دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

فصل پنجم

زبان‌های مستقل از متن
Context Free Language

ریحانه عامری

اردیبهشت ماه ۱۳۹۹

فهرست مطالب

- ✓ زبان‌های مستقل از متن : **Context Free Language (CFL)**
- ✓ برای ساخت زبان‌های قوی‌تر باید تا حدی از قید محدودیت‌های موجود در گرامرهای منظم رها شویم.
- ✓ زبان‌های مستقل از متن در طراحی زبان‌های برنامه‌سازی و ساخت کامپایلرها کاربرد فراوانی دارد.

- ✓ گرامر $G(V,T,S,P)$ را در صورتی مستقل از متن می‌نامیم که همه قوانین p به صورت:

$$\begin{cases} A \rightarrow X \\ X \in (V \cup T)^* \, , A \in V \end{cases}$$

باشند و زبان L مستقل از متن است، اگر و تنها اگر گرامر مستقل از متن G موجود باشد بطوریکه $L = L(G)$

- ✓ در این نوع گرامرها، محدودیت سمت راست قوانین موجود در گرامرهای منظم حذف شده است.

- ✓ هر گرامر منظم، مستقل از متن است و درنتیجه زبان منظم نیز، زبان مستقل از متن است. ولی زبان $L = \{a^n b^n \mid n \geq 0\}$ منظم نیست و با گرامر مستقل از متنی بوجود می‌آید.

بنابراین: زبان‌های منظم زیرمجموعه‌ای از زبان‌های مستقل از متن است.

- ✓ در گرامرهای مستقل از متن، جایگزینی متغیرهای سمت چپ یک قانون را در هر زمانی که این متغیر در یک شکل جمله‌ای دیده می‌شود، می‌توان انجام داد و به بقیه شکل جمله بستگی ندارد. به این دلیل مستقل از متن نامیده شده است.

✓مثال

$$G = (\{S\},\{a,b\},S,P)$$

$$P \begin{cases} S \to aSa \\ S \to bSb \\ S \to \lambda \end{cases}$$

$S \to aSa \to aaSaa \to aabbaa$ نمونه اشتقاق

گرامر G مستقل از متن است ولی منظم نیست.

$$L(G) = \{ ww^R \mid w \in \{a,b\}^* \}$$

✓ مثال

$G = (\{S,A,B\},\{a,b\},S,P)$

$$P \begin{cases} S \to abB \\ A \to aaB \mid \lambda \\ B \to bbA \end{cases}$$

$$L(G) = \{\, ab(bbaa)^n bb \mid n >= 0 \,\}$$

- این گرامر، گرامر مستقل از متن و خطی است. گرامرهای منظم و خطی مستقل از متن هستند ولی یک گرامر مستقل از متن لزوما خطی یا منظم نیست.

✓ مثال

$$L(G) = \{ a^n b^m \mid n \neq m \}$$

حالت $n > m$

$$\begin{cases} S \rightarrow AS_1 \\ S_1 \rightarrow aS_1b \mid \lambda \\ A \rightarrow aA \mid a \end{cases}$$

حالت $n < m$

$$\begin{cases} S \rightarrow S_1B \\ S_1 \rightarrow aS_1b \mid \lambda \\ B \rightarrow bB \mid b \end{cases}$$

$$\begin{cases} S \rightarrow AS1 \mid S1B \\ S1 \rightarrow aS1b \mid \lambda \\ A \rightarrow aA \mid a \\ B \rightarrow bB \mid b \end{cases}$$

✓ زبان L مستقل از متن است.

✓ مثال

$$G = (\{S\},\{a,b\},S,P)$$

$$P : S \rightarrow SS \mid aSb \mid bSa \mid \lambda$$ →

$$L(G) = \{ w \in \{a,b\}^* \mid n_a(w) = n_b(w) \}$$

✓ گرامر G مستقل از متن است.

✓ در زبان‌های برنامه‌سازی می‌توان از این نوع گرامر زمانی استفاده کرد که مثلا در عبارت $(((a+b)-c)/d)*e$ باید تعداد) و (یکسان باشد.

✓ اشتقاق‌های چپ و راست:

زبان‌های مستقل از متن که خطی نیستند ⬅ اشتقاق‌ها حاوی جملاتی با بیشتر از یک متغیر هستند.

