Sigma, S, P)$ را گرامر خطی چپ [5] می‌گوییم اگر همه قواعد آن به فرم $A \rightarrow B\sigma$ یا $A \rightarrow \Lambda$ باشد که در آن $A, B \in V$ و $\sigma \in \Sigma$.

تعریف ۵. گرامر مستقل از متن $G = (V, \Sigma, S, P)$ را گرامر منظم [6] می‌گوییم اگر خطی راست یا خطی چپ باشد.

قضیه ۲. زبان L منظم است اگر و تنها اگر گرامر منظم G وجود داشته باشد که $L = L(G)$.

اثبات. (شهودی) اثبات معادل بودن گرامرهای خطی راست با اتوماتاهای متناهی ساده است. فرض کنید $M = (Q, \Sigma, q_۰, A, \delta)$ یک اتوماتای متناهی قطعی [7] باشد. گرامر خطی راست $G = (V, \Sigma, S, P)$ را این‌گونه می‌سازیم. قرار می‌دهیم $V = Q$ و $S = q_۰$. همچنین برای q که $\delta(p, \sigma) = q$

[4] Right Linear Grammars (RLG)
[5] Left Linear Grammars (LLG)
[6] Regular Grammars
[7] Deterministic Finite Automata (DFA)

$p,q \in Q$ و $\sigma \in \Sigma$ قاعده $p \to \sigma q$ نیز در P قرار خواهد داشت. ضمنا اگر $r \in A$ آنگاه قاعده $r \to \Lambda$ نیز در P را دارد. به سادگی می‌توان دید که $L(G) = L(M)$. از طرفی اگر $G = (V, \Sigma, S, P)$ گرامر خطی راست باشد، اتوماتای متناهی غیر قطعی [8] [9] $M = (Q, \Sigma, q_\circ, A, \delta)$ را این‌گونه می‌سازیم. قرار می‌دهیم $Q = V$ و $q_\circ = S$. همچنین تابع $\delta : Q \times \Sigma \to Q$ برای هر قاعده $X \to \sigma Y$ در P، این‌گونه تعریف می‌کنیم $\delta(X, \sigma) = Y$ و برای هر قاعده به فرم $X \to \Lambda$، X در A خواهد بود. دوباره به سادگی می‌توان دید که $L(M) = L(G)$.

برای معادل بودن گرامرهای خطی چپ کار کمی دشوارتر است. اگر هر قاعده در یک گرامر خطی چپ مثل $X \to Y\sigma$ با قاعده $X \to \sigma Y$ جایگزین شود، زبان تولید شده توسط این گرامر شامل عکس تمام رشته‌های زبان گرامر اولیه خواهد بود. (نه فقط شامل آنها بلکه دقیقا همان‌ها می‌شود.) از طرفی در صورت داشتن یک **اتوماتای متناهی قطعی** می‌توان یک اتوماتای متناهی دیگر ساخت که دقیقا عکس کلمات پذیرفته شده توسط اتوماتای نخست را پذیرش کند. همچنین هر اتوماتای غیرقطعی معادل یک اتوماتای قطعی است. پس به کمک این دو گزاره، فرآیند زیر برای اثبات قضیه طی می‌شود:

اتوماتای پذیرنده معکوس کلمات $\longleftrightarrow$ اتوماتای قطعی $\longleftrightarrow$ اتوماتای غیرقطعی $\longleftrightarrow$ گرامر خطی راست $\longleftrightarrow$ برعکس کردن قواعد $\longleftrightarrow$ گرامر خطی چپ

$\square$

پس بنابر قضیه بالا **گرامرهای منظم** قدرت محاسباتی بیشتری از اتوماتاهای متناهی ندارند.

۲.۱ ابهام و درخت اشتقاق

مثال: گرامر $G = (\{S\}, \{a, +, *\}, S, P)$ با قواعد زیر مبهم است:

$$S \to S + S$$
$$S \to S * S$$
$$S \to a$$

دو درخت اشتقاق [10] برای کلمه $a + a * a$ در زیر کشیده شده‌اند:

پس درخت اشتقاق به صورت تصویری نشان می‌دهد برای تولید یک کلمه چه قواعدی استفاده شده‌اند.

