درس مبانی نظریه محاسبه

جلسه چهاردهم

مسئله ابهام در گرامر، فرم نرمال چامسکی

Ambiguity in grammers and the Chomsky normal form

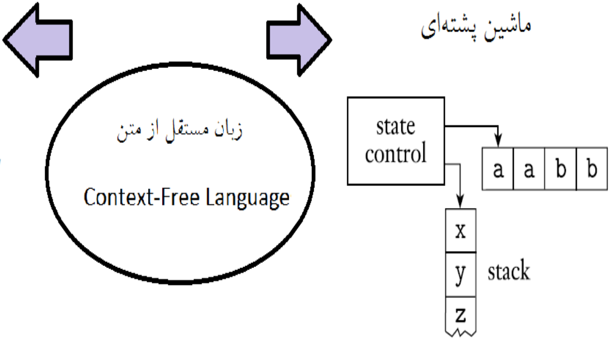
ماشینهای پشته‌ای و گرامرهای مستقل از متن معادل هستند

یادآوری: زبان A را مستقل از متن گوییم اگر گرامر مستقل از متن G وجود داشته باشد بطوریکه $L(G) = A$

قضیه: زبان A مستقل از متن است اگر و فقط اگر ماشین پشته‌ای M وجود داشته باشد بطوریکه $L(M) = A$

گرامر مستقل از متن

$$
\begin{aligned}
S_0 &\to AA_1 \mid UB \mid \mathrm{a} \mid SA \mid AS \\
S &\to AA_1 \mid UB \mid \mathrm{a} \mid SA \mid AS \\
A &\to \mathrm{b} \mid AA_1 \mid UB \mid \mathrm{a} \mid SA \mid AS \\
A_1 &\to SA \\
U &\to \mathrm{a} \\
B &\to \mathrm{b}
\end{aligned}
$$



قضیه اسلاید قبل را می‌توانیم در قالب دو لم زیر بنویسیم:

لم ۱ : اگر G یک گرامر مستقل از متن باشد آنگاه ماشین پشته‌ای M وجود دارد بطوریکه $L(M) = L(G)$

$$G \quad \Rightarrow \quad M$$

لم ۲ : اگر M یک ماشین پشته‌ای باشد آنگاه گرامر مستقل از متن G وجود دارد بطوریکه

$$L(G) = L(M)$$

$$M \quad \Rightarrow \quad G$$

اثبات لم ۱ آسانتر است و ما ایده اثبات آن را بطور مختصر ارائه می‌کنیم اما اثبات لم ۲ به آسانی لم ۱ نیست و نیاز به قدم ها و جزئیات بسیار دارد. اثبات این لم را می‌توانید در کتاب مرجع ببینید.

چگونه یک گرامر را به یک ماشین پشته‌ای تبدیل کنیم؟

$$
\begin{aligned}
S &\to ASA \mid \mathrm{a}B \\
A &\to B \mid S \\
B &\to \mathrm{b} \mid \varepsilon
\end{aligned}
$$

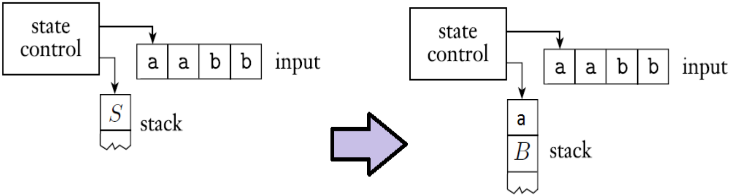
فرض کنید گرامر G داده شده است. می‌خواهیم ماشین پشته‌ای M را بسازیم بطوریکه اگر رشته w توسط گرامر G تولید شود، توسط ماشین M نیز پذیرفته شود. علاوه بر این اگر رشته w توسط گرامر G تولید نشود، توسط ماشین پشته‌ای M هم پذیرفته نشود.

از ابزار عدم قطعیت در ماشین پشته‌ای استفاده می‌کنیم. با داشتن قوانین جایگذاری گرامر G، ماشین پشته‌ای M در صورت لزوم هر بار قانون جایگذاری بعدی را حدس می‌زند و اگر این حدسها منجر به تولید رشته ورودی شد آنگاه توسط ماشین M هم پذیرفته می‌شود.