مثال:

1) اشتقاق چپ $S \rightarrow AB \rightarrow aaAB \rightarrow aaB \rightarrow aaBb \rightarrow aab$

2) اشتقاق راست $S \rightarrow AB \rightarrow ABb \rightarrow aaABb \rightarrow aaAb \rightarrow aab$

هر دو اشتقاق یک رشته تولید می‌کنند.

$$G = (\{S,A,B\},\{a,b\},S,P)$$

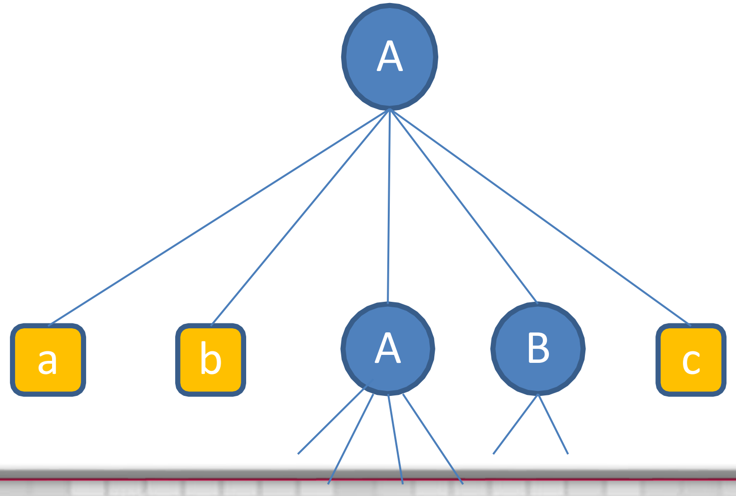
$$P \begin{cases} S \rightarrow AB \\ A \rightarrow aaA \mid \lambda \\ B \rightarrow Bb \mid \lambda \end{cases}$$

$$L(G) = \{ a^{2n} b^{m} \mid n >= 0 \ \& \ m >= 0 \}$$

✓ درخت اشتقاق

یک درخت است که در آن گره‌ها با سمت چپ قوانین علامت‌گذاری می‌شوند و بچه‌های یک گره، سمت راست آن قانون هستند.

مثال: درخت اشتقاق قانون $A \rightarrow abABc$ به شکل ذیل است:



✓ تعریف درخت اشتقاق:

با فرض مستقل از متن بودن گرامر $G = (V,T,S,P)$، یک درخت، درخت اشتقاق برای G محسوب می‌شود، اگر و تنها اگر دارای خواص ذیل باشد:

1. **ریشه برچسب s داشته باشد.**
2. **هر یک از برگ‌ها، برچسبی از $T \cup \{\lambda\}$** باشد.
3. **هر یک از گره‌های درونی (غیر برگ) دارای برچسبی از V باشد.**
4. **اگر گره ای با برچسب $A \in V$ و فرزندان آن از چپ به راست به صورت** $a_1, a_2, a_3 \ldots a_n$ **برچسب گذاری شوند آنگاه p دارای قانون** $A \rightarrow a_1, a_2, a_3 \ldots a_n$ **می‌باشد.**
5. **گره‌ای که برچسب λ داشته باشد، هیچ فرزند دیگری ندارد.**

درخت اشتقاق ذاتا مبهم است.