تعریف ۶. گرامر مستقل از متن G مبهم است اگر کلمه $x \in L(G)$ موجود باشد به طوریکه x دارای حداقل دو درخت اشتقاق باشد.

در نتیجه گرامر فوق مبهم است. رفع ابهام نیازمند کمی خلاقیت است. در اینجا می‌توان گرامر زیر را ارائه کرد:

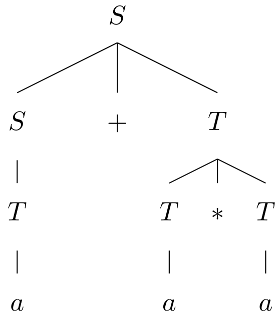
$$S \to S + T \mid T$$
$$T \to T * T \mid a$$

[8] چرا اتوماتای غیر قطعی تعریف می‌کنیم؟ چرا مستقیم سراغ قطعی نمی‌رویم؟

[9] Nondeterministic Finite Automata

[10] Derivation tree

با درخت اشتقاق زیر برای کلمه $a+a*a$:



۳.۱ فرم‌های نرمال

به مسئله تصمیم در گرامرهای مستقل از متن فکر کنید. یعنی با داشتن پروداکشن‌های گرامر G و رشته x مشخص کنیم x توسط قواعد G تولید می‌شود یا خیر. دانستن نکاتی درباره فرم پروداکشن‌ها به ما در پاسخ به این سوال کمک می‌کند. در مراحل اشتقاق [۱۱] یک رشته به کمک قواعد گرامر تعداد ترمینال‌های رشته (t) کاهش نمی‌یابند. اگر تصور کنیم گرامر هیچ لاندا-قاعده‌ای [۱۲] نداشته باشد، (یعنی هیچ قاعده‌ای به فرم $X \to \Lambda$) در این صورت طول رشته (l) نیز در مراحل اشتقاق کاهش نمی‌یابد. (در تعبیر درخت اشتقاق، تعداد رئوس در هر سطح درخت کاهش نمی‌یابد.) همچنین اگر بدانیم گرامر هیچ یونیت-پروداکشنی [۱۳] (یعنی هیچ قاعده‌ای به فرم $X \to Y$ که Y متغیر باشد.) نیز نداشته باشد، می‌توانیم بگوییم تعداد رئوس در هر سطح درخت اشتقاق اکیدا صعودی است. یعنی $l+t$ اکیدا صعودی است. اما وقتی رشته x تولید شده باشد داریم $l+t=|x|+|x|=2|x|$. از طرفی هنگام شروع تولید از متغیر S داریم $l+t=1+0=1$. در نتیجه تولید رشته x بیش از $2|x|-1$ مرحله نیاز ندارد. پس با بررسی همه درخت های اشتقاق با حداکثر عمق $2|x|-1$ می‌توانیم درباره وجود x در $L(G)$ تصمیم بگیریم.

در ادامه نشان می‌دهیم هر گرامر را می‌توان به گرامری تبدیل کرد که هیچ لاندا-قاعده و هیچ یونیت پروداکشنی نداشته باشد. سپس پا را فرا تر گذاشته و نشان می‌دهیم می‌توان قاعده‌ها را محدود به قاعده‌هایی به فرم $X \to YZ$ کرد که به فرم نرمال چامسکی [۱۴] معروف است. در نهایت به معرفی الگوریتم CYK [۱۵] می‌پردازیم که در مرتبه چندجمله‌ای عضویت رشته دلخواه x در $L(G)$ را بررسی می‌کند.

قضیه ۳. برای هر گرامر G، گرامر G' وجود دارد که هیچ لاندا قاعده‌ای نداشته باشد و همچنین $L(G')=L(G)-\{\Lambda\}$.

اثبات. مراجعه شود به صفحه ۱۵۰ کتاب مارتین. □

قضیه ۴. برای هر گرامر بدون لاندا-قاعده G، گرامر G' وجود دارد که هیچ یونیت پروداکشنی نداشته باشد و همچنین $L(G')=L(G)$.

