به نام خدا

نظریهٔ زبان‌ها و ماشین‌ها

آرش شفیعی

زبان‌های مستقل از متن

مقدمه

- حال که با محدودیت‌های زبان‌های منظم آشنا شدیم، یک دستهٔ دیگر از زبان‌ها را بررسی می‌کنیم که زبان‌های مستقل از متن نامید می‌شوند و محدودیت‌های زبان‌های منظم را ندارند.
- زبان‌های منظم دسته‌ای محدود شده از زبان‌های مستقل از متن و بنابراین زیرمجموعهٔ این دسته از زبان‌ها به شمار می‌آیند.
- یکی از زبان‌های سادهٔ غیرمنظم زبان $L = \{a^n b^n : n \geq 0\}$ بود. در این قسمت این دسته از زبان‌ها را به همراه گرامر آنها بررسی می‌کنیم.

گرامرهای مستقل از متن

- برای قوانین تولید در گرامرهای منظم محدودیتی در نظر گرفتیم. سمت چپ قانون در گرامرهای منظم همیشه یک متغیر و سمت راست ترکیبی از متغیرها و نمادهای پایانی است به طوری که متغیر فقط در ابتدا و یا فقط در انتهای عبارت سمت راست قرار می‌گیرد.
- در گرامرهای مستقل از متن از این محدودیت می‌کاهیم.

گرامرهای مستقل از متن

- گرامر $G = (V, T, S, P)$ یک گرامر مستقل از متن [1] نامید می‌شود اگر همهٔ قوانین تولید P در آن به صورت $A \to x$ باشند به طوری که $A \in V$ یک متغیر و $x \in (V \cup T)^*$ ترکیبی از متغیرها و پایانه‌ها باشد.

- زبان L یک زبان مستقل از متن نامیده می‌شود اگر و فقط اگر یک گرامر مستقل از متن وجود داشته باشد به طوری که $L = L(G)$.

- زبان‌های مستقل از متن به این دلیل اینگونه نامیده می‌شوند که در هر مرحله از اشتقاق برای به دست آوردن یک جمله از زبان، هر قانونی که یکی از متغیرهای صورت جمله‌ای را در سمت چپ داشته باشد می‌تواند اعمال شود و بنابراین اعمال قانون در فرایند اشتقاق به وابسته به متن نیست.

[1] context-free grammar

گرامرهای مستقل از متن

- گرامر $G = (\{S\}, \{a, b\}, S, P)$ با قوانین تولید $S \rightarrow \lambda \quad , \quad S \rightarrow bSb \quad , \quad S \rightarrow aSa$ یک گرامر مستقل از متن است.
- یک مثال از اشتقاق در این گرامر بدین صورت است: $S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabbaa$
- بنابراین زبان $L = \{ww^R : w \in \{a, b\}^*\}$ که توسط این گرامر وصف می‌شود یک زبان مستقل از متن است.

گرامرهای مستقل از متن

- گرامر G با قوانین تولید $S \to abB \quad , \quad A \to aaBb \quad , \quad B \to bbAa \quad , \quad A \to \lambda$ یک گرامر مستقل از متن است.
- این گرامر چه زبانی را توصیف می‌کند؟

گرامرهای مستقل از متن

- گرامر G با قوانین تولید $S \to abB \;,\; A \to aaBb \;,\; B \to bbAa \;,\; A \to \lambda$ یک گرامر مستقل از متن است.
- این گرامر زبان $L(G) = \{ab(bbaa)^n bba(ba)^n : n \geq 0\}$ را توصیف می‌کند.
- این گرامر همچنین یک گرامر خطی [1] است. در یک گرامر خطی در سمت راست هر قانون حداکثر یک متغیر وجود دارد.

[1] linear grammar

گرامرهای مستقل از متن

- نشان دهید زبان $L = \{a^n b^m : n \neq m\}$ یک زبان مستقل از متن است.

گرامرهای مستقل از متن

- برای این که نشان دهیم زبان $L = \{a^n b^m : n \neq m\}$ یک زبان مستقل از متن است، باید یک گرامر مستقل از متن برای آن پیدا کنیم.

- ابتدا حالتی را در نظر می‌گیریم که در آن $n > m$ باشد. بنابراین ابتدا تعداد مساوی a و b تولید می‌کنیم و سپس تعداد b ها را می‌افزاییم. این کار را توسط گرامر $S \to AS_1 \ , \ S_1 \to aS_1b|\lambda \ , \ A \to aA|a$ انجام می‌دهیم.

- سپس حالتی را در نظر می‌گیریم که در آن $n < m$ باشد.

- در نتیجه داریم: $S \to AS_1|S_1B \ , \ S_1 \to aS_1b|\lambda \ , \ A \to aA|a \ , \ B \to bB|b$ جواب نهایی

- این گرامر یک گرامر مستقل از متن است ولی خطی نیست.

گرامرهای مستقل از متن

- گرامر $S \rightarrow aSb|bSa|SS|\lambda$ را در نظر بگیرید.
- این گرامر چه زبانی را توصیف می‌کند؟

گرامرهای مستقل از متن

- گرامر $S \to aSb|bSa|SS|\lambda$ را در نظر بگیرید.
- این گرامر زبانی را توصیف می‌کند که در همهٔ جملات آن تعداد a و b با هم برابرند.

گرامرهای مستقل از متن

- در یک گرامر مستقل از متن که خطی نیست، در فرایند اشتقاق، اگر یک صورت جمله‌ای بیش از یک متغیر داشته باشد، آنگاه چندین انتخاب برای اشتقاق وجود دارد.
- گرامر $G = (\{S, A, B\}, \{a, b\}, S, P)$ را با قوانین تولید P در نظر بگیرید:
- ۱. $S \rightarrow AB$ ، ۲. $A \rightarrow aaA$ ، ۳. $A \rightarrow \lambda$ ، ۴. $B \rightarrow Bb$ ، ۵. $B \rightarrow \lambda$
- این گرامر چه زبانی را توصیف می‌کند؟

گرامرهای مستقل از متن

- در یک گرامر مستقل از متن که خطی نیست، در یک اشتقاق که در آن صورت جمله‌ای بیش از یک متغیر دارد، چندین انتخاب برای اشتقاق وجود دارد.
- گرامر $G = (\{S, A, B\}, \{a, b\}, S, P)$ را با قوانین تولید P در نظر بگیرید:
- $۱.\ S \rightarrow AB \ ,\ ۲.\ A \rightarrow aaA \ ,\ ۳.\ A \rightarrow \lambda \ ,\ ۴.\ B \rightarrow Bb \ ,\ ۵.\ B \rightarrow \lambda$
- این گرامر زبان $L = \{a^{۲n}b^m : n \geq ۰, m \geq ۰\}$ را توصیف می‌کند.
- اشتقاق یک جملهٔ واحد توسط این گرامر می‌تواند به اشکال گوناگون باشد. به طور مثال:
- $S \overset{۱}{\Rightarrow} AB \overset{۲}{\Rightarrow} aaAB \overset{۳}{\Rightarrow} aaB \overset{۴}{\Rightarrow} aaBb \overset{۵}{\Rightarrow} aab$
- $S \overset{۱}{\Rightarrow} AB \overset{۴}{\Rightarrow} ABb \overset{۲}{\Rightarrow} aaABb \overset{۵}{\Rightarrow} aaAb \overset{۳}{\Rightarrow} aab$

گرامرهای مستقل از متن

- یک اشتقاق را اشتقاق از چپ [1] (اشتقاق چپ یا اشتقاق چپ‌ترین) می‌نامیم اگر در هر مرحله از اشتقاق سمت چپ‌ترین متغیر در صورت جمله‌ای جایگزین شود.
- اگر در هر مرحله از اشتقاق سمت راست‌ترین صورت جمله‌ای جایگزین شود، آن اشتقاق را یک اشتقاق از راست [2] می‌نامیم.

[1] leftmost derivation
[2] rightmost derivation

- گرامری را با قوانین تولید $B \rightarrow A|\lambda \quad , \quad A \rightarrow bBb \quad , \quad S \rightarrow aAB$ در نظر بگیرید.
- اشتقاق $S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abAbB \Rightarrow abbBbbB \Rightarrow abbbbB \Rightarrow abbbb$ یک اشتقاق از چپ است.
- اشتقاق $S \Rightarrow aAB \Rightarrow aA \Rightarrow \underset{abBb}{\cancel{aBbB}} \Rightarrow abAb \Rightarrow abbBbb \Rightarrow abbbb$ یک اشتقاق از راست است.

گرامرهای مستقل از متن

- گرامر $S \rightarrow aSb|bSa|SS|\lambda$ را در نظر بگیرید.
- یک اشتقاق چپ برای جملهٔ aaabbabb پیدا کنید.