✓ درخت اشتقاق جزئی

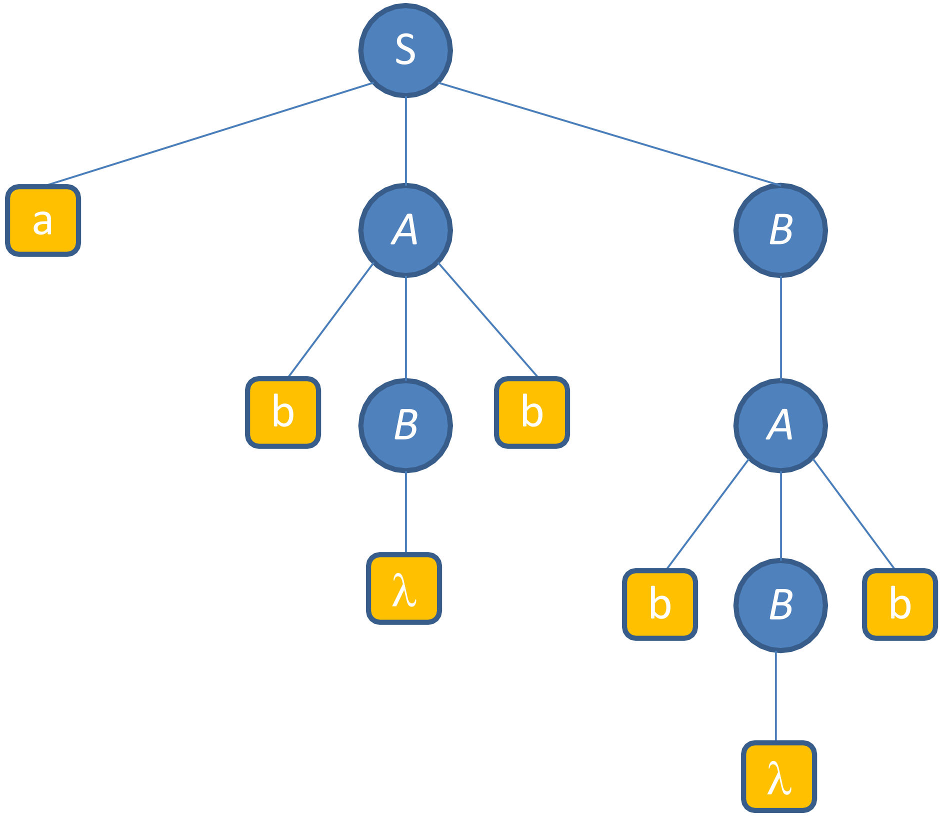
درختی که دارای خاصیت ۳،۴ و ۵ باشد و خاصیا ۱ در آن لزوما صدق نکند و خاصیت ۲ به شکل λ U T U v باشد.

✓ رشته‌های که از خواندن برگ‌های درخت از چپ به راست، بدست می‌آید حاصل درخت گویند.

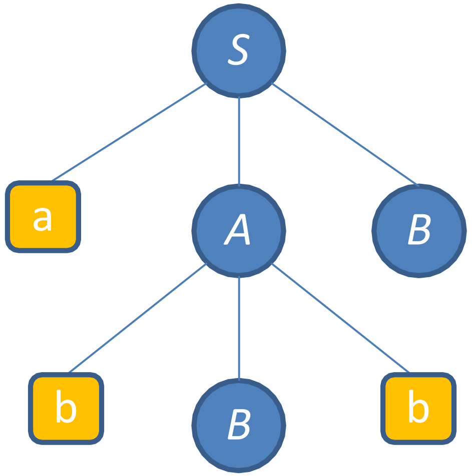
مثال: $G = (\{S,A,B\},\{a,b\},S,P)$

$$P \begin{cases} S \rightarrow aAB \\ A \rightarrow bBb \\ B \rightarrow A \mid \lambda \end{cases}$$

