

گزارش پروژه درس نظریه زبان‌ها و ماشین‌ها

ساده‌سازی گرامرهای context-free

دانشگاه صنعتی امیرکبیر، دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پارسا اسکندرئزاد 9531003

فهرست

1	مقدمه
2	پیاده‌سازی
2	حذف قوانین ۸: 1
2	حذف قوانین یک: 2
2	حذف قوانین بی‌فایده: 3
3	صحت عملکرد
3	حذف قوانین ۸: 1
3	مثال شماره 6.4 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م
4	مثال شماره 6.5 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م
5	حذف قوانین یک: 2
5	مثال شماره 6.6 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م
6	سوال مطرح شده در سایت پرسش و پاسخ <i>GATE Overflow</i>
7	حذف قوانین بی‌فایده: 3
7	مثال شماره 6.3 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م
8	مثال موجود در صفحه ویکیپدیا https://en.wikipedia.org/wiki/Useless_rules
9	تمرین بدون پاسخ شماره 8 فصل 6 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م
10	مثال lecture note درس CS3100 - Models of Computation دانشگاه Utah
11	کارهای آینده
11	منابع و مراجع

مقدمه

هدف این پروژه پیاده‌سازی الگوریتم‌های ساده‌سازی CFG¹ها می‌باشد. به منظور رسیدن به این هدف سه گام اساسی وجود دارد که به این شرح است:

1. حذف قوانین λ : یعنی قوانینی که به این صورت هستند $(A \rightarrow \lambda)$ به صورت مناسب حذف شوند.

2. حذف قوانین یک‌ه: یعنی قوانینی که به این صورت هستند $(A \rightarrow B)$ به صورت مناسب حذف شوند.

3. حذف قوانین بی‌فایده: یعنی قوانینی که منجر به هیچ رشته‌ای نمی‌شوند و یا از متغیر شروع دسترسی به آن‌ها امکان‌پذیر نیست به همراه قوانینشان حذف شوند.

توجه: برای دستیابی به گرامر ساده‌شده معادل با گرامر اولیه، مراحل مذکور باید به ترتیبی که گفته شده است اجرا شوند.

توجه: در ادامه تا پایان این گزارش فرض می‌شود که گرامر داده شده λ -free است و رشته خالی در گرامر داده شده نیست. توجه شود که اگر هم این فرض برقرار نبود به راحتی متغیر شروع را عوض کرده (مثلاً $\$$) و قانون زیر را اضافه می‌کنیم:

$$\$ \rightarrow S|\lambda$$

در پیاده‌سازی این پروژه این نکته نیز در نظر گرفته شده است.

¹ Context-free grammar

پیاده‌سازی

1. حذف قوانین λ :

برای حذف این قوانین ابتدا همه متغیرهایی که منجر به λ می‌شوند (یا به اصطلاح nullable هستند) را پیدا می‌کنیم. سپس قوانینی که سمت چپ آن‌ها λ وجود ندارد را اضافه می‌کنیم. حالا برای هر کدام از قوانین اضافه شده، تمام ترکیبات آن‌ها که متغیرهای nullable با λ جایگزین بشوند را اضافه می‌کنیم. فقط باید توجه کنیم که قوانین $A \rightarrow \lambda$ اضافه نشوند.

برای پیدا کردن همه ترکیبات مذکور از اعداد باینری استفاده می‌کنیم. مثلاً فرض کنید متغیرهای A و B منجر به λ می‌شوند و سمت چپ یک قانون به شکل $aAbB$ داشته باشیم، ترکیبات گفته شده به این شرح می‌باشند:

00	نیاید اضافه شود.
01	abB
10	aAb
11	aAbB

2. حذف قوانین یک‌ه:

به این منظور ابتدا گراف متغیرهای گرامر را می‌کشیم. گراف را به این صورت می‌سازیم که اگر قانونی به شکل $A \rightarrow x_1 B x_2$ داشتیم یالی از A به B خواهیم داشت. سپس همه قوانین غیر یک‌ه را اضافه کرده و برای قوانین یک‌ه اگر از متغیر سمت چپ به متغیر سمت راست راهی در گراف داشتیم (یا به طور معادل $A \Rightarrow^* B$) قوانین جدیدی که سمت چپ آن‌ها A و سمت راست آن‌ها مربوط به قوانین اضافه شده‌ای است که B سمت راست آن‌هاست را اضافه می‌کنیم.

برای یافتن راه ذکر شده در گراف از DFS^2 استفاده می‌کنیم و اگر نود سمت چپ بازدید شده بود یعنی راه وجود دارد.

3. حذف قوانین بی‌فایده:

این مرحله شامل دو قسمت است، ابتدا متغیرهایی که هیچ وقت رشته تولید نمی‌کنند را حذف می‌کنیم و سپس همانند قبل با کشیدن گراف متغیرها و اجرای DFS ، گرامر را از قوانینی که از متغیر شروع دسترسی‌پذیر نیستند پاک می‌کنیم.