گرامرهای مستقل از متن

- گرامر $S \rightarrow aSb|bSa|SS|\lambda$ را در نظر بگیرید.
- یک اشتقاق چپ برای جملهٔ aaabbabb به صورت زیر است.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaSSbb \Rightarrow aaaSbSbb \Rightarrow aaabSbb \Rightarrow aaabbSabb \Rightarrow aaabbabb$

درخت اشتقاق

- یک روش دیگر برای نمایش اشتقاق که مستقل از ترتیب اعمال قوانین در فرایند اشتقاق است، استفاده از درخت اشتقاق [1] یا درخت تجزیه [2] است.

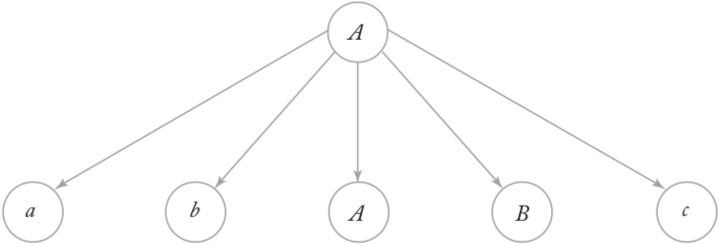
- درخت تجزیه یک درخت مرتب [3] است که در آن ریشه با متغیر آغازی برچسب زده شده است. فرزندان ریشه، نمادهایی از یک صورت جمله‌ای (با حفظ ترتیب) هستند که متغیر آغازی با آن جایگزین شده است. به همین ترتیب هر رأس میانی (رئوسی که برگ نیستند) با یک متغیر X برچسب زده شده است و فرزندان رأس X با نمادهای صورت جمله‌ای x (با حفظ ترتیب نمادهای x) برچسب زده شده‌اند به طوری که $X \to x$ یک قانون تولید در گرامر است. برگ‌های این درخت را پایانه‌ها تشکیل می‌دهند.

---

[1] derivation tree
[2] parse tree
[3] ordered tree

درخت اشتقاق

- برای مثال اگر قانون $\mathrm{A} \rightarrow \mathrm{abABc}$ را داشته باشیم، آنگاه در درخت اشتقاق می‌توانیم زیردرخت زیر را بیابیم.



درخت اشتقاق

- فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن باشد. یک درخت مرتب، درخت اشتقاق برای این گرامر است اگر و فقط اگر ویژگی‌های زیر را داشته باشد.

۱. ریشه با نماد S برچسب زده شده باشد.

۲. هر برگ با یکی از نمادهای $T \cup \{\lambda\}$ برچسب شده باشد.

۳. هر رأس میانی (رئوسی که ریشه و برگ نیستند) یک برچسب V داشته باشد.

۴. اگر رأسی برچسب $A \in V$ داشته باشد و فرزندان آن از چپ به راست برچسب‌های $a_۱, a_۲, \cdots, a_n$ داشته باشند، آنگاه یکی از قوانین تولید باید به صورت $A \to a_۱a_۲ \cdots a_n$ باشد.

۵. برگی که با نماد λ برچسب زده شده است، هیچ همزادی (رأسی که با آن رأس از یک والد باشد) ندارد. به عبارت دیگر رأسی که فرزند آن λ است هیچ فرزند دیگری ندارد.

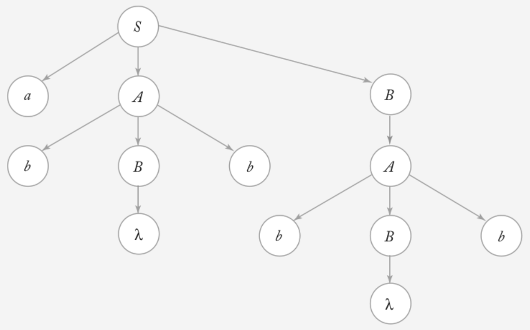
درخت اشتقاق

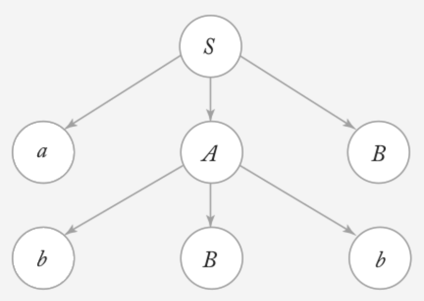
- حال اگر در یک درخت اشتقاق ریشه با یک متغیر غیر آغازی برچسب زده شده باشد و برگ‌ها متغیر یا نماد پایانی باشند، این درخت را یک درخت اشتقاق جزئی [1] می‌نامیم.
- اگر برچسب برگ‌های یک درخت را از چپ به راست به یک دیگر الحاق کنیم، محصول [2] درخت را به دست آورده‌ایم.
- در یک درخت اشتقاق محصول یک درخت، رشته‌ای از نمادهاست که توسط گرامر مربوط به آن درخت به دست آمده است.

[1] partial derivation tree
[2] yield

درخت اشتقاق

- گرامر G با قوانین تولید زیر را در نظر بگیرید: $S \rightarrow aAB \;,\; A \rightarrow bBb \;,\; B \rightarrow A|\lambda$
- درخت سمت راست یک درخت اشتقاق جزئی است که محصول آن صورت جمله‌ای abBbB است و درخت سمت چپ یک درخت اشتقاق است که محصول آن رشتهٔ abbbb است که جمله‌ای از زبان L(G) است.





درخت اشتقاق

- فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن باشد.
- آنگاه به ازای هر $w \in L(G)$ یک درخت اشتقاق وجود دارد که محصول آن w است. همچنین محصول یک درخت اشتقاق برای گرامر G رشته‌ای در $L(G)$ است.
- همچنین اگر t_G یک درخت اشتقاق جزئی برای گرامر G باشد که ریشه آن S باشد، آنگاه محصول درخت t_G یک صورت جمله‌ای است که از گرامر G به دست می‌آید (مشتق می‌شود).

درخت اشتقاق

- یک درخت اشتقاق نشان می‌دهد که جملات یک زبان چگونه از یک گرامر به دست آمده‌اند، اما چیزی در مورد ترتیب اعمال قوانین در یک اشتقاق به ما نمی‌گوید.
- بنابراین با پیمایش درخت اشتقاق با ترتیب‌های مختلف می‌توانیم یک اشتقاق از چپ یا یک اشتقاق از راست به دست بیاوریم.

مثال

- بر روی $\Sigma = \{0, 1\}$، یک گرامر مستقل از متن برای زبانی بیابید که نماد اول و آخر جملات آن یکسان باشند.

مثال

- بر روی $\Sigma = \{۰, ۱\}$، یک گرامر مستقل از متن برای زبانی بیابید که نماد اول و آخر جملات آن یکسان باشند.
- $S \rightarrow ۱T۱|۰T۰|۰|۱$
  $T \rightarrow ۰T|۱T|\lambda$

مثال

- بر روی $\Sigma = \{0, 1\}$، یک گرامر مستقل از متن برای زبانی بیابید که جملات آن حداقل سه نماد ۱ داشته باشند.

مثال

- بر روی $\Sigma = \{۰, ۱\}$، یک گرامر مستقل از متن برای زبانی بیابید که جملات آن حداقل سه نماد ۱ داشته باشند.
- $S \rightarrow T۱T۱T۱T \quad , \quad T \rightarrow ۱T|۰T|\lambda$

مثال

- بر روی $\Sigma = \{۰, ۱\}$، یک گرامر مستقل از متن برای زبانی بیابید که طول رشته‌های آن فرد است و نماد وسط رشته صفر است.

