پروژه نظریه زبانها و ماشین ها (بخش دوم)

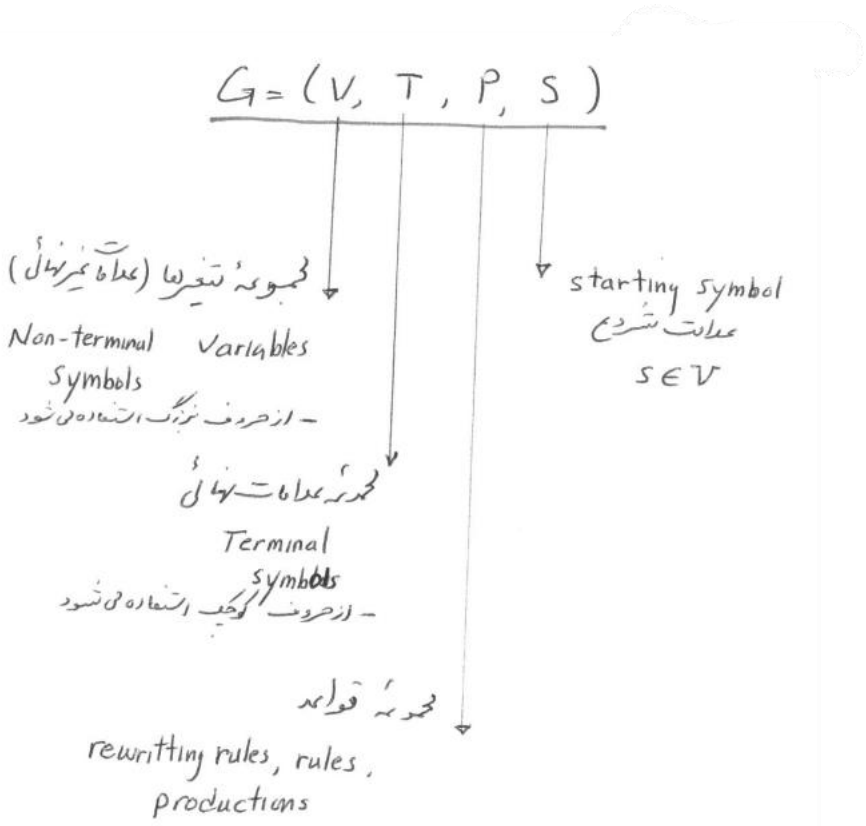
پیاده سازی برنامه تشخیص گرامر

علی فدائی منش – ۹۸۳۱۱۳۰

هدف: طراحی یک برنامه که یک گرامر به عنوان ورودی بگیرد و منظم یا مستقل از متن بودن یا هیچ کدام را مشخص کند بعلاوه ماشین متناهی یا پشته ای بعلاوه فرم نرمال گریباخ را تولید و کند و نمایش دهد.

مفروضات:

یک گرامر به طور کلی از تعدادی عضو تشکیل می شود. (V,T,P,S) که من S (کاراکتر شروع) را کاراکتر 'S' در نظر می گیرم. بعلاوه V را زیرمجموعه ای از حروف بزرگ انگلیسی در نظر می گیرم. همینطور T را زیرمجموعه ای از حروف کوچک انگلیسی در نظر می گیرم. و ورودی برنامه صرفا تعدادی قائده (Rule) از همین الفباها می باشد.



سوال: آیا گرامر ورودی منظم است؟

برای پاسخ به این سوال ابتدا نوع پیاده سازی ام به همراه جزئیات را توضیح خواهم داد. من یک کلاس Grammer درست کردم که کلاس FormalGrammer از آن ارث بری می کند. کلاس FormalGrammer برای شروع یک آرگومان ورودی می گیرد و آن **لیست قوانین گرامری** است که می خواهیم. سپس در constructor اش یک فیلد با نام isFormal اضافه می کند که متود checkFormal را صدا می کند. این متود قوانین (rules) داده شده را برسی می کند که خطی راست یا خطی چپ باشند و به عنوان خروجی نوع گرامر که 'right' یا 'left' است را برمی گرداند. پس تا اینجا گرامر را دریافت کردیم و برسی کردیم که منظم هست یا خیر. بنابراین با دسترسی به فیلد isFormal می توان منظم بودن یا نبودن گرامر را متوجه شد.

