به نام خدا

دانشگاه اصفهان

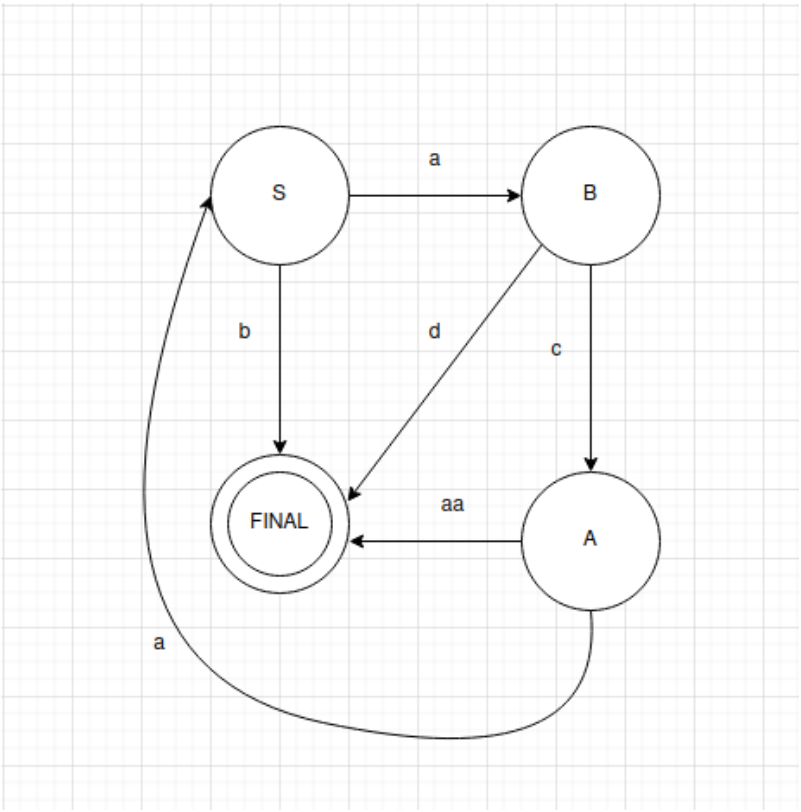
پروژه برنامه نویسی نظریه زبان‌ها و ماشین‌ها