مثال

- بر روی $\Sigma = \{۰, ۱\}$، یک گرامر مستقل از متن برای زبانی بیابید که طول رشته‌های آن فرد است و نماد وسط رشته صفر است.
- $S \rightarrow ۰ | ۰S۰ | ۰S۱ | ۱S۰ | ۱S۱$

مثال

- یک گرامر مستقل از متن برای زبان $L = \{w \in \{a, b\}^* : n_a(w) \neq n_b(w)\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای زبان $L = \{w \in \{a,b\}^* : n_a(w) \neq n_b(w)\}$ بیابید.
- دو حالت را در نظر می‌گیریم: (۱) تعداد نمادهای a بیشتر است و (۲) تعداد نمادهای b بیشتر است.
- راه حل اول:
- $$S \rightarrow S_1 a S_1 | S_2 b S_2$$
$$S_1 \rightarrow S_1 S_1 | a S_1 b | b S_1 a | A \quad , \quad A \rightarrow aA | \lambda$$
$$S_2 \rightarrow S_2 S_2 | a S_2 b | b S_2 a | B \quad , \quad B \rightarrow bB | \lambda$$
- راه حل دوم:
- $$S \rightarrow A | B$$
$$A \rightarrow AAb | AbA | bAA | aA | a$$
$$B \rightarrow BBa | BaB | aBB | bB | b$$

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^n b^m c^k : n = m \vee m \leq k\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^n b^m c^k : n = m \vee m \leq k\}$ بیابید.
- مسأله را به دو قسمت تقسیم می‌کنیم: $L_1 = \{a^n b^m c^k : n = m\}$
  $L_2 = \{a^n b^m c^k : m \leq k\}$
  $L = L_1 \cup L_2$
- $S \to S_1 | S_2$
  $S_1 \to T_1 C \ , \ T_1 \to a T_1 b | \lambda \ , \ C \to cC | \lambda$
  $S_2 \to A T_2 \ , \ T_2 \to b T_2 c | C \ , \ A \to aA | \lambda$

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^n b^m c^k : k = n + ۲m\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^n b^m c^k : k = n + ۲m\}$ بیابید.
- می‌توانیم زبان L را بدین شکل درآوریم: $L = \{a^n b^m c^{۲m} c^n\}$
- $S \rightarrow aSc|B \quad , \quad B \rightarrow bBcc|\lambda$

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^n w w^R b^n : w \in \Sigma^*, n \geq ۱\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^n ww^R b^n : w \in \Sigma^*, n \geq ۱\}$ بیابید.
- $S \rightarrow aSb|aTb \quad , \quad T \rightarrow aTa|bTb|\lambda$

مثال

- بر روی $\Sigma = \{a, b\}$، یک گرامر مستقل از متن برای زبانی بیابید که همهٔ جمله‌های آن واروخوانه (پالیندروم[1]) باشند. واروخوانه به واژه‌هایی گفته می‌شود که خواندن آنها از چپ به راست و راست به چپ یکسان است.
به طور رسمی تعریف می‌کنیم w یک جملهٔ واروخوانه است اگر $w = w^R$.

[1] palindrome

مثال

- بر روی $\Sigma = \{a, b\}$، یک گرامر مستقل از متن برای زبانی بیابید که همهٔ جمله‌های آن واروخوانه (پالیندروم[1]) باشند. واروخوانه به واژه‌هایی گفته می‌شود که خواندن آنها از چپ به راست و راست به چپ یکسان است. به طور رسمی تعریف می‌کنیم w یک جملهٔ واروخوانه است اگر $w = w^R$.
- $S \rightarrow aSa|bSb|a|b|\lambda$ این تقریبا همان ww^r است

[1] palindrome

مثال

- یک گرامر مستقل از متن برای زبانی بیابید که جملات آن واروخوانه (پالیندروم) نباشند: $L = \{w \in \{a, b\}^* : w \neq w^R\}$.

مثال

- یک گرامر مستقل از متن برای زبانی بیابید که جملات آن واروخوانه (پالیندروم) نباشند: $L = \{w \in \{a, b\}^* : w \neq w^R\}$.
- $S \rightarrow aSa|bSb|aTb|bTa \quad , \quad T \rightarrow aTa|bTb|aTb|bTa|a|b|\lambda$

مثال

- بر روی $\Sigma = \{a, b\}$، یک گرامر مستقل از متن برای زبان $L = \{a^n b^m : n \neq m\}$ بیابید.

مثال

- بر روی $\Sigma = \{a, b\}$، یک گرامر مستقل از متن برای زبان $L = \{a^n b^m : n \neq m\}$ بیابید.
- دو حالت را در نظر می‌گیریم: یا تعداد نمادهای a بیشتر است و یا تعداد نمادهای b .
- $S \to AS_1|S_1B \quad , \quad S_1 \to aS_1b|\lambda \quad , \quad A \to aA|a \quad , \quad B \to bB|b$

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^m b^n c^p d^q : m, n, p, q \geq 0, m + n = p + q\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^m b^n c^p d^q : m, n, p, q \geq ۰, m + n = p + q\}$ بیابید.
- به ازای تولید هر a (و همینطور به ازای تولید هر b) باید یک c یا یک d تولید کنیم.
- دو حالت در نظر می‌گیریم:
$$m \geq q \;\; , \;\; n \leq p \; (۱)$$
$$m \leq q \;\; , \;\; n \geq p \; (۲)$$
- برای حالت اول داریم:
$$S \to aSd|A \;\; , \;\; A \to aAc|C \;\; , \;\; C \to bCc|\lambda$$
- برای حالت دوم داریم:
$$S \to aSd|B \;\; , \;\; B \to bBd|C \;\; , \;\; C \to bCc|\lambda$$
- از اجتماع این دو حالت داریم:
$$S \to aSd|A|B \;\; , \;\; A \to aAc|C \;\; , \;\; B \to bBd|C \;\; , \;\; C \to bCc|\lambda$$

مثال

- یک گرامر مستقل از متن برای متمم زبان $L = \{a^n b^n : n \geq ۰\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای متمم زبان $L = \{a^n b^n : n \geq ۰\}$ بیابید.
- $\overline{L} = \{a^m b^n : m < n\} \cup \{a^m b^n : m > n\} \cup L((a+b)^* b (a+b)^* a (a+b)^*)$
- پس سه حالت در نظر می‌گیریم: یا $m < n$ یا $m > n$ و یا در جملهٔ مورد نظر حداقل یک b قبل از a قرار دارد.
- $S \to S_۱ | S_۲ | S_۳$
  $S_۱ \to aS_۱b | aS_۱ | a$
  $S_۲ \to aS_۲b | S_۲b | b$
  $S_۳ \to TbTaT$
  $T \to aT | bT | \lambda$

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^m b^n c^p : m, n, p \geq ۰, n \neq m + p\}$ بیابید.

مثال

- یک گرامر مستقل از متن برای زبان $L = \{a^m b^n c^p : m, n, p \geq ۰, n \neq m + p\}$ بیابید.
- دو حالت در نظر می‌گیریم: $n < m + p$ یا $n > m + p$
- ابتدا گرامری برای زبان $L_۲ = \{a^m b^n c^p : m, n, p \geq ۰, n = m + p\}$ می‌نویسیم. سپس یا نمادهای a یا نمادهای c یا هر دو نماد a و c را افزایش می‌دهیم تا به دست آوریم $n < m + p$ یا نمادهای b را افزایش می‌دهیم تا به دست آوریم $n > m + p$
- همچنین زبان $L_۲$ را می‌توانیم به صورت $\{a^m b^m b^p c^p : m, p \geq ۰\}$ بازنویسی کنیم.

تجزیه

- اکنون به ازای یک رشتهٔ داده شده w می‌خواهیم بدانیم آیا اشتقاقی برای این رشته توسط قوانین تولید گرامر G وجود دارد یا خیر.
- به عبارت دیگر می‌خواهیم بدانیم آیا رشتهٔ w عضوی از زبان L(G) است یا خیر.
- پیدا کردن دنباله‌ای از قوانین تولید گرامر G که توسط آن رشتهٔ w به دست می‌آید (مشتق می‌شود) را تجزیه[1] می‌گوییم.

[1] parsing

تجزیه

- بدیهی‌ترین راه برای تجزیه، جستجو در تمام اشتقاق‌های ممکن است. اگر از یکی از اشتقاق‌ها در گرامر G جملهٔ w به دست بیاید می‌گوییم w عضوی از زبان L(G) است.

- به عبارت دیگر، اگر $S \stackrel{*}{\Rightarrow} w$ آنگاه $w \in L(G)$.

- این نوع تجزیه را تجزیهٔ جستجوی کامل [1] یا تجزیهٔ از بالا به پایین [2] می‌نامیم.

[1] exhaustive search parsing
[2] top-down parsing

تجزیه

- یک الگوریتم ساده برای تجزیهٔ از بالا به پایین چنین است:

۱. از متغیر آغازی S شروع می‌کنیم و تمام قوانین تولید را اعمال می‌کنیم. با اعمال این قوانین تعدادی صورت جمله‌ای و یا جمله به دست می‌آید.

۲. در صورتی که در این مرحله یکی از جمله‌های به دست آمده رشتهٔ w باشد، به نتیجه رسیده‌ایم در غیراینصورت تمام قوانین تولید ممکن بر روی یکی از متغیرهای صورت‌های جمله‌ای به دست آمده در مرحلهٔ قبل اعمال می‌شود (این اعمال قانون می‌تواند به عنوان مثال بر روی چپ‌ترین متغیر صورت بگیرد).

۳. این روند ادامه پیدا می‌کند تا جایی که یا رشتهٔ w از این گرامر مشتق شود و یا همهٔ حالت‌ها بررسی شده و رشتهٔ w به دست نیاید.

تجزیه

- گرامر $S \to SS|aSb|bSa|\lambda$ را در نظر بگیرید. یک اشتقاق برای رشتهٔ $w = aabb$ با استفاده از تجزیهٔ جستجوی کامل پیدا کنید.