طرز کار ماشین M را می‌توان بصورت زیر خلاصه کرد:

- رشته ورودی w روی نوار input قرار گرفته است.
- متغیر شروع را در پشته قرار می‌دهیم و سپس پروسه زیر را تکرار می‌کنیم:
- اگر حرف بالای پشته یک حرف الفبا باشد، آن را با حرف روی نوار input مطابقت می‌دهیم. اگر یکسان بودند، حرف بالای پشته را برمی‌داریم و نوک خواندن روی نوار input یک واحد به جلو می‌رود.
اگر علامت بالای پشته یک متغیر بود، یکی از قوانین که طرف چپ آن متغیر مذکور است را انتخاب می‌کنیم و بالای پشته را با متغیر مورد نظر جایگزین می‌کنیم.
- پروسه بالا را آنقدر تکرار می‌کنیم تا اینکه پشته خالی شود و رشته ورودی کاملا خوانده شود.

$$
\begin{aligned}
S &\rightarrow ASA \mid \mathtt{a}B \\
A &\rightarrow B \mid S \\
B &\rightarrow \mathtt{b} \mid \varepsilon
\end{aligned}
$$



ابهام در زبان انسانی

ابهام در زبان انسانی امری طبیعی است و گاها حتی باعث غنای سخن می‌شود اگر چه در بعضی جاها مشکل‌ساز است.

Jane saw the man with the telescope.

آیا جین مردی را با استفاده از تلسکوپ دیده یا مردی را دیده که تلسکوپ با خود داشته؟

ابهام لغوی (ایهام) در شعر حافظ:

ز گریه مردم چشمم نشسته بر خون است
ببین که در طلبت حال مردمان چون است

ابهام نحوی در شعر حافظ:

جلوه‌ای کرد رخت دید ملک عشق نداشت
عین آتش شد از این غیرت و بر آدم زد

ابهام در گرامرها

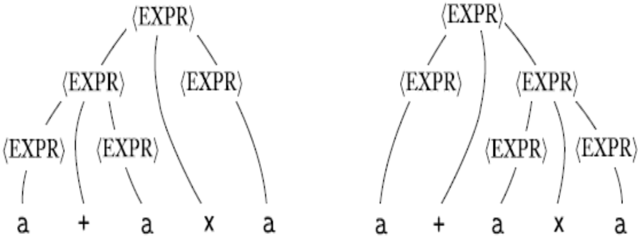
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle$$

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle$$

$$\langle \text{EXPR} \rangle \rightarrow (\langle \text{EXPR} \rangle)$$

$$\langle \text{EXPR} \rangle \rightarrow \text{a}$$

تعریف: می‌گوییم گرامر G ابهام دارد اگر یک رشته در زبان G باشد که دو درخت تجزیه متفاوت داشته باشد. مانند مثال زیر



زبانهای برنامه‌نویسی نباید ابهام داشته باشند چون در صورت وجود ابهام یک برنامه نوشته شده در آن زبان را می‌توان به چند طریق تفسیر کرد. تفسیرهای مختلف می‌توانند منجر به اجراهای کاملا مختلف شوند.

گرامر زیر یک نمونه خیلی کوچک از گرامرهایی است که در طراحی زبانهای برنامه‌نویسی مرسوم هستند. قطعه گرامر زیر ابهام دارد. چرا؟

$$
\begin{aligned}
\langle \text{STMT} \rangle &\to \langle \text{ASSIGN} \rangle \mid \langle \text{IF-THEN} \rangle \mid \langle \text{IF-THEN-ELSE} \rangle \\
\langle \text{IF-THEN} \rangle &\to \texttt{if condition then } \langle \text{STMT} \rangle \\
\langle \text{IF-THEN-ELSE} \rangle &\to \texttt{if condition then } \langle \text{STMT} \rangle \texttt{ else } \langle \text{STMT} \rangle \\
\langle \text{ASSIGN} \rangle &\to \texttt{a:=1}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma &= \{\texttt{if}, \texttt{condition}, \texttt{then}, \texttt{else}, \texttt{a:=1}\} \\
V &= \{\langle \text{STMT} \rangle, \langle \text{IF-THEN} \rangle, \langle \text{IF-THEN-ELSE} \rangle, \langle \text{ASSIGN} \rangle\}
\end{aligned}
$$