² Depth-first search

صحت عملکرد

به منظور بررسی درستی عملکرد الگوریتم‌های ذکر شده ابتدا هر کدام از آن‌ها را به صورت جدا بررسی می‌کنیم. برای اطمینان از درستی جواب داده شده سعی شده است از همه‌ی مثال‌های موجود در کتاب مرجع درس و همچنین مثال‌های پیدا شده در اینترنت استفاده شود.

1. حذف قوانین λ :

مثال شماره 6.4 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م

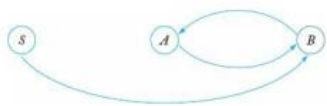
	ورودی	خروجی
برنامه	$S \rightarrow aZb$ $Z \rightarrow aZb \mid \wedge$	$S \rightarrow aZb \mid ab$ $Z \rightarrow aZb \mid ab$
کتاب	<p>Example 6.4</p> <p>Consider the grammar</p> $S \rightarrow aS_1b,$ $S_1 \rightarrow aS_1b \mid \lambda,$ <p>with start variable S. This grammar generates the λ-free language $\{a^n b^n : n \geq 1\}$. The λ-production $S_1 \rightarrow \lambda$ can be removed after adding new productions obtained by substituting λ for S_1 where it occurs on the right. Doing this we get the grammar</p> $S \rightarrow aS_1b \mid ab,$ $S_1 \rightarrow aS_1b \mid ab.$	

مثال شماره 6.5 کتاب نظریه زبان ها و ماشین ها، پیتر لینز، ویرایش 5م

	ورودی	خروجی
برنامه	$\begin{aligned} S &\rightarrow ABaC \\ A &\rightarrow BC \\ B &\rightarrow b ^{\wedge} \\ C &\rightarrow D ^{\wedge} \\ D &\rightarrow d \end{aligned}$	$\begin{aligned} S &\rightarrow ABaC Aa ABa a aC AaC BaC Ba \\ A &\rightarrow BC B C \\ B &\rightarrow b \\ C &\rightarrow D \\ D &\rightarrow d \end{aligned}$
کتاب	<p>Example 6.5</p> <p>Find a context-free grammar without λ-productions equivalent to the grammar defined by</p> $\begin{aligned} S &\rightarrow ABaC, \\ A &\rightarrow BC, \\ B &\rightarrow b ^{\wedge}, \\ C &\rightarrow D ^{\wedge}, \\ D &\rightarrow d. \end{aligned}$ <p>From the first step of the construction in Theorem 6.3, we find that the nullable variables are A, B, C. Then, following the second step of the construction, we get</p> $\begin{aligned} S &\rightarrow ABaC BaC AaC ABa aC Aa Ba a, \\ A &\rightarrow B C BC, \\ B &\rightarrow b, \\ C &\rightarrow D, \\ D &\rightarrow d. \end{aligned}$	

2. حذف قوانین یکه:

مثال شماره 6.6 کتاب نظریه زبان ها و ماشین ها، پیتر لینز، ویرایش 5م

	ورودی	خروجی
برنامه	$\begin{array}{l} S \rightarrow Aa \mid B \\ B \rightarrow A \mid bb \\ A \rightarrow a \mid bc \mid B \end{array}$	$\begin{array}{l} S \rightarrow Aa \mid a \mid bc \mid bb \\ A \rightarrow a \mid bc \mid bb \\ B \rightarrow bb \mid a \mid bc \end{array}$
کتاب	<p>Example 6.6</p> <p>Remove all unit-productions from</p> $\begin{array}{l} S \rightarrow Aa \mid B, \\ B \rightarrow A \mid bb, \\ A \rightarrow a \mid bc \mid B. \end{array}$ <p>The dependency graph for the unit-productions is given in Figure 6.3; we see from it that $S \xRightarrow{*} A$, $S \xRightarrow{*} B$, $B \xRightarrow{*} A$, and $A \xRightarrow{*} B$. Hence, we add to the original non-unit productions</p> $\begin{array}{l} S \rightarrow Aa, \\ A \rightarrow a \mid bc, \\ B \rightarrow bb, \end{array}$ <p>Figure 6.3</p>  <p>the new rules</p> $\begin{array}{l} S \rightarrow a \mid bc \mid bb, \\ A \rightarrow bb, \\ B \rightarrow a \mid bc, \end{array}$ <p>to obtain the equivalent grammar</p> $\begin{array}{l} S \rightarrow a \mid bc \mid bb \mid Aa, \\ A \rightarrow a \mid bb \mid bc, \\ B \rightarrow a \mid bb \mid bc. \end{array}$ <p>Note that the removal of the unit-productions has made B and the associated productions useless.</p>	