اثبات. مراجعه شود به صفحه ۱۵۲ کتاب مارتین. □

تعریف ۷. گرامر G به فرم نرمال چامسکی است اگر هر قاعده گرامر به فرم $A \to BC$ یا $A \to \sigma$ باشد که $\sigma \in \Sigma$.

قضیه ۵. برای هر گرامر مستقل از متن G، گرامر G' به فرم نرمال چامسکی وجود دارد که $L(G')=L(G)-\{\Lambda\}$.

اثبات. مراجعه شود به صفحه ۱۵۲ و ۱۵۳ کتاب مارتین. □

الگوریتم CYK برای بررسی وجود یک کلمه در زبان تولیدشده توسط یک گرامر مستقل از متن استفاده می‌شود. کاربرد آن در بخش تجزیه [۱۶] کامپایلرها است. این الگوریتم همچنین نحوه اشتقاق کلمه را نیز نشان می‌دهد یعنی اینکه در هر مرحله از کدام قاعده استفاده شود تا کلمه تولید شود. برای ورودی، این الگوریتم نیاز دارد تا گرامر ورودی به فرم نرمال چامسکی باشد.

[11] Derivation steps
[12] Λ-production
[13] Unit production
[14] Chomsky normal form
[15] Cocke–Younger–Kasami algorithm
[16] Parser

مثال: گرامر زبان $AEqB$ را در نظر بگیرید:

$$S \to aSbS \mid bSaS \mid \Lambda$$

ابتدا لاندا-قاعده‌های آن را حذف می‌کنیم:

$$S \to abS \mid aSbS \mid aSb \mid ab$$
$$S \to baS \mid bSaS \mid bSa \mid ba$$

این گرامر به طور طبیعی شامل هیچ یونیت-پروداکشنی نیست، بنابراین می توانیم مراحل تبدیل به فرم چامسکی را آغاز کنیم. برای این کار ابتدا حروف ترمینال a و b را در قاعده‌هایی که سمت راست آن‌ها ترمینال نیست، به ترتیب با A و B تعویض می‌کنیم. بنابراین خواهیم داشت:

$$S \to ABS \mid ASBS \mid ASB \mid AB$$
$$S \to BAS \mid BSAS \mid BSA \mid BA$$
$$A \to a$$
$$B \to b$$

گرامر فوق طبق تعریف ۷ در فرم نرمال چامسکی قرار ندارد بنابراین آن را به صورت زیر اصلاح می کنیم:

$$S \to AY \mid XY \mid YX \mid BX$$
$$S \to AB \mid BA \mid XB \mid YA$$
$$X \to AS$$
$$Y \to BS$$

اجرای الگوریتم CYK برای رشته $aababb$:[۱۷] [۱۸]

$\{A\}$	$\varnothing$	$\{X\}$	$\varnothing$	$\{X\}$	$\{S\}$
	$\{A\}$	$\{S\}$	$\{X\}$	$\{S\}$	$\varnothing$
		$\{B\}$	$\{S\}$	$\{Y\}$	$\varnothing$
			$\{A\}$	$\{S\}$	$\varnothing$
				$\{B\}$	$\varnothing$
					$\{B\}$

فرض کنید رشته مورد بررسی این الگوریتم $x = x_۱x_۲\ldots x_n$ باشد. در این‌صورت جدولی $n \times n$ تشکیل می‌شود و در درایه سطر iام و ستون jام آن، مجموعه متغیرهایی قرار می‌گیرند که می‌توانند زیررشته $x_i \ldots x_j$ را تولید کنند. این مجموعه را X_{ij} با می‌نامیم. به وضوح مقادیر زیر قطر اصلی جدول معنایی ندارند. مقادیر قطر اصلی را به کمک قاعده‌های $C \to \sigma$ پر می‌کنیم $(\sigma \in \Sigma)$؛ یعنی در صورتی که $x_i = \sigma$ و قاعده $C \to \sigma$ در گرامر باشد، آنگاه

[۱۷] با تشکر فراوان از کوروش مسلمی که در تکمیل بخش الگورتم CYK کمک کرد.