درخت اشتقاق G



درخت اشتقاق جزئی



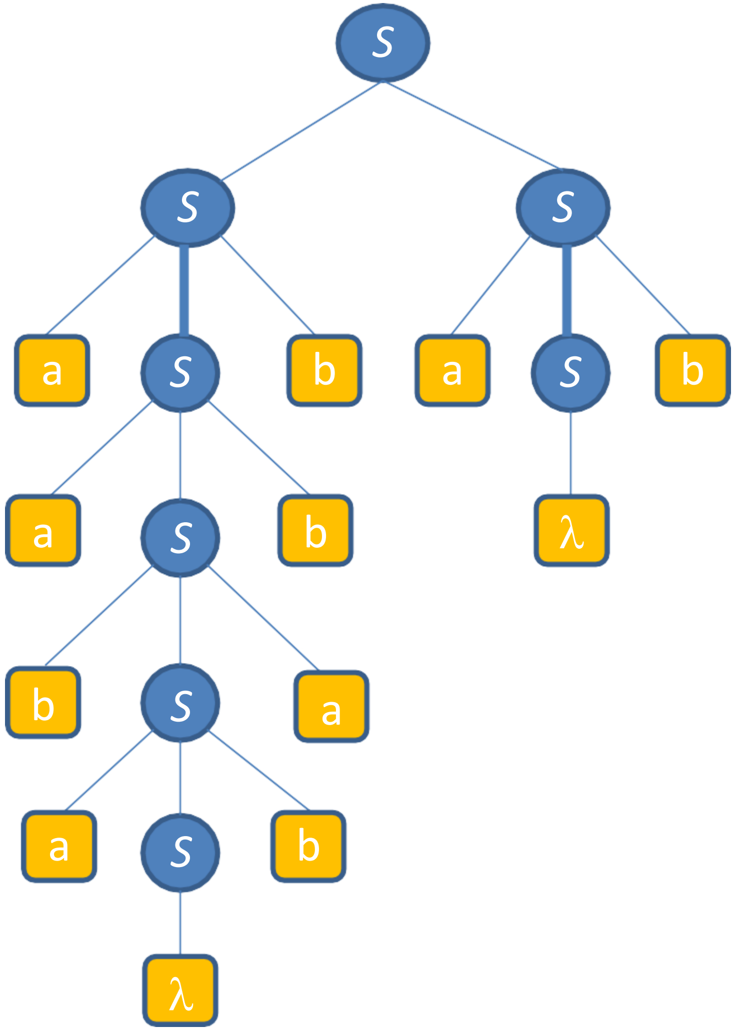
$$L(G) = \{ w \in \{a,b\}^* \mid n_a(w) = n_b(w) \}$$

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$

اشتقاق چپ!

رشته فرضي

aababababbab



✓ قضیه:

با فرض مستقل از متن بودن، $G = (V,T,S,P)$ به ازای هر $\forall w \in L(G)$ یک درخت اشتقاق برای G وجود دارد که حاصل آن W است و برعکس.

حاصل درخت اشتقاق در L(G) است.

اگر t_G یک درخت اشتقاق جزئی برای G با ریشه S باشد، آنگاه حاصل t_G یک شبه جمله‌ای از G می‌باشد.

✓ پویش (Parsing)

یافتن یک سری قانون که با استفاده از آن‌ها $W \in L(G)$ مشتق می‌شود.

✓ الگوریتم عضویت (Membership)

کلیه اشتقاق‌های چپ ممکن را می‌سازیم و بررسی می‌کنیم که آیا یکی از آن‌ها با **W** منطبق می‌شود یا نه.

در دور اول همه‌ی قوانینی که به شکل $S \rightarrow x$ هستند را می‌یابیم. اگر هیچ یک با **w** انطباق نداشت، آنگاه در دور دوم، همه‌ی قوانینی که قابل اعمال بر روی متغیر چپ همه‌ی **x** ها است، می‌یابیم.

این روش، یک اشتقاق چپ از **w** ارائه می‌کند. این روش، روش پویش جامع است و نوعی پویش بالا به پایین (ریشه به برگ‌ها) می‌باشد.

✓ **مثال:** گرامر ذیل را درنظر بگیرید.

$$S \xrightarrow{} \overset{1}{SS} \mid \overset{2}{aSb} \mid \overset{3}{bSa} \mid \overset{4}{\lambda}$$

$w = aabb$

حذف دستور 3 و 4 (عدم انطباق با w)

دور اول → P

$$\begin{cases} 1.\ S \to SS \\ 2.\ S \to aSb \\ 3.\ S \to bSa \\ 4.\ S \to \lambda \end{cases}$$

رشته نهایی:

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$

این روش دارای مشکل نامعین بودن و عدم توقف است. اگر رشته عضوی اززبان نباشد و قوانینی به شکل $A \rightarrow B \ , \ A \rightarrow \lambda$ داشته باشیم، الگوریتم هیچ گاه به پایان نمی‌رسد.

دور دوم → P

1. $S \rightarrow SS \rightarrow SSS$
2. $S \rightarrow SS \rightarrow aSbS$
3. $S \rightarrow SS \rightarrow bSaS$
4. $S \rightarrow SS \rightarrow S$

دور سوم → P

5. $S \rightarrow aSb \rightarrow aSSb$
6. $S \rightarrow aSb \rightarrow aaSbb$
7. $S \rightarrow aSb \rightarrow abSab$
8. $S \rightarrow aSb \rightarrow ab$

✓ گرامر ساده (Simple Grammar) :

$G = (V,T,S,P)$، گرامر ساده است اگر تمام قوانین آن به فرم $A \rightarrow aX$ است که در آن

$X \in V^*$, $a \in T$, $A \in V$ و هر زوج (A,a) حداکثر یک بار در P وجود داشته باشد.