امیر ارشیا همت
علیرضا احمدی طباطبایی

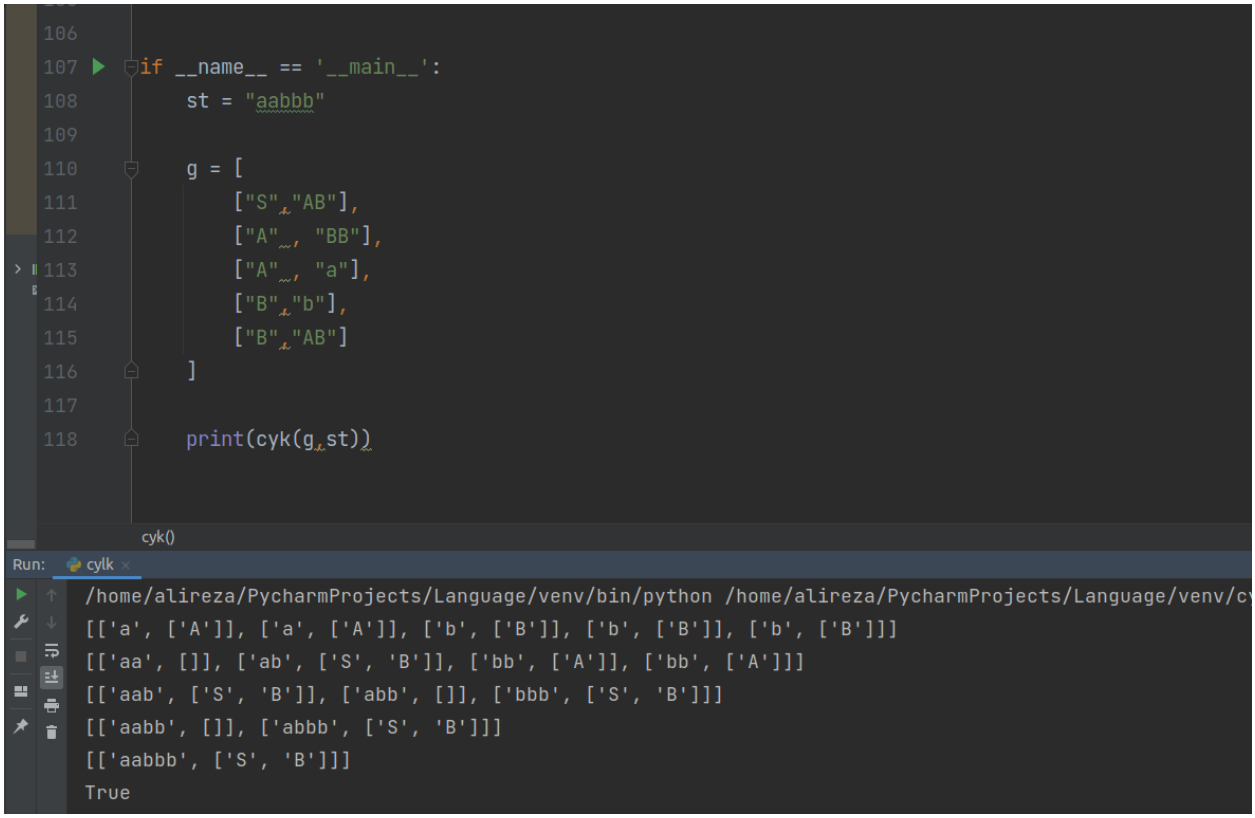
خرداد ۱۴۰۰

برای پیاده سازی بخش اول پروژه (تعین عبارت منظم بر اساس گرامر) ما ابتدا NFA گرامر داده شده را رسم کرده و براساس آن عبارت منظم را پیدا می کردیم.
برای تشخیص دور که به * در عبارت منظم منجر می شود از الگوریتم BFS استفاده کردیم و چنان چه چند حالت مختلف داشت از + استفاده می کنیم ، البته جزییات بیشتر در کد قابل مشاهده است .
مثلا چنانچه `['S" , "aB"],["S","b"],["A","aS"],["A",'aa'],["B","cA"],['B','d"]`
به عنوان گرامر داده شود به کمک NFA آن که در شکل زیر رسم شده به عبارت منظم زیر میرسیم : (aca)*b + (ac(aac)*aa + a(caa)*d

کد مربوط به این قسمت در تابع re موجود است.



برای قسمت دوم پروژه ابتدا به فرم چامسکی در آمده و سپس عضویت با توجه به الگوریتم cyk بررسی گردد.
نمونه کار الگوریتم cyk در تصویر زیر آمده :
کد مربوط به در تابع های cyk . fu1 . fu2 . fu3 و to_chomsky_form در کد قابل مشاهده است



بررسی عضویت رشته st در گرامر g

پی نوشت :

ما قصد داشتیم در ابتدای کار حذف قوانین لاندا ، واحد و بی فایده را هم اجرا کنیم تا برخی ایرادات احتمالی پیش نیاید ولی این کار کامل نشد ، لذا بر گرامر ورودی باید این مراحل طی شده باشد.

در ادامه کد حذف قوانین لاندا آمده است :

```python
def delete_landa(l:list):
    ad = []
    k = []
    for i in l:
        if i[1] == "LANDA":
            k.append(i)
            for j in l :

                if i[0] in j[1]:
                    ad.append( [j[0] , j[1][:j[1].find(i[0])]] )

                if i[0] == j[0] and j[1] != "LANDA":

                    for o in l:
                        if j[0] in o[1]:
                            ad.append([o[0], o[1][:o[1].find(i[0])] +j[1]])
    ll=[]
    # print("l",l)
    # print("k",k)
    # print("ad",ad)

    for i in ad:
        ll.append(i)

    for i in l:
        if not ( i in k):
            ll.append(i)

    return ll
```