If condition then if condition then a:=1 else a:=1

دو جور اشتقاق چپ می‌توان برای جمله بالا پیشنهاد داد. اشتقاق چپ، اشتقاقی است که در هر قدم متغیر انتهای سمت چپ جایگزین می‌شود.

یک اشتقاق چپ:

<STMT> ⇒ <IF-THEN-ELSE>
⇒ if condition then <STMT> else <STMT>
⇒ if condition then if condition then <STMT> else <STMT>
⇒ if condition then if condition then a:=1 else <STMT>
⇒ if condition then if condition then a:=1 else a:=1

یک اشتقاق چپ متفاوت:

<STMT> ⇒ <IF-THEN>
⇒ if condition then <STMT>
⇒ if condition then if condition then <STMT> else <STMT>
⇒ if condition then if condition then a:=1 else <STMT>
⇒ if condition then if condition then a:=1 else a:=1

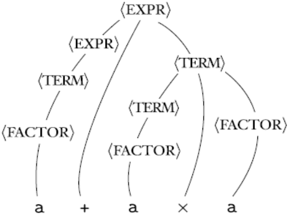
یک گرامر بدون ابهام برای تولید عبارات حسابی

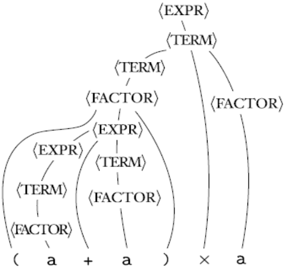
$G_4 = (V, \Sigma, R, \langle \text{EXPR} \rangle)$

V is $\{\langle \text{EXPR} \rangle, \langle \text{TERM} \rangle, \langle \text{FACTOR} \rangle\}$

Σ is $\{\mathtt{a}, \mathtt{+}, \mathtt{x}, \mathtt{(}, \mathtt{)}\}$

$$
\begin{aligned}
\langle \text{EXPR} \rangle &\rightarrow \langle \text{EXPR} \rangle \mathtt{+} \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\
\langle \text{TERM} \rangle &\rightarrow \langle \text{TERM} \rangle \mathtt{x} \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle \\
\langle \text{FACTOR} \rangle &\rightarrow \mathtt{(} \langle \text{EXPR} \rangle \mathtt{)} \mid \mathtt{a}
\end{aligned}
$$





بعضی از زبانهای مستقل از متن بصورت ذاتی مبهم هستند، یعنی هر گرامر مستقل از متن که معادل با این زبانها باشد مبهم خواهد بود. برای مثال زبان زیر بطور ذاتی مبهم است.

$$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

گرامرهای مستقل از متن معین deterministic context-free grammers که معادل با ماشینهای پشته‌ای معین هستند، فارغ از ابهام هستند. این گرامرها نقش اساسی در طراحی و تولید زبانهای برنامه نویسی ایفا می‌کنند.

فرم نرمال چامسکی

تعریف: گرامر مستقل از متن G بصورت فرم نرمال چامسکی است اگر هر قانون G یکی از حالات زیر را داشته باشد.

$$A \to BC$$

$$A \to a$$

یعنی طرف راست هر قانون یا اتصال دو متغیر باشد یا اینکه یک ترمینال باشد. علاوه بر این، متغیر شروع نمی‌تواند در طرف راست یک قانون بیاید. همچنین، طرف راست هیچ قانونی نمی‌تواند ϵ باشد به غیر از حالتی که طرف چپ متغیر شروع باشد. به عبارت دیگر، تنها قانون $S \to \epsilon$ مجاز است.

قضیه: هر گرامر مستقل از متن را می‌توان به فرم نرمال چامسکی تبدیل کرد بدون اینکه زبان آن تغییر کند.