تجزیه

- گرامر $S \to SS|aSb|bSa|\lambda$ را در نظر بگیرید. یک اشتقاق برای رشتهٔ $w = aabb$ با استفاده از تجزیهٔ جستجوی کامل پیدا کنید.
- در مرحلهٔ اول داریم $۱.\ S \Rightarrow SS,\ ۲.\ S \Rightarrow aSb,\ ۳.\ S \Rightarrow bSa,\ ۴.\ S \Rightarrow \lambda$ اما اشتقاق سوم و چهارم حذف می‌شود.
- در مرحلهٔ دوم داریم
$۱.\ S \Rightarrow SS \Rightarrow SSS,\ ۲.\ S \Rightarrow SS \Rightarrow aSbS,\ ۳.\ S \Rightarrow SS \Rightarrow bSaS,\ ۴.\ S \Rightarrow SS \Rightarrow S$
و همچنین
$۵.\ S \Rightarrow aSb \Rightarrow aSSb,\ ۶.\ S \Rightarrow aSb \Rightarrow aaSbb,\ ۷.\ S \Rightarrow aSb \Rightarrow abSab,\ ۸.\ S \Rightarrow aSb \Rightarrow ab$
- در مرحلهٔ سوم از اشتقاق ششم مرحلهٔ قبل به دست می‌آوریم: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

تجزیه

- تجزیهٔ جستجوی کامل چندین نقص دارد.
- اول اینکه از لحاظ زمان اجرا پرهزینه است زیرا تمام مسیرها جستجو می‌شوند. پس در جایی که به تجزیهٔ کارامد نیاز است نمی‌توان از آن استفاده کرد.
- دوم اینکه برای جمله‌هایی که عضوی از زبان آن گرامر نیستند، الگوریتم ممکن است هیچ‌گاه به پایان نرسد، مگر اینکه راهی برای توقف آن بیابیم.

- برای حل مشکل خاتمه‌ناپذیری الگوریتم تجزیهٔ جستجوی کامل، باید محدودیتی بر روی شکل قوانین اعمال کنیم.
- برای مثال اگر قوانینی که به شکل $A \rightarrow \lambda$ و $A \rightarrow B$ هستند را حذف کنیم، در این صورت طول صورت‌های جمله‌ای همیشه افزایش پیدا می‌کند و بنابراین وقتی طول همهٔ صورت‌های جمله‌ای از جملهٔ w بیشتر شد می‌توانیم الگوریتم را متوقف کنیم.

تجزیه

- برای مثال گرامر $S \rightarrow SS|aSb|bSa|ab|ba$ معادل گرامر $S \rightarrow SS|aSb|bSa|\lambda$ است و زبانی که هر دو تولید می‌کنند برابر است (با این تفاوت که با استفاده از گرامر اول رشتهٔ تهی تولید نمی‌شود).
- تفاوت این دو گرامر در هنگام تجزیه این است که اگر از تجزیهٔ جستجوی کامل در گرامر اول استفاده کنیم در هر مرحله طول صورت‌های جمله‌ای افزایش پیدا می‌کند بنابراین بعد از چندین مرحله معین می‌توانیم الگوریتم را به پایان برسانیم.

تجزیه

- قضیه: فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن است که هیچ یک از قوانین آن به شکل $A \to \lambda$ یا $A \to B$ نیستند، جایی که $A, B \in V$. آنگاه الگوریتم جستجوی کامل یا رشتهٔ مورد نظر w را تولید می‌کند و یا بعد از چندین مرحله متوقف می‌شود.

تجزیه

- قضیه: فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن است که هیچ یک از قوانین آن به شکل $A \rightarrow \lambda$ یا $A \rightarrow B$ نیستند، جایی که $A, B \in V$. آنگاه الگوریتم جستجوی کامل یا رشتهٔ مورد نظر w را تولید می‌کند و یا بعد از چندین مرحله متوقف می‌شود.

- اثبات: در هر مرحله از فرایند اشتقاق یا یکی از متغیرها با یک پایانه جایگزین می‌شود و یا طول صورت جمله‌ای حداقل یک واحد افزایش پیدا می‌کند. این افزایش طول یا توسط یک متغیر است و یا توسط یک پایانه. در بدترین حالت در $|w|$ مرحله طول صورت جمله‌ای توسط متغیرها افزایش می‌یابد و در $|w|$ مرحله هر متغیر با یک پایانه جایگزین می‌شود، بنابراین در بدترین حالت بعد از $۲|w|$ مرحله می‌توان تعیین کرد یک جمله عضو زبان این گرامر است یا خیر.

- اگر تعداد قوانین $|P|$ باشد، حداکثر $|P|^{۲|w|}$ حالت مختلف باید بررسی شوند تا بتوانیم اشتقاق جملهٔ w را به دست آوریم.

تجزیه

- بهترین الگوریتمی که برای تجزیهٔ یک گرامر مستقل از متن وجود دارد در $|w|^3$ گام می‌تواند جملهٔ w را تجزیه کند.
- الگوریتم‌های بهینه‌تری نیز برای تجزیه در حالات خاص وجود دارند که در نظریهٔ کامپایلرها به این الگوریتم‌ها پرداخته می‌شود و از حوزهٔ نظریهٔ زبان‌ها و ماشین‌ها خارج است.

- یک گرامر مستقل از متن $G = (V, T, S, P)$ یک گرامر ساده[1] گفته می‌شود اگر همهٔ قوانین آن به شکل $A \to ax$ باشد به طوری که $A \in V, a \in T, x \in V^*$ و هر جفت (A, a) حداکثر یک بار در قوانین تولید تکرار شده باشد.

[1] simple grammar or s-grammar

تجزیه

- گرامر $S \rightarrow aS|bSS|c$ یک گرامر ساده است.
- گرامر $S \rightarrow aS|bSS|aSS|c$ یک گرامر ساده نیست زیرا زوج (S, a) در دو قانون تکرار شده است.

تجزیه

- برای یک گرامر ساده، الگوریتم تجزیه توسط جستجوی کامل برای جملهٔ w به زمانی با پیچیدگی $|w|$ نیاز دارد.
- فرض کنید داشته باشیم $w = a_1 a_2 \cdots a_n$.
- از آنجایی که تنها یک قانون وجود دارد که در سمت چپش S و در سمت راستش با a_1 آغاز می‌شود، بنابراین می‌توانیم اشتقاق $S \Rightarrow a_1 A_1 A_2 \cdots A_m$ را در یک گام به دست آوریم.
- سپس باید متغیر A_1 را جایگزین کنیم : $S \overset{*}{\Rightarrow} a_1 a_2 B_1 B_2 \cdots A_2 \cdots A_m$
- بدین ترتیب در هر مرحله یک پایانه به دست می‌آید و در $|w|$ گام جملهٔ w مشتق می‌شود.

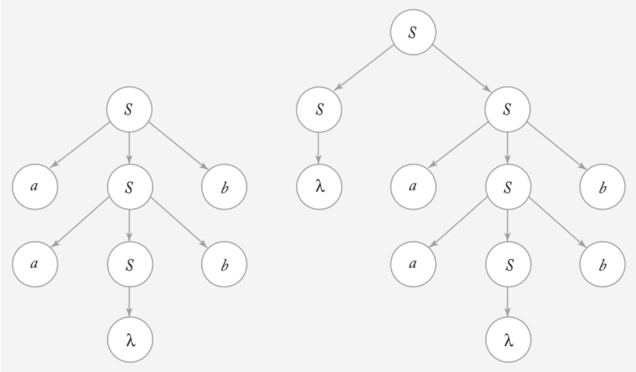
ابهام در گرامرها

- یک گرامر مستقل از متن مبهم [1] است اگر به ازای یک جمله $w \in L(G)$ حداقل دو درخت اشتقاق وجود داشته باشد. و یا به عبارت دیگر حداقل دو اشتقاق چپ یا راست برای یکی از جملات زبان آن گرامر وجود داشته باشد.

[1] ambiguous

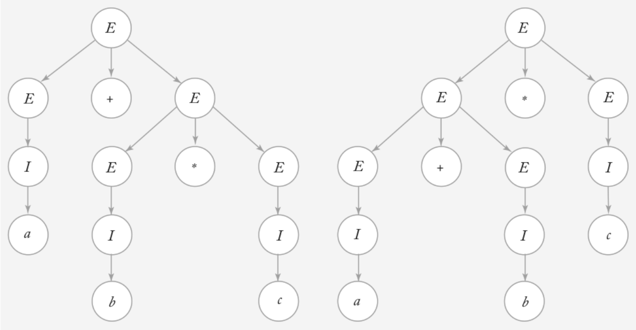
ابهام در گرامرها

- گرامر $S \rightarrow aSb|SS|\lambda$ مبهم است زیرا برای جملهٔ aabb دو درخت اشتقاق زیر وجود دارد.