سوال مطرح شده در سایت پرسش و پاسخ GATE Overflow

	ورودی	خروجی
برنامه	<pre> Z -> S S -> ASA aB a SA AS S A -> B S B -> b </pre>	<pre> Z->ASA aB a SA AS S->ASA aB a SA AS A->ASA aB a SA AS b B->b </pre>
سوال و جواب	<div> <div> 0 votes </div> <div>187 views</div> </div> <div> <p>Remove unit productions from the following CFG:</p> <pre> So --> S S --> ASA aB a SA AS S A-->B S B --> b </pre> <p>S--->S is trivial, you can directly eliminate it at first.</p> <p>resulting unit production free CFG will be,</p> <pre> So-> ASA aB a SA AS S --> ASA aB a SA AS A-->b ASA aB a SA AS B --> b </pre> <p>commented Sep 5, 2017 by Joshi_nitish Boss</p> <hr/> <p> @joshi_nitish is correct!!</p> <p>commented Sep 5, 2017 by Shubhanshu Boss</p> </div>	

3. حذف قوانین بی فایده:

مثال شماره 6.3 کتاب نظریه زبان ها و ماشین ها، پیتز لینز، ویرایش 5م

	ورودی	خروجی
برنامه	$ \begin{aligned} S &\rightarrow aS A C \\ A &\rightarrow a \\ B &\rightarrow aa \\ C &\rightarrow aCb \end{aligned} $	$ \begin{aligned} S &\rightarrow aS A \\ A &\rightarrow a \end{aligned} $
کتاب	<p>Example 6.3</p> <p>Eliminate useless symbols and productions from $G = (V, T, S, P)$, where $V = \{S, A, B, C\}$ and $T = \{a, b\}$, with P consisting of</p> $ \begin{aligned} S &\rightarrow aS A C, \\ A &\rightarrow a, \\ B &\rightarrow aa, \\ C &\rightarrow aCb. \end{aligned} $ <p>First, we identify the set of variables that can lead to a terminal string. Because $A \rightarrow a$ and $B \rightarrow aa$, the variables A and B belong to this set. So does S, because $S \Rightarrow A \Rightarrow a$. However, this argument cannot be made for C, thus identifying it as useless. Removing C and its corresponding productions, we are led to the grammar G_1 with variables $V_1 = \{S, A, B\}$, terminals $T = \{a\}$, and productions</p> $ \begin{aligned} S &\rightarrow aS A, \\ A &\rightarrow a, \\ B &\rightarrow aa. \end{aligned} $ <p>Next we want to eliminate the variables that cannot be reached from the start variable. For this, we can draw a dependency graph for the variables. Dependency graphs are a way of visualizing complex relationships and are found in many applications. For context-free grammars, a dependency graph has its vertices labeled with variables, with an edge between vertices C and D if and only if there is a production of the form</p> $C \rightarrow xDy.$ <p>A dependency graph for V_1 is shown in Figure 6.1. A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable. In our case, Figure 6.1 shows that B is useless. Removing it and the affected productions and terminals, we are led to the final answer</p> $G = (\hat{V}, \hat{T}, S, \hat{P}) \text{ with } \hat{V} = \{S, A\}, \hat{T} = \{a\}, \text{ and productions}$ $ \begin{aligned} S &\rightarrow aS A, \\ A &\rightarrow a. \end{aligned} $	