کاربرد فرم نرمال چامسکی

اگر گرامر G بصورت فرمال نرمال چامسکی باشد، آنگاه الگوریتمی قطعی وجود دارد که، با داشتن G و رشته w، در زمان متناهی مشخص می‌کند که رشته w توسط گرامر G تولید می‌شود یا نه.

شیوه کار الگوریتم: اگر $w = \epsilon$ و گرامر G قانون $S \to \epsilon$ را داشته باشد آنگاه G رشته w را تولید می‌کند در غیر اینصورت چون هیچ متغیر دیگری تبدیل به ϵ نمی‌شود واضح است که $\epsilon \notin L(G)$.

فرض کنید که $w \neq \epsilon$ و طول رشته w برابر با ℓ باشد. گرامر همه اشتقاقهای چپی که طول عبارات تولید شده حداکثر ℓ باشد را چک میکند. اگر یکی از این عبارات برابر با w شد الگوریتم گزارش می‌کند که رشته تولید شده در زبان G واقع است در غیر اینصورت گزارش میکند که $w \notin L(G)$.

دقت کنید هر قانون جایگذاری به شکل $A \to BC$ باعث می‌شود طول عبارت حاصل یک واحد افزایش پیدا کند. پس بعد از حداکثر ℓ بار اعمال این قانون طول عبارت حاصل به ℓ می‌رسد.

مثال

گرامر زیر در فرم نرمال چامسکی قرار دارد.

$$
\begin{aligned}
S_0 &\rightarrow AA_1 \mid UB \mid \mathsf{a} \mid SA \mid AS \\
S &\rightarrow AA_1 \mid UB \mid \mathsf{a} \mid SA \mid AS \\
A &\rightarrow \mathsf{b} \mid AA_1 \mid UB \mid \mathsf{a} \mid SA \mid AS \\
A_1 &\rightarrow SA \\
U &\rightarrow \mathsf{a} \\
B &\rightarrow \mathsf{b}
\end{aligned}
$$

می‌خواهیم بدانیم آیا رشته $w = abab$ توسط این گرامر تولید می‌شود یا نه.

باید ببینیم با شروع از متغیر S_0 چند عبارت مختلف با طول 4 قابل تولید است و آیا یکی از آنها برابر با w است یا نه.

یک اشتقاق چپ را در نظر بگیرید.

$$S_0 \Rightarrow AA_1 \Rightarrow AA_1A_1 \Rightarrow AA_1A_1A_1$$

این عبارت قابل تبدیل به ترمینال نیست. باید یک اشتقاق دیگر را امتحان می‌کنیم.

$$S_0 \Rightarrow UB \Rightarrow ab$$

باز هم ناموفق بود. باید یک اشتقاق دیگر را امتحان کنیم. همینطور همه اشتقاقها که طول عبارت نهایی 4 باشد را امتحان می‌کنیم.

یک الگوریتم بسیار سریعتر به نام الگوریتم CYK برای تجزیه رشته داده شده w توسط گرامر مستقل از متن G که در فرم نرمال چامسکی قرار دارد وجود دارد.

تبدیل به فرم نرمال چامسکی

چگونه گرامر مستقل از متن G را به فرم نرمال چامسکی تبدیل کنیم؟

$$
\begin{aligned}
S &\to ASA \mid \mathtt{a}B \\
A &\to B \mid S \\
B &\to \mathtt{b} \mid \varepsilon
\end{aligned}
$$

◄ قانون $S_0 \to S$ را اضافه می‌کنیم و S_0 را متغیر شروع قرار می‌دهیم.

$$
\begin{aligned}
S &\to ASA \mid \mathtt{a}B \\
A &\to B \mid S \\
B &\to \mathtt{b} \mid \varepsilon
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid \mathtt{a}B \\
A &\to B \mid S \\
B &\to \mathtt{b} \mid \varepsilon
\end{aligned}
$$