ابهام در گرامرها

- گرامر $G = (V = \{E, I\}, T = \{a, b, c, +, *, (,)\}, E, P)$ را با قوانین تولید زیر در نظر بگیرید.
- $E \rightarrow I|E + E|E * E|(E) \quad , \quad I \rightarrow a|b|c$
- در این گرامر برای جملهٔ $a + b * c$ دو درخت اشتقاق وجود دارد.

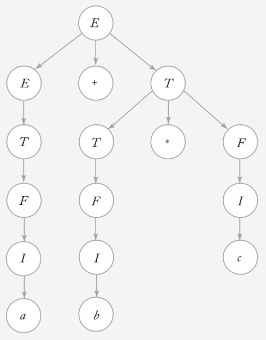


ابهام در گرامرها

- بنابراین گرامر قبل برای تولید جملات جبری با دو عملگر جمع و ضرب مبهم است.
- برای حل این مشکل، گرامر را به شکلی دیگر می‌نویسیم.

ابهام در گرامرها

- گرامر $G = (V = \{E, T, F, I\}, T = \{a, b, c, +, *, (,)\}, E, P)$ را با قوانین تولید زیر در نظر بگیرید.
- $E \to T|E + T \quad , \quad T \to F|T * F \quad , \quad F \to I|(E) \quad , \quad I \to a|b|c$
- در این گرامر برای جملهٔ $a + b * c$ فقط یک درخت اشتقاق وجود دارد.



ابهام در گرامرها

- گرچه توانستیم برای گرامری که جملات جبری تولید می‌کند یک گرامر غیرمبهم طراحی کنیم ولی الگوریتمی کلی برای تبدیل یک گرامر مبهم به یک گرامر غیرمبهم وجود ندارد.

ابهام در گرامرها

- اگر L یک زبان مستقل از متن باشد که برای آن یک گرامر غیرمبهم وجود داشته باشد، زبان L یک زبان غیرمبهم است.
- اگر هر گرامری که برای زبان L وجود دارد مبهم باشد، آنگاه زبان L ذاتا مبهم [1] است.
- اگر برای زبانی یک گرامر ساده وجود داشته باشد، آن زبان ذاتا مبهم نیست، چون با استفاده از گرامر ساده برای تجزیهٔ یک جمله تنها یک درخت می‌توان رسم کرد.

[1] inherently ambiguous

ابهام در گرامرها

- زبان $L = \{a^n b^n c^m : n, m \geq ۰\} \cup \{a^n b^m c^m : n, m \geq ۰\}$ ذاتا مبهم است.
- می‌توان گرامر $S \to S_۱|S_۲ \quad , \quad S_۱ \to S_۱c|A \quad , \quad A \to aAb|\lambda \quad , \quad S_۲ \to aS_۲|B \quad , \quad B \to bBc|\lambda$ را برای آن طراحی کرد.
- چرا این گرامر مبهم است؟

ابهام در گرامرها

- زبان $L = \{a^n b^n c^m : n, m \geq ۰\} \cup \{a^n b^m c^m : n, m \geq ۰\}$ ذاتا مبهم است.
- می‌توان گرامر $S \to S_۱ | S_۲ \quad , \quad S_۱ \to S_۱ c | A \quad , \quad A \to aAb | \lambda \quad , \quad S_۲ \to aS_۲ | B \quad , \quad B \to bBc | \lambda$ را برای آن طراحی کرد.
- این گرامر مبهم است، زیرا برای جملهٔ $a^n b^n c^n$ دو اشتقاق وجود دارد که یکی با $S \Rightarrow S_۱$ و دیگری با $S \Rightarrow S_۲$ آغاز می‌شود.
- اثبات این که این زبان ذاتا مبهم است اثباتی طولانی است که در مقاله‌ای در سال ۱۹۸۷ چاپ شده است.

ساده‌سازی گرامرهای مستقل از متن

- بررسی کردیم که تجزیهٔ جملات یک زبان توسط الگوریتم جستجوی کامل با استفاده از قوانین تولید گرامر مستقل از متن ممکن است خاتمه‌پذیر نباشد.

- برای حل مشکل خاتمه‌پذیری گفتیم می‌توانیم قوانینی را که تهی یا یک متغیر واحد تولید می‌کنند را حذف کنیم.

- در اینجا روشی برای حذف قوانین تولید تهی [1] (قانون تولید لامبدا یا اپسیلون) و همچنین حذف قوانین تولید یکه [2] (قانون تولید واحد یا تک‌متغیر) ارائه می‌کنیم.

[1] λ–productions
[2] unit–productions

ساده‌سازی گرامرهای مستقل از متن

- همچنین بررسی کردیم که الگوریتم جستجوی کامل (که یک الگوریتم حریصانه است) یک جمله را در زمانی از مرتبهٔ نمایی n به ازای جمله‌های با طول n تجزیه می‌کند.
- در اینجا در مورد دو فرم (شکل) نرمال [1] برای گرامرهای مستقل از متن، به نام فرم نرمال چامسکی [2] و فرم نرمال گریباخ [3] صحبت می‌کنیم.
- سپس الگوریتمی ارائه می‌کنیم که با استفاده از فرم نرمال چامسکی تجزیهٔ یک جمله را در زمان چندجمله‌ای (n^3) برای جمله‌های با طول n انجام می‌دهد.

---

[1] normal form
[2] Chomsky normal form
[3] Greibach normal form

ساده‌سازی گرامرهای مستقل از متن

- فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن است. فرض کنید P قانونی به شکل $A \rightarrow x_۱Bx_۲$ دارد.

- فرض کنید A و B دو متغیر متفاوتند و $B \rightarrow y_۱|y_۲|\cdots|y_n$ مجموعهٔ قوانین تولید در P است که در طرف چپ آنها متغیر B است.

- فرض کنید $\widehat{G} = (V, T, S, \widehat{P})$ گرامری است که $\widehat{P}$ به گونه‌ای ساخته شده است که در آن همهٔ قوانین $A \rightarrow x_۱Bx_۲$ در P با قوانین $A \rightarrow x_۱y_۱x_۲|x_۱y_۲x_۲|\cdots|x_۱y_nx_۲$ جایگزین شده‌اند.

- آنگاه داریم: $L(\widehat{G}) = L(G)$

- اثبات: به ازای هر $w \in L(G)$ اگر یک اشتقاق از w را در نظر بگیریم این اشتقاق توسط گرامر $\widehat{G}$ نیز امکان پذیر است و بالعکس برای هر $w \in L(\widehat{G})$ اشتقاق آن در G نیز امکان‌پذیر است.

ساده‌سازی گرامرهای مستقل از متن

- گرامر $G = (\{A, B\}, \{a, b, c\}, A, P)$ با قوانین تولید $A \rightarrow a|aaA|abBc \quad , \quad B \rightarrow abbA|b$ را در نظر بگیرید.
- می‌توانیم متغیر B را در این گرامر بدین صورت جایگزین کنیم: $A \rightarrow a|aaA|ababbAc|abbc$

ساده‌سازی گرامرهای مستقل از متن

- در صورتی که یک قانون تولید هیچ نقشی در فرایند اشتقاق برای هیچ جمله‌ای نداشته باشد، می‌توانیم آن قانون را حذف کنیم.
- برای مثال در گرامری با قوانین تولید $A \rightarrow aA$, $S \rightarrow aSb|\lambda|A$ قانون $S \rightarrow A$ منجر به تولید هیچ جمله‌ای نمی‌شود و می‌توان آن را حذف کرد.

ساده‌سازی گرامرهای مستقل از متن

- فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن باشد. متغیر $A \in V$ مفید است اگر و تنها اگر جملهٔ $w \in L(G)$ وجود دارد به طوری که $S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$ جایی که $x, y \in (V \cup T)^*$.

- به عبارت دیگر متغیر A مفید است اگر و تنها اگر در اشتقاق یکی از جملات حضور داشته باشد.

- اگر متغیری مفید نباشد آن را یک متغیر غیرمفید (بی‌استفاده) می‌خوانیم. می‌توانیم تمام قوانین تولیدی که شامل متغیرهای غیرمفید هستند را حذف کنیم.

ساده‌سازی گرامرهای مستقل از متن

- در گرامر زیر با متغیر آغازی S متغیر B غیر مفید است: $B \rightarrow bA \quad , \quad A \rightarrow aA|\lambda \quad , \quad S \rightarrow A$
- گرچه با شروع از متغیر B می‌توان رشته‌هایی تولید کرد اما با شروع از متغیر آغازی S هیچ رشته‌ای با استفاده از متغیر B تولید نمی‌شود و بنابراین متغیر B غیرمفید است.