مثال موجود در صفحه ویکیپدیا https://en.wikipedia.org/wiki/Useless_rules

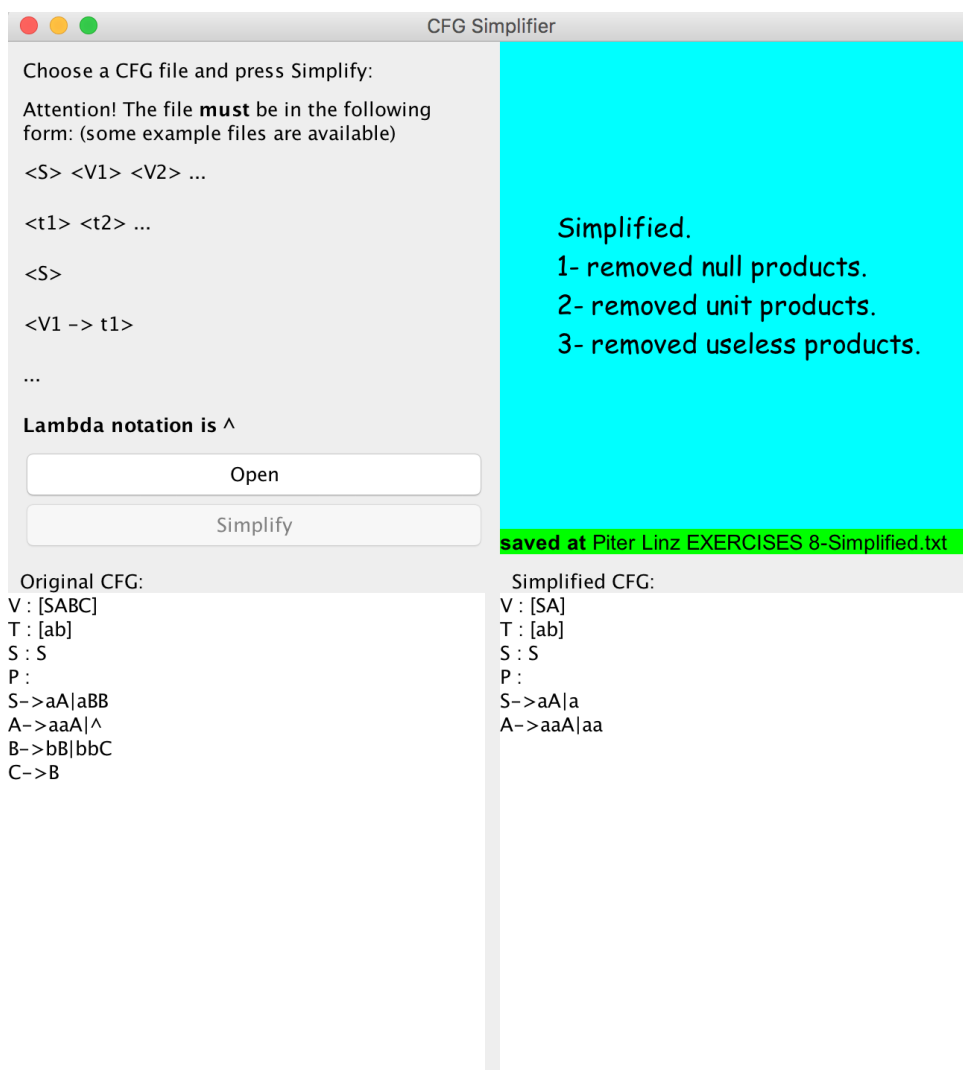
	ورودی	خروجی
برنامه	$ \begin{aligned} S &\rightarrow Bb \mid Cc \mid Ee \\ B &\rightarrow Bb \mid b \\ C &\rightarrow Cc \mid c \\ D &\rightarrow Bd \mid Cd \mid d \\ E &\rightarrow Ee \end{aligned} $	$ \begin{aligned} S &\rightarrow Bb \mid Cc \\ B &\rightarrow Bb \mid b \\ C &\rightarrow Cc \mid c \end{aligned} $
ویکیپدیا	<p>Examples [edit]</p> <p>Denoting nonterminal and terminal symbols by upper- and lower-case letters, respectively, in the following regular grammar with start symbol S</p> $ \begin{aligned} S &\rightarrow Bb \mid Cc \mid Ee \\ B &\rightarrow Bb \mid b \\ C &\rightarrow Cc \mid c \\ D &\rightarrow Bd \mid Cd \mid d \\ E &\rightarrow Ee \end{aligned} $ <p>the nonterminal D is unreachable, and E is unproductive. Hence, omitting the last two rules doesn't change the language accepted by the grammar, nor does omitting the alternative "Ee" from the right-hand side of the rule for S.</p>	

حالا درستی برنامه را برای اجرای هر سه الگوریتم به صورت پشت سر هم بررسی می‌کنیم.

همچنین اسکرین شات از اجرای برنامه نیز بعد از جدول‌ها آورده شده است:

تمرین بدون پاسخ شماره 8 فصل 6 کتاب نظریه زبان‌ها و ماشین‌ها، پیتز لینز، ویرایش 5م

	ورودی	خروجی
برنامه	<pre> S -> aA aBB A -> aaA ^ B -> bB bbC C -> B </pre>	<pre> S -> aA a A -> aaA aa </pre>



کارهای آینده

به نظر می‌رسد موارد زیر برای توسعه و بهتر کردن این پروژه در آینده مناسب باشند:

- اضافه کردن قابلیت نمایش گرامر به فرم‌های نرمال
- توانایی اضافه کردن قانون به گرامر از طریق GUI
- توانایی تعریف علامت دلخواه برای λ از طریق خود برنامه
- توانایی انجام الگوریتم‌های ساده‌سازی به صورت جدا از طریق GUI
- و ...

منابع و مراجع

- 1- An Introduction to Formal Languages and Automata 5th– 2011 by Peter Linz
- 2- Applied Automata Theory and Logic, Authors: Gopalakrishnan, Ganesh
- 3- https://en.wikipedia.org/wiki/Useless_rules
- 4- ...و