- حال باید همه قوانین به فرم $A \to \epsilon$ وقتی که A متغیر شروع نباشد را حذف کنیم. برای اینکه در اثر حذف این قوانین زبان گرامر تغییر نکند، برای هر قانونی که A در طرف سمت راست آن قرار دارد یک قانون جدید اضافه می‌کنیم که در آن A حذف شده است. برای مثال اگر قانون $R \to aAbAc$ را داشته باشیم آنگاه باید قوانین

$$R \to abAc, \quad R \to aAbc, \quad R \to abc$$

را اضافه کنیم. اگر قانون $R \to A$ را داشته باشیم باید قانون $R \to \epsilon$ را اضافه کنیم مگر اینکه قبلا قانون $R \to \epsilon$ را حذف کرده باشیم.

$$\begin{array}{l} S_0 \to S \\ S \to ASA \mid aB \\ A \to B \mid S \\ B \to b \mid \varepsilon \end{array} \quad \Rightarrow \quad \begin{array}{l} S_0 \to S \\ S \to ASA \mid aB \mid a \\ A \to B \mid S \mid \varepsilon \\ B \to b \end{array} \quad \Rightarrow \quad \begin{array}{l} S_0 \to S \\ S \to ASA \mid aB \mid a \mid SA \mid AS \mid S \\ A \to B \mid S \\ B \to b \end{array}$$

◂ حال باید قوانین به فرم $A \to B$ را حذف کنیم. بعد از حذف این قانون هر جا که قانونی مثل $B \to u$ داشته باشیم، قانون $A \to u$ را نیز اضافه می‌کنیم. دقت کنید اینجا u می‌تواند هر عبارتی باشد و لزوما یک ترمینال نیست. اگر u یک متغیر تنها باشد و ما قبلا قانون $A \to u$ را حذف کرده‌ایم، این قانون را اضافه نمی‌کنیم.

$$\begin{array}{l} S_0 \to S \\ S \to ASA \mid aB \mid a \mid SA \mid AS \mid S \\ A \to B \mid S \\ B \to b \end{array} \;\Rightarrow\; \begin{array}{l} S_0 \to S \\ S \to ASA \mid aB \mid a \mid SA \mid AS \\ A \to B \mid S \\ B \to b \end{array} \;\Rightarrow\; \begin{array}{l} S_0 \to \mathbf{ASA \mid aB \mid a \mid SA \mid AS} \\ S \to ASA \mid aB \mid a \mid SA \mid AS \\ A \to B \mid S \\ B \to b \end{array}$$

$$\Rightarrow\; \begin{array}{l} S_0 \to ASA \mid aB \mid a \mid SA \mid AS \\ S \to ASA \mid aB \mid a \mid SA \mid AS \\ A \to S \mid \mathbf{b} \\ B \to b \end{array} \;\Rightarrow\; \begin{array}{l} S_0 \to ASA \mid aB \mid a \mid SA \mid AS \\ S \to ASA \mid aB \mid a \mid SA \mid AS \\ A \to b \mid \mathbf{ASA \mid aB \mid a \mid SA \mid AS} \\ B \to b \end{array}$$

- حال همه قوانین به شکل

$$A \to u_1 u_2 \dots u_k$$

را باید حذف کنیم (طول عبارت سمت راست بیشتر از 2 است.) اینجا u_i یک متغیر است یا یک ترمینال. بعد از این حذف سلسله قوانین زیر را اضافه می‌کنیم.

$$A \to u_1 A_1$$
$$A_1 \to u_2 A_2$$
$$\dots$$
$$A_{k-2} \to u_{k-1} u_k$$

حال برای هر u_i یک قانون به شکل $U_i \to u_i$ اضافه می‌کنیم مگر اینکه قبلا ترمینال u_i بصورت تکی در طرف سمت راست یک قانون باشد.

$$
\begin{aligned}
S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\
S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\
A &\to \mathtt{b} \mid \boldsymbol{ASA} \mid \mathbf{a}\boldsymbol{B} \mid \mathbf{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \\
B &\to \mathtt{b}
\end{aligned}
$$

$$\Longrightarrow$$

$$
\begin{aligned}
S_0 &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
S &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
A &\to \mathtt{b} \mid AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
A_1 &\to SA \\
U &\to \mathtt{a} \\
B &\to \mathtt{b}
\end{aligned}
$$