مثال:

1) $S \rightarrow aS|bSS|c$ *S-Grammar*

2) $S \rightarrow aS|bSS|aSS|c$ تکرار زوج (S,a)

✓ گرامر مبهم

گرامر مستقل از متن G را درصورتی مبهم می‌گوییم که یک $w \in L(G)$ وجود داشته باشد، که حداقل دو درخت اشتقاق متفاوت داشته باشد.

✓ مثال :

$S \rightarrow aSb|SS|\lambda$ مبهم است $\rightarrow$(aaabbb)

✓ مثال:

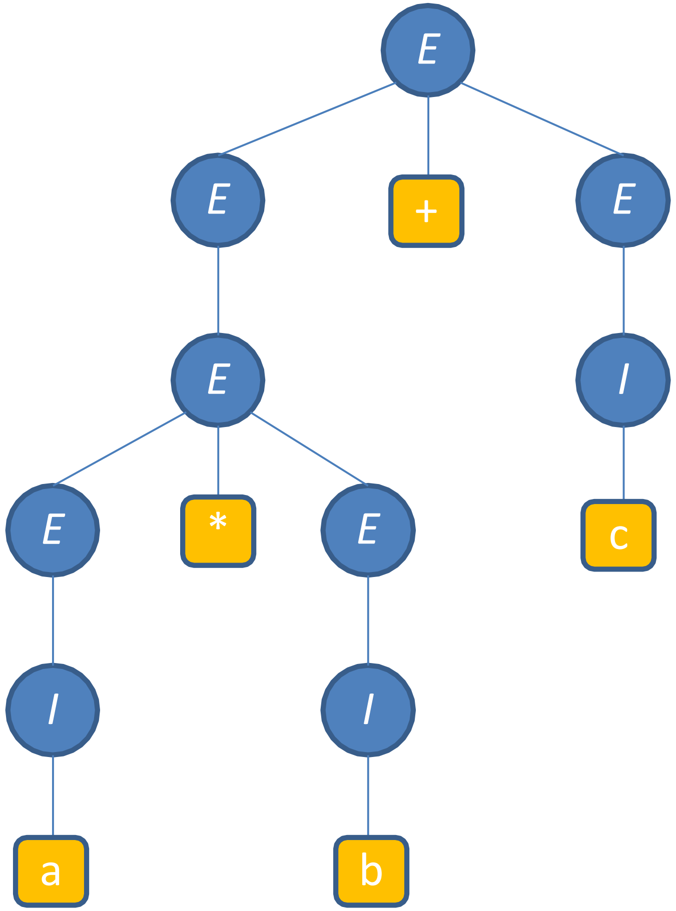
گرامر $G = (V,T,S,P)$ و $T = \{ a,b,c,+,*,(,) \}$ و $V = \{ E,I \}$ وقوانین:

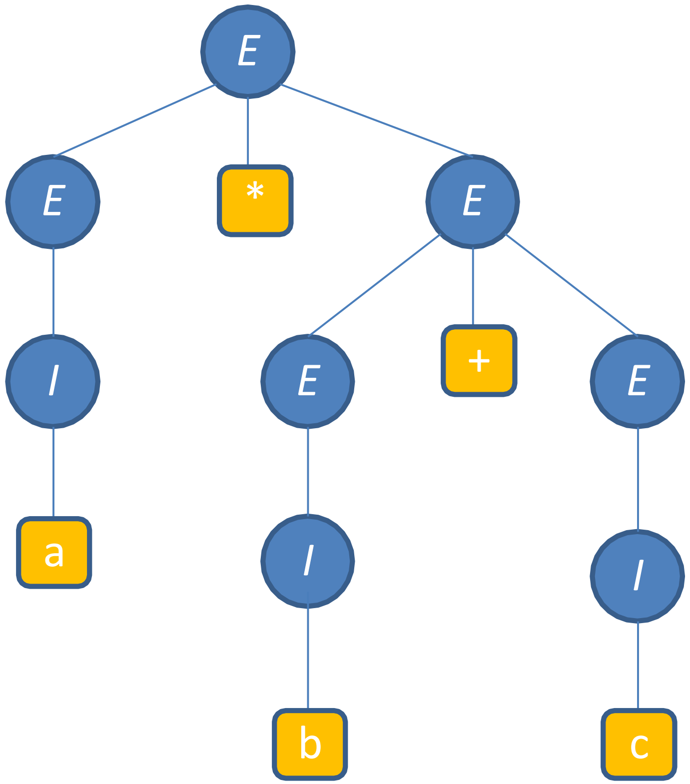
$$P \begin{cases} E \rightarrow I \mid E+E \mid E*E \mid (E) \\ I \rightarrow a \mid b \mid c \end{cases}$$