نمونه ورودی:

```
rules = [('B', 'aB'), ('S', 'aB'), ('B', 'b')]
```

نمونه خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
right
```

نمونه ورودی:

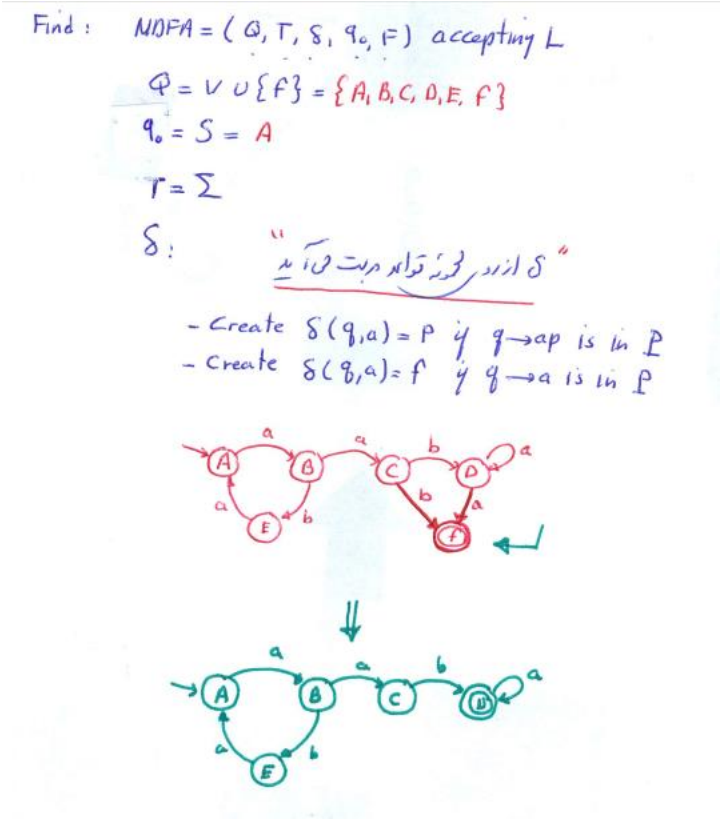
```
rules = [('B', 'AB'), ('S', 'aB'), ('B', 'b')]
```

نمونه خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
False
```

سوال: ماشین متناهی گرامر چیست؟

از گرامر می توان با نگاه کردن به لیست قوانین به توابع انتقال ماشین متناهی آن دست یافت. لذا طبق قضیه زیر عمل کردم و متود printMachine رو پیاده سازی کردم.



برای تولید ماشین های متناهی یک کلاس با نام FiniteMachine طراحی کردم. که prototype آن به شکل زیر است:

```
class FiniteMachine:
    def __init__(self, transitions) -> None: ···

    def printMachine(self): ···

    def accepts(self, string, startIndex, initialState):
```

که در واقع Transition ها به صورت یک لیست از tuple های سه تایی (tempState, string, nextState) دریافت می کند و از آن لیست state ها و character ها را استخراج می کند.

متود printMachine همه property های ماشین را چاپ می کند.

نمونه ورودی:

```
rules = [('B', 'AB'), ('S', 'aB'), ('B', 'b')]
```

نمونه خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
Starting state(q0):  S
states(Q):  {'S', 'F', 'B'}
transitions(T):  ['δ(B, a) => B', 'δ(S, a) => B', 'δ(B, b) => F']
AllowedCharacters(Σ):  {'b', 'a'}
FinalState:  F
PS C:\Users\afada\Desktop\AutomataGrammer>
```

متود آخر نیز accepts هست که در قسمت بعد توضیح می دهم.

سوال: متودی طراحی کنید که رشته ای را بگیرد و بگوید توسط این گرامر قابل قبول است یا خیر؟

برای حل این سوال یک الگوریتم بازگشتی طراحی کردم. این الگوریتم در واقع با دانستن توابع انتقال و state فعلی و نشانگر نوار (tapeIndex) کار می کند و در نهایت True یا False برمیگرداند.