ساده‌سازی گرامرهای مستقل از متن

- برای ساده سازی گرامر ابتدا متغیرهایی را که منجر به هیچ رشته‌ای نمی‌شوند (متغیرهای غیرسازنده) را حذف می‌کنیم.
- سپس متغیرهایی را که از متغیر آغازی قابل دسترسی نیستند (متغیرهای غیرقابل دسترس) حذف می‌کنیم.
- ترتیب حذف این متغیرها مهم است، زیرا بعد از حذف متغیرهای غیرسازنده ممکن است متغیرهای غیرقابل دسترس به وجود آید.
- برای مثال در قوانین تولید $S \to AB|a \ , \ A \to a \ , \ B \to Bc$ متغیر B غیرسازنده است. حذف متغیر B باعث حذف قانون $S \to AB$ می‌شود و بعد از این حذف، متغیر A غیرقابل دسترس می‌شود (گرچه در ابتدا متغیر A قابل دسترسی بود).

ساده‌سازی گرامرهای مستقل از متن

- در یک گرامر مستقل از متن به هر قانونی که به شکل $A \to \lambda$ باشد یک قانون تولید تهی[1] می‌گوییم.
- هر متغیری که از آن یک رشتهٔ تهی مشتق می‌شود، یعنی $A \overset{*}{\Rightarrow} \lambda$ یک متغیر تهی‌شدنی[2] می‌گوییم.
- یک گرامر ممکن است رشتهٔ تهی تولید نکند ولی شامل تعدادی قانون تولید تهی باشد. در چنین گرامری قوانین تولید تهی قابل حذف هستند.

[1] λ–production
[2] nullable

ساده‌سازی گرامرهای مستقل از متن

- برای مثال در گرامر $S \rightarrow aS_1b \quad , \quad S_1 \rightarrow aS_1b|\lambda$ می‌توان قانون تولید تهی را بدین صورت حذف کرد:
- $S \rightarrow aS_1b|ab \quad , \quad S_1 \rightarrow aS_1b|ab$

ساده‌سازی گرامرهای مستقل از متن

- فرض کنید G یک گرامر مستقل از متن برای زبان $L(G)$ باشد به طوری که رشتهٔ تهی در این زبان نباشد. در اینصورت گرامر $\widehat{G}$ وجود دارد که معادل گرامر G است و در آن قانون تولید تهی وجود ندارد.

- الگوریتم: ابتدا به ازای هر قانون تولید $A \to \lambda$ متغیر A را در مجموعهٔ V_N شامل همهٔ متغیرهای تهی‌شدنی اضافه می‌کنیم. سپس به ازای هر قانون تولید $A_۱ A_۲ \cdots A_n \to B$ که $A_۱, A_۲, \cdots A_n$ در مجموعه V_N قرار دارند، متغیر B را نیز به مجموعهٔ V_N اضافه می‌کنیم.

- سپس به ازای هر قانون $A \to x_۱ x_۲ \cdots x_m$ به طوری که $x_i \in V \cup T$ همهٔ حالت‌هایی را محاسبه می‌کنیم که یک یا تعدادی از متغیرهای تهی‌شدنی از طرف راست قانون حذف شوند. هر یک از این حالت‌ها را متغیر A ممکن است تولید کند، پس این حالت‌ها را به طرف راست قانون با علامت یا (خط عمودی |) اضافه می‌کنیم.

- برای مثال اگر دو متغیر x_i و x_j تهی‌شدنی باشند، آنگاه سه حالت وجود دارد: (۱) هر دو متغیر حذف شوند، (۲) متغیر x_i حذف شود، (۳) متغیر x_j حذف شود.

ساده‌سازی گرامرهای مستقل از متن

- یک گرامر مستقل از متن معادل گرامر زیر بدون قانون تهی پیدا کنید:
- $S \rightarrow ABaC \ , \ A \rightarrow BC \ , \ B \rightarrow b|\lambda \ , \ C \rightarrow D|\lambda \ , \ D \rightarrow d$

ساده‌سازی گرامرهای مستقل از متن

- یک گرامر مستقل از متن معادل گرامر زیر بدون قانون تهی پیدا کنید:
- $S \rightarrow ABaC \ , \ A \rightarrow BC \ , \ B \rightarrow b|\lambda \ , \ C \rightarrow D|\lambda \ , \ D \rightarrow d$
- ابتدا متغیرهای تهی‌شدنی را به مجموعهٔ V_N اضافه می‌کنیم. متغیرهای B و C و A تهی‌شدنی هستند.
- سپس قوانین تولید تهی را حذف می‌کنیم.
- در پایان همه حالت‌هایی را محاسبه می‌کنیم که یک یا چند متغیر تهی‌شدنی، در سمت راست قوانین حذف شوند. بنابراین داریم:

$S \rightarrow ABaC|BaC|AaC|ABa|aC|Aa|Ba|a \ , \ A \rightarrow B|C|BC \ , \ B \rightarrow b \ , \ C \rightarrow D \ , \ D \rightarrow d$

ساده‌سازی گرامرهای مستقل از متن

- بعد از حذف قوانین تولید تهی، قوانین تولید یکه را حذف می‌کنیم.

- در یک گرامر مستقل از متن قانون $A \rightarrow B$ به طوری که $A, B \in V$ باشند، قانون تولید یکه [1] نامیده می‌شود.

[1] unit-production

ساده‌سازی گرامرهای مستقل از متن

- فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن بدون قانون تولید تهی باشد. آنگاه یک گرامر مستقل از متن $\widehat{G} = (\widehat{V}, \widehat{T}, \widehat{S}, \widehat{P})$ وجود دارد که هیچ قانون تولید یکه ندارد.

- تمام قوانین تولید $A \to A$ می‌توانند بدون هیچ تأثیری در گرامر حذف شوند، پس فقط قوانین $A \to B$ را در نظر می‌گیریم، به طوری که $A \neq B$.

- سپس به ازای هر متغیر $A \in V$ متغیرهای $B \in V$ را محاسبه می‌کنیم به طوری که $A \overset{*}{\Rightarrow} B$.

- همهٔ قوانین تولید یکه به صورت $A \to B$ را حذف می‌کنیم.

- حال اگر داشته باشیم $B \to y_1|y_2|\cdots|y_n$ و $A \overset{*}{\Rightarrow} B$ آنگاه قوانین تولید $A \to y_1|y_2|\cdots|y_n$ را به قوانین تولید اضافه می‌کنیم.

ساده‌سازی گرامرهای مستقل از متن

- همهٔ قوانین تولید یکه را در گرامر $A \rightarrow a|bc|B \quad , \quad B \rightarrow A|bb \quad , \quad S \rightarrow Aa|B$ حذف کنید.

ساده‌سازی گرامرهای مستقل از متن

- همهٔ قوانین تولید یکه را در گرامر $A \rightarrow a|bc|B \quad , \quad B \rightarrow A|bb \quad , \quad S \rightarrow Aa|B$ حذف کنید.
- ابتدا به ازای هر متغیر در گرامر، همهٔ تک‌متغیرهایی را که تولید می‌کند را محاسبه می‌کنیم. در این گرامر داریم:
$$A \overset{*}{\Rightarrow} B, A \overset{*}{\Rightarrow} A, B \overset{*}{\Rightarrow} A, S \overset{*}{\Rightarrow} B, S$$
- سپس قوانین $A \rightarrow B, B \rightarrow A, A \rightarrow B, S$ را حذف و قوانین $A \rightarrow a|bc, B \rightarrow bb, S \rightarrow bb|a|bc$ را به قوانین اضافه می‌کنیم.
- در نهایت مجموعه قوانین $B \rightarrow a|bb|bc \quad , \quad A \rightarrow a|bb|bc \quad , \quad S \rightarrow a|bc|bb|Aa$ را به دست می‌آوریم.
- توجه کنید که بعد از حذف قوانین تولید یکه، متغیر B به یک متغیر غیرمفید تبدیل شد که می‌توان آن را حذف کرد.

ساده‌سازی گرامرهای مستقل از متن

- فرض کنید L یک زبان مستقل از متن باشد که شامل رشتهٔ تهی نباشد. آنگاه یک گرامر مستقل از متن برای آن وجود دارد که شامل هیچ قانون تولید غیرمفید، هیچ قانون تولید تهی، و هیچ قانون تولید یکه نشود.

- گرامر مستقل از متن G را برای زبان L در نظر می‌گیریم.

- (۱) ابتدا قوانین تولید تهی را با استفاده از الگوریتمی که اشاره شد حذف می‌کنیم. (۲) سپس قوانین تولید یکه را حذف می‌کنیم. (۳) در پایان قوانین تولید و متغیرهای غیرمفید (غیرسازنده و غیرقابل دسترس) را حذف می‌کنیم.