[۱۸] این الگوریتم مشابه الگوریتم ضرب زنجیر ماتریس‌ها (Chain Matrix Multiplication) است که در آن نحوه پرانتز گذاری برای ضرب n ماتریس $A_۱A_۲\ldots A_n$ با تعداد سطر و ستون‌های زیر

$$A_۱(r_۱, c_۱), A_۲(r_۲ = c_۱, c_۲), \ldots, A_n(r_n = c_{n-۱}, c_n)$$

مطلوب است که کل تعداد عملیات‌های ضرب کمینه شود.

$C \in X_{ii}$. سپس سراغ قطرهای بالاتر می‌رویم. این‌بار برای پرکردن مجموعه X_{ij}، باید زیررشته $x_i \dots x_j$ را به دو زیررشته $x_i \dots x_k$ و $x_{k+1} \dots x_j$ بشکنیم ($i \le k \le j - 1$) و در صورت وجود قاعده‌ای در گرامر چون $U \to VW$ که $VW \in \bigcup_{k=i}^{j-1} X_{ik}X_{k+1j}$ باشد، آنگاه $U \in X_{ij}$.

به عنوان مثال در اینجا قطر اصلی به کمک A و B پر شده‌اند چون تنها متغیرهایی هستند که تک کاراکتر تولید می‌کنند. سپس در قطر بالاتر فقط متغیر S قرار گرفته‌است چون $S \to AB \mid BA$ و از طرفی AA و BB در سمت راست هیچ متغیری نیستند. اکنون مجموعه X_{25} را در نظر بگیرید. بنابر جدول $X_{25} = \{S\}$. برای تحقیق این موضوع $X_{22}X_{35} = \{A\}.\{Y\}$، $X_{23}X_{45} = \{S\}.\{S\}$ و $X_{24}X_{55} = \{X\}.\{B\}$ باید بررسی شوند که AY و XB در سمت راست قواعد ما موجودند. در نهایت، درصورت وجود متغیر شروع گرامر (S) در مجموعه X_{1n} الگوریتم با خروجی YES پایان می‌یابد. در غیر این‌صورت خروجی NO خواهد بود. الگوریتم از مرتبه $\mathcal{O}(n^3|V|^2|P|)$ خواهد بود که P مجموعه پروداکشن‌های گرامر است و V مجموعه متغیرهای آن است. (چون P و V مستقل از n هستند الگوریتم از مرتبه $\mathcal{O}(n^3)$ است.) لازم به ذکر است که هنگام محاسبه X_{ij} شکستن رشته به دو زیررشته کافی است زیرا گرامر به فرم نرمال چامسکی است و قواعد آن دو متغیره هستند.

روش الگوریتمی تبدیل گرامر بدون لاندا–قاعده زبان $AEqB$ به فرم نرمال چامسکی به صورت زیر است:

$$S \to X_aX_bS \mid X_aSX_bS \mid X_aSX_b \mid X_aX_b$$
$$S \to X_bX_aS \mid X_bSX_aS \mid X_bSX_a \mid X_bX_a$$
$$X_a \to a$$
$$X_b \to b$$

و درنهایت:

$$S \to X_aY_1 \mid X_aY_2 \mid X_aY_3 \mid X_aX_b$$
$$S \to X_bY_4 \mid X_bY_5 \mid Y_2Y_5 \mid X_bX_a$$
$$Y_1 \to SX_b$$
$$Y_2 \to X_bS$$
$$Y_3 \to SY_2$$
$$Y_4 \to SX_a$$
$$Y_5 \to X_aS$$

اکنون گرامر در فرم نرمال چامسکی قرار دارد و می توان الگوریتم CYK را اجرا کرد. در جدول زیر اجرای این الگوریتم برای رشته $aababb$ نشان داده شده است:

$\{X_a\}$	$\varnothing$	$\{Y_5\}$	$\varnothing$	$\varnothing$	$\{S\}$
	$\{X_a\}$	$\{S\}$	$\{Y_4, Y_5\}$	$\{S\}$	$\{Y_1\}$
		$\{X_b\}$	$\{S\}$	$\{Y_1, Y_2\}$	$\varnothing$
			$\{X_a\}$	$\{S\}$	$\{Y_1\}$
				$\{X_b\}$	$\varnothing$
					$\{X_b\}$