```
def accepts(self, string, startIndex, initialState):
        if(initialState == 'F'):
            return True
        possibleTransitions = list(
            filter(lambda trans: trans[0] == initialState,
self.transitions))
        for trans in possibleTransitions:
            # if the transition fits in current state and couuld be
applied
            word = trans[1]  # the transition string
            nextIndex = startIndex + len(word)
            if string[startIndex:nextIndex] == word:
                if self.accepts(string, nextIndex, trans[2]):
                    return True
        return False
```

مثال ورودی:

```
RULES = [('B', 'aB'), ('S', 'aB'), ('B', 'b')]
STRING = 'assab'
```

مثال خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
False
```

مثال ورودی:

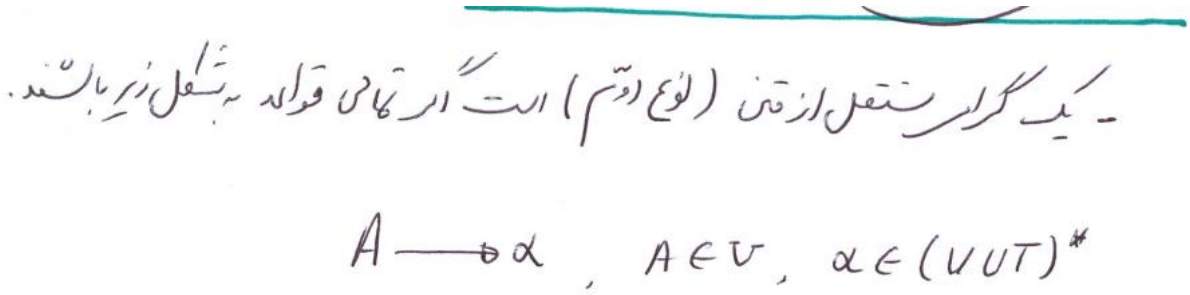
```
RULES = [('B', 'aB'), ('S', 'aB'), ('B', 'b')]
STRING = 'aaaaab'
```

مثال خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
True
```

سوال: آیا زبان ورودی مستقل از متن است؟

طبق تعریف زیر عمل می کنیم:



یک کلاس ContextFreeGrammer تعریف کردم که به عنوان ورودی rule ها را می گیرد و سپس مشخص می کند که ContextFree هست یا خیر.

```python
class ContextFreeGrammer(Grammer):
    def __init__(self, rules) -> None:
        super().__init__(rules)
        self.isContextFree = self.checkContextFree()

    def checkContextFree(self):
        for rule in self.rules:
            if len(rule[0]) != 1:
                return False
        return True
```

مثال ورودی:

```python
RULES = [('B', 'aB'), ('SA', 'aB'), ('B', 'b')]
```

مثال خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
False
```

مثال ورودی:

```python
RULES = [('B', 'aBaabahd'), ('A', 'aBasdBA'), ('B', 'b')]
```

مثال خروجی:

```
PS C:\Users\afada\Desktop\AutomataGrammer> python grammer.py
True
```

سوال: فرم نرمال گریباخ گرامر را پیدا کنید:

طبق تعریف داریم:

- فرم نرمال گریباخ (Greibach Normal Form)

- یک گرامر نوع ۲ (مستقل از متن) فرم نرمال گریباخ می باشد اگر قواعد شکل زیر باشند.

$$A \longrightarrow aX \quad , \quad a \in T, \; X \in V^*$$

زیر سوال: چگونه یک گرامر مستقل از متن را به فرم گریباخ در بیاوریم؟

۱. مرحله اول: تبدیل قواعد به فرم نرمال چامسکی
۲. مرحله دوم: حذف قواعد بازگشتی چپ
۳. مرحله سوم: تبدیل قواعد باقی مانده به فرم گریباخ

زیر سوال: فرم نرمال چامسکی چیست و چگونه به آن فرم تبدیل کنیم؟

- فرم نرمال چامسکی

- یک گرامر نوع ۲ فرم نرمال چامسکی می باشد اگر قواعد شکل زیر باشند.

$$A \longrightarrow BC \qquad A, B, C \in V, \; a \in T$$
$$A \longrightarrow a$$

مثال:

$$S \longrightarrow AA \mid BA \mid a$$
$$A \longrightarrow AA \mid a$$
$$B \longrightarrow BA \mid a$$

الگوریتم تبدیل به چامسکی:

۱. مرحله اول: حذف کاراکتر شروع
۲. مرحله دوم: حذف قوائد پوچ، واحد و بلااستفاده
۳. مرحله سوم: حذف قوائدی که در آن Terminal در کنار Terminal یا Variable قرار می گیرد.
۴. مرحله چهارم: حذف قوائدی که در آن بیش از دو Variable وجود دارند.