- توجه کنید که ترتیب حذف کردن مهم است، زیرا حذف قانون تولید تهی ممکن است قانون تولید یکه ایجاد کند اما حذف قانون تولید یکه، قانون تولید تهی ایجاد نمی‌کند. همچنین حذف قانون تولید یکه ممکن است قوانین غیرمفید ایجاد کند اما حذف قوانین غیرمفید، قوانین تولید تهی و یکه ایجاد نمی‌کند.

فرم نرمال

- برای گرامرهای مستقل از متن دو فرم نرمال وجود دارد که به طور گسترده‌ای استفاده شده‌اند:
- فرم نرمال چامسکی
- فرم نرمال گریباخ
- این فرم‌های نرمال برخی الگوریتم‌های تجزیه و اثبات‌ها را ساده می‌کنند و بدین دلیل به مطالعه آنها می‌پردازیم.

فرم نرمال چامسکی

- یک گرامر مستقل از متن به صورت فرم نرمال چامسکی است اگر همهٔ قوانین آن به صورت $A \to BC$ یا به صورت $A \to a$ باشند، به طوری که $A, B, C \in V$ و $a \in T$
- برای مثال گرامر $A \to SA|b \quad , \quad S \to AS|a$ به صورت فرم نرمال چامسکی است.

فرم نرمال چامسکی

- هر گرامر مستقل از متن $G = (V, T, S, P)$ به طوری که $\lambda \notin L(G)$ یک گرامر معادل به صورت فرم نرمال چامسکی دارد.

- اثبات: از آنجایی که در هر گرامر می‌توان قوانین تولید یکه و تهی و غیرمفید را حذف کرد، فرض می‌کنیم همهٔ این قوانین در G حذف شده‌اند. سپس گرامر $G_۱ = (V_۱, T, S, P_۱)$ را با استفاده از همهٔ قوانین تولید گرامر G که به صورت $A \to x_۱x_۲ \cdots x_n$ هستند می‌سازیم. در اینجا $x_i \in (V \cup T)$.

- اگر $n = ۱$ آنگاه $x_۱$ باید یک نماد پایانی باشد، زیرا در گرامر G هیچ قانون تولید یکه‌ای وجود ندارد.

- اگر $n \geq ۲$ باشد، آنگاه به ازای هر نماد پایانی $x_i = a \in T$ یک متغیر B_a در نظر می‌گیریم. بنابراین داریم $A \to C_۱C_۲ \cdots C_n$ به طوری که $C_i = x_i$ اگر $x_i \in V$ و $C_i = B_a$ اگر $x_i = a \in T$

- به ازای هر متغیر B_a قانون $B_a \to a$ را می‌سازیم.

فرم نرمال چامسکی

- سپس همهٔ قوانین به صورت $A \rightarrow C_1C_2$ را به گرامر G_1 اضافه می‌کنیم.
- به ازای هر قانون $A \rightarrow C_1C_2 \cdots C_n$ جایی که $n > 2$ است، متغیرهای $D_1, D_2, \cdots D_{n-2}$ را به متغیرهای گرامر G_1 اضافه می‌کنیم.
- سپس قوانین جدید به صورت فرم نرمال چامسکی را بدین صورت می‌سازیم:
$A \rightarrow C_1D_1 \quad , \quad , D_1 \rightarrow C_2D_2 \quad , \quad \cdots D_{n-2} \rightarrow C_{n-1}C_n$

فرم نرمال چامسکی

- گرامر G با قوانین تولید $S \to ABa \quad , \quad A \to aab \quad , \quad B \to Ac$ را به صورت فرم نرمال چامسکی در آورید.

فرم نرمال چامسکی

- گرامر G با قوانین تولید $S \rightarrow ABa \;\; , \;\; A \rightarrow aab \;\; , \;\; B \rightarrow Ac$ را به صورت فرم نرمال چامسکی در آورید.
- این گرامر قانون تولید تهی، قانون تولید یکه، و قانون غیرمفید ندارد.
- ابتدا نمادهای پایانی را با متغیر جایگزین می‌کنیم: $S \rightarrow ABB_a \;\; , \;\; A \rightarrow B_aB_aB_b$
  $B \rightarrow AB_c \;\; , \;\; B_a \rightarrow a \;\; , \;\; B_b \rightarrow b \;\; , \;\; B_c \rightarrow c$
- سپس گرامر را به صورت فرم نرمال چامسکی در می‌آوریم:
  $S \rightarrow AD_۱ \;\; , \;\; D_۱ \rightarrow BB_a \;\; , \;\; A \rightarrow B_aD_۲ \;\; , \;\; D_۲ \rightarrow B_aB_b$
  $B \rightarrow AB_c \;\; , \;\; B_a \rightarrow a \;\; , \;\; B_b \rightarrow b \;\; , \;\; B_c \rightarrow c$

فرم نرمال گریباخ

- یک گرامر مستقل از متن به صورت فرم نرمال گریباخ است اگر همهٔ قوانین تولید آن به صورت $A \to ax$ باشد به طوری که $a \in T$ و $x \in V^*$.

فرم نرمال گریباخ

- گرامر $S \to AB \quad , \quad A \to aA|bB|b \quad , \quad B \to b$ را به صورت فرم نرمال گریباخ درآورید.

فرم نرمال گریباخ

- گرامر $B \rightarrow b \ , \ A \rightarrow aA|bB|b \ , \ S \rightarrow AB$ را به صورت فرم نرمال گریباخ درآورید.
- قانون $S \rightarrow AB$ به صورت فرم نرمال گریباخ نیست و بنابراین می‌توانیم A را با مقادیر آن جایگزین کنیم.
- بنابراین داریم: $B \rightarrow b \ , \ A \rightarrow aA|bB|b \ , \ S \rightarrow aAB|bBB|bB$

فرم نرمال گریباخ

- گرامر $S \rightarrow abSb|aa$ را به صورت فرم نرمال گریباخ درآورید.
- می‌توانیم از تکنیک استفاده شده در تبدیل فرم نرمال چامسکی استفاده کنیم و تعدادی از نمادهای پایانی را با متغیرها جایگزین کنیم.
- بنابراین داریم: $S \rightarrow aBSB|aA \quad , \quad A \rightarrow a \quad , \quad B \rightarrow b$

الگوریتم سی‌وای‌کا

- الگوریتم سی‌وای‌کا [1] الگوریتمی است برای تجزیهٔ جملات با استفاده از یک گرامر مستقل از متن.
- فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل از متن در فرم نرمال چامسکی باشد و $w = w_1 w_2 \cdots w_n$ رشته‌ای باشد که می‌خواهیم با استفاده از این گرامر تجزیه کنیم.
- تعریف می‌کنیم $w_{ij} = w_i \cdots w_j$ و همچنین $X_{ij} = \{X \in V : X \overset{*}{\Rightarrow} w_{ij}\}$.
- مجموعهٔ X_{ij} در واقع مجموعهٔ همهٔ متغیرهایی است که می‌توانند زیر رشتهٔ w_{ij} را تولید کنند.
- بنابراین داریم $w \in L(G)$ اگر و فقط اگر $S \in X_{1n}$

[1] CYK (Cocke – Younger – Kasami) algorithm

الگوریتم سی‌وای‌کا

- می‌دانیم $X \in X_{ii}$ اگر و فقط اگر درگرامر G قانونی به صورت $X \to w_i$ وجود داشته باشد. به عبارت دیگر $X_{ii} = \{X : (X \to w_i) \in P\}$. بنابراین می‌توانیم X_{ii} را به ازای $1 \leq i \leq n$ محاسبه کنیم.

- همچنین می‌دانیم $w_{ij} \overset{*}{\Rightarrow} X$ اگر و فقط اگر قانون تولید $X \to YZ$ وجود داشته باشد به طوری که $Y \overset{*}{\Rightarrow} w_{ik}$ و $Z \overset{*}{\Rightarrow} w_{(k+1)j}$ به ازای $i \leq k < j$

- به عبارت دیگر داریم: $X_{ij} = \bigcup_{i \leq k < j} \{X : (X \to YZ) \in P, Y \in X_{ik}, Z \in X_{(k+1)j}\}$

- بنابراین با استفاده از برنامه‌نویسی پویا [1] می‌توانیم مقدار X_{1n} را با استفاده از مقادیر به دست آمده از زیر مسئله‌ها به دست آوریم.

- برای این کار ابتدا $X_{11}, X_{22}, \cdots, X_{nn}$ را محاسبه می‌کنیم، سپس $X_{12}, X_{23}, \cdots, X_{(n-1)n}$ و بعد از آن $X_{13}, X_{24}, \cdots, X_{(n-2)n}$ و همینطور الی آخر تا مقدار X_{1n} محاسبه شود.