$\left\{ \begin{array}{l} a*b+c \\ a+b*c \end{array} \right.$ ⇦ مبهم است به علت پیمایش عبارت‌های

پیمایش

$\xrightarrow{a*b+c}$





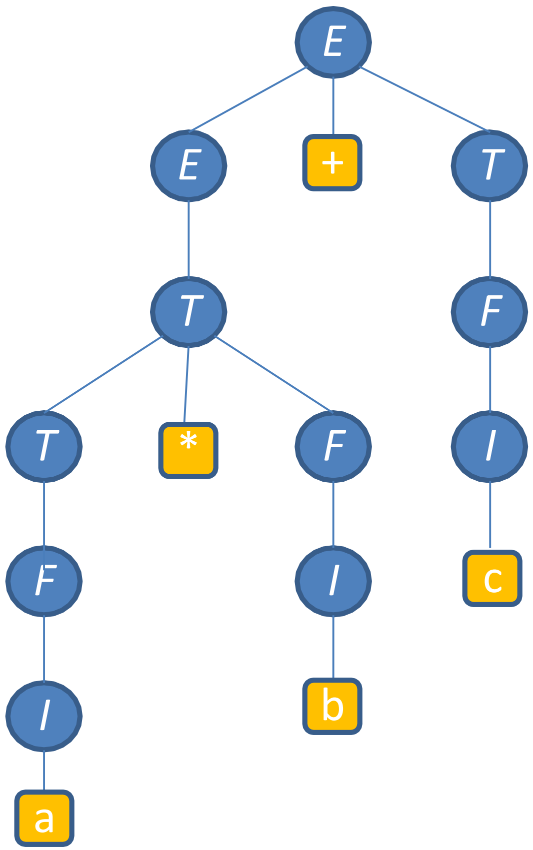
✓ برای رفع ابهام می‌توان از تقدم عملگرها استفاده کرد: گرامر ذیل مبهم است

$E \rightarrow T | E+T$

$T \rightarrow F | T*F$

$F \rightarrow (E) | I$

$I \rightarrow a | b | c$



expr → *expr* +*term* | *expr* – *term* | *term*

term → *term* * *factor* | *term* / *factor* |

factor *factor* → (*expr*) | *digit*

digit → 0|1|…|9

✓ بعضی از زبان‌ها ذاتا مبهم هستند و گرامر غیر مبهمی برای آن‌ها وجود ندارد.

✓ مثال: زبان ، $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$ با فرض $m,n >= 0$ ،مستقل از متن است و ذاتا مبهم است.

$L \rightarrow L_1 \cup L_2 \,;\, S \rightarrow S_1 \mid S_2$

$S \rightarrow S_1$: S_1 تعداد a , b ها را محدود مي كند.

$S \rightarrow S_2$: S_2 تعداد b , c ها را محدود مي كند.

$S_1 \rightarrow S_1 c \mid A \quad A \rightarrow aAb \mid \lambda$

$S_2 \rightarrow aS_2 \mid B \quad B \rightarrow bAc \mid \lambda$

- ✓ هر گرامر ساده غیرمبهم است.
- ✓ الگوریتم عضویت برای گرامرهای ساده با $\theta(|w|)$ وجود دارد.
- ✓ زبان منظم زبانی است که نمی‌تواند ذاتا مبهم باشد.

بــا سپــا س فــر او ان