[1] dynamic programming

الگوریتم سی‌وای‌کا

- برای محاسبهٔ مقادیر X_{ij}، می‌توانیم جدولی به صورت زیر (برای مثال وقتی طول جمله ۶ است) تهیه کنیم:

۶	$X_{۱۶}$					
۵	$X_{۱۵}$	$X_{۲۶}$				
۴	$X_{۱۴}$	$X_{۲۵}$	$X_{۳۶}$			
۳	$X_{۱۳}$	$X_{۲۴}$	$X_{۳۵}$	$X_{۴۶}$		
۲	$X_{۱۲}$	$X_{۲۳}$	$X_{۳۴}$	$X_{۴۵}$	$X_{۵۶}$	
۱	$X_{۱۱}$	$X_{۲۲}$	$X_{۳۳}$	$X_{۴۴}$	$X_{۵۵}$	$X_{۶۶}$
	$w_{۱}$	$w_{۲}$	$w_{۳}$	$w_{۴}$	$w_{۵}$	$w_{۶}$

الگوریتم سی‌وای‌کا

- در این جدول، مقادیر X_{ii} مستقیما از مقادیر w_i به دست می‌آیند. به عبارت دیگر $X_{ii} = \{X : (X \to w_i) \in P\}$.

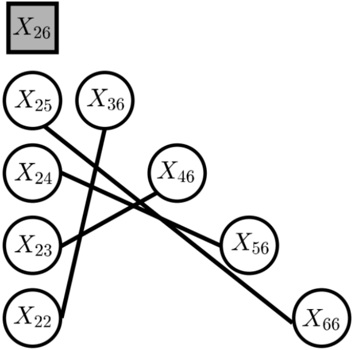
- همچنین مقادیر X_{ij} از اجتماع $j - i$ حالت به صورت $X_{ij} = \bigcup_{i \le k < j} \{X : (X \to YZ) \in P, Y \in X_{ik}, Z \in X_{(k+۱)j}\}$ به دست می‌آید که این جدول محاسبه این حالت‌ها را تسهیل می‌کند.

۶	$X_{۱۶}$					
۵	$X_{۱۵}$	$X_{۲۶}$				
۴	$X_{۱۴}$	$X_{۲۵}$	$X_{۳۶}$			
۳	$X_{۱۳}$	$X_{۲۴}$	$X_{۳۵}$	$X_{۴۶}$		
۲	$X_{۱۲}$	$X_{۲۳}$	$X_{۳۴}$	$X_{۴۵}$	$X_{۵۶}$	
۱	$X_{۱۱}$	$X_{۲۲}$	$X_{۳۳}$	$X_{۴۴}$	$X_{۵۵}$	$X_{۶۶}$
	$w_{۱}$	$w_{۲}$	$w_{۳}$	$w_{۴}$	$w_{۵}$	$w_{۶}$

الگوریتم سی‌وای‌کا

- برای مثال مقدار $X_{۲۶}$ را در این جدول، از اجتماع چهار حالت ممکن به دست می‌آوریم:

$$X_{۲۶} = \{X : (X \to YZ) \in P, Y \in X_{۲۲}, Z \in X_{۳۶}\} \cup \{X : (X \to YZ) \in P, Y \in X_{۲۳}, Z \in X_{۴۶}\}$$
$$\cup \{X : (X \to YZ) \in P, Y \in X_{۲۴}, Z \in X_{۵۶}\} \cup \{X : (X \to YZ) \in P, Y \in X_{۲۵}, Z \in X_{۶۶}\}$$



الگوریتم سی‌وای‌کا

- با استفاده از الگوریتم سی‌وای‌کا بررسی کنید که آیا رشتهٔ $\mathrm{w} = \mathrm{aabbb}$ متعلق به زبان تولید شده توسط گرامر $\mathrm{S} \rightarrow \mathrm{AB} \;\; , \;\; \mathrm{A} \rightarrow \mathrm{BB}|\mathrm{a} \;\; , \;\; \mathrm{B} \rightarrow \mathrm{AB}|\mathrm{b}$ است یا خیر.

الگوریتم سی‌وای‌کا

- با استفاده از الگوریتم سی‌وای‌کا بررسی کنید که آیا رشتهٔ $w = aabbb$ متعلق به زبان تولید شده توسط گرامر $S \to AB \;\; , \;\; A \to BB|a \;\; , \;\; B \to AB|b$ است یا خیر.
- از آنجایی که $w_{11} = a$ بنابراین X_{11} شامل متغیرهایی است که از آنها a مشتق می‌شود و همینطور الی آخر.
- بنابراین: $X_{11} = \{A\}, X_{22} = \{A\}, X_{33} = \{B\}, X_{44} = \{B\}, X_{55} = \{B\}$
- سپس X_{12} را محاسبه می‌کنیم. از آنجایی که $X_{12} = \{X : (X \to YZ) \in P, Y \in X_{11}, Z \in X_{22}\}$ و $X_{11} = X_{22} = \{A\}$ بنابراین $X_{12} = \emptyset$.
- سپس X_{23} را محاسبه می‌کنیم: $X_{23} = \{X : (X \to YZ) \in P, Y \in X_{22}, Z \in X_{33}\} = \{S, B\}$

الگوریتم سی‌وای‌کا

- اگر همهٔ مقادیر X_{ij} را محاسبه کنیم داریم:

$$
\begin{array}{c|lllll}
۵ & X_{۱۵}=\{S,B\} & & & & \\
۴ & X_{۱۴}=\{A\} & X_{۲۵}=\{S,B\} & & & \\
۳ & X_{۱۳}=\{S,B\} & X_{۲۴}=\{A\} & X_{۳۵}=\{S,B\} & & \\
۲ & X_{۱۲}=\emptyset & X_{۲۳}=\{S,B\} & X_{۳۴}=\{A\} & X_{۴۵}=\{A\} & \\
۱ & X_{۱۱}=\{A\} & X_{۲۲}=\{A\} & X_{۳۳}=\{B\} & X_{۴۴}=\{B\} & X_{۵۵}=\{B\} \\
\hline
 & a & a & b & b & b
\end{array}
$$

- بنابراین $S \in X_{۱۵}$ و در نتیجه $w \in L(G)$

الگوریتم سی‌وای‌کا

- پیچیدگی الگوریتم سی‌وای‌کا برابر است با $O(n^3)$ به طوری که n طول کلمهٔ w برای تجزیه است.
- برای محاسبه پیچیدگی الگوریتم تعداد همهٔ X_{ij} ها را می‌شماریم. به ازای هر X_{ij} باید $j-i$ مجموعه محاسبه شوند.
- تعداد X_{ii} ها برابر است با n و در این حالت مقادیر X_{ii} را مستقیما از w به دست می‌آوریم. تعداد $X_{i(i+1)}$ ها برابر است با $n-1$ و در این حالت $j-i$ برابر است با ۱ ، تعداد $X_{i(i+2)}$ ها برابر است با $n-2$ و در این حالت $j-i$ برابر است با ۲ ، و الی آخر. در نهایت تعداد X_{1n} برابر است با ۱ و در این حالت $j-i$ برابر است با $n-1$.

الگوریتم سی‌وای‌کا

- بنابراین پیچیدگی الگوریتم برابر است با:

$$
\begin{aligned}
n + (n-۱) \times ۱ + (n-۲) \times ۲ + \cdots + ۱ \times (n-۱) &= n + \Sigma_{k=۱}^{n-۱}(n-k) \times k \\
&= n + \Sigma_{k=۱}^{n}(n-k) \times k \\
&= n + n\Sigma_{k=۱}^{n}k - \Sigma_{k=۱}^{n}k^{۲} \\
&= n + n\frac{n(n+۱)}{۲} - \frac{n(n+۱)(۲n+۱)}{۶} \\
&= n + \frac{n(n+۱)(n-۱)}{۶} \\
&= \frac{n^{۳}+۵n}{۶} = O(n^{۳})
\end{aligned}
$$

الگوریتم سی‌وای‌کا

- جملهٔ bbabb را توسط الگوریتم سی‌وای‌کا به ازای گرامر زیر تجزیه کنید.

$$S \to AB|AC|AA$$
$$A \to CB|a \quad , \quad B \to AC|b \quad , \quad C \to CC|b$$

الگوریتم سی‌وای‌کا

- جملهٔ bbabb را توسط الگوریتم سی‌وای‌کا به ازای گرامر زیر تجزیه کنید.

$$S \rightarrow AB|AC|AA$$
$$A \rightarrow CB|a \;\;,\;\; B \rightarrow AC|b \;\;,\;\; C \rightarrow CC|b$$

5	$\{S, A, B\}$				
4	$\{S, A\}$	$\{S, A, B\}$			
3	$\{S\}$	$\{A\}$	$\{S, B\}$		
2	$\{A, C\}$	$\emptyset$	$\{S, B\}$	$\{A, C\}$	
1	$\{B, C\}$	$\{B, C\}$	$\{A\}$	$\{B, C\}$	$\{B, C\}$
	1	2	3	4	5
	b	b	a	b	b