باسمه تعالی

پروژه نهایی نظریه زبان ها و ماشین ها

عنوان پروژه: وجود رشته در NFA

اعضا گروه :نوید عباسی - سینا اخوی

مقدمه : این پروژه در دو قسمت فرانت (بخش UI برنامه) و بکند پیاده سازی شده

و نمونه مقدار ورودی در فایل txt. قرار داده شده در فایل Zip موجود هست

بنابراین به دو بخشی بودن برنامه توضیحات کد در دو قسمت صورت میگیره

بخش فرانت :

ابتدا تو پوشه view میرویم و فایل با پسوند HTML. رو باز میکنیم در صفحه باز شده با دو بخش مواجه میشویم :

۱- States ۲- Alphabet

برای مورد اول حالت های NFA رو تعریف میکنیم و برای بخش دوم الفبا مورد استفاده در زبان تعیین میشود

توجه: در بخش دوم نیازی به وارد کردن الفبا الپسیلون نیست چون در قسمت بکند کد در نظر گرفته شده است

پس از به درستی وارد کردن بخش بالا با دو قسمت جدید مواجه میشویم :

۱- Transition Function

۲-Start & Accept States

در بخش اول تابع انتقال رو تعریف میکنیم

در بخش دوم حالت شروع و حالت پایانی رو مشخص میکنیم

و در پایان generate NFA رو میزنیم

و شکل گرافیکی و پذیرفته شدن رشته نشون داده میشود

بخش بکند:

در بخش بکند به معرفی تابع های مورد استفاده میپردازیم (داخل پوشه public فایل با پسوند js.):

۱-function apply

۲-function clear

۳-class NFA state

٤- class NFA

٥- compute

Function apply:

در این تابع رشته های ورودی در قسمت UI رو مورد بررسی قرار میدهیم

و برای نمایش دادن از حالت "none" به حالت "block " میبریم که نمایش داده بشه

Function clear:

در صورت اشتباه وارد کردن فیلد های مربوط state , alphabet میتوان با زدن این بخش ورودی ها رو پاک کرد

و از نو نوشت

Class NFA State :

در این کلاس هر گره رو با توجه به ساختاری که داره ایجاد میکنیم

که شامل : Label , Transition , alphabet

و شامل متد createInitailTransion هست

کاربرد متد بالا: زمانی از این متد استفاده میشود که بخواهیم گره جدیدی تعریف کنیم

Class NFA:

در اینجا برای هر NFA ویژگی هایش مثل start state , final state , alphabet , transition

مشخص شده است

متد های این کلاس به صورته:

:findStateByName

در این تابع رشته میگیریم (اسم حالت رو به صورت ورودی گرفته) و حالت رو به صورت object برمیگردونیم

:Compute

تو این بخش رشته به عنوان ورودی میدهیم (مقادیر وارد شده توسط کاربر رو بررسی میکنیم) و در بدنه تابع بررسی میکنیم که آیا این رشته توسط NFA پذیرفته میشود یا خیر

به ازای خوانده شدن هر letter وارد قسمت newBranch میشویم

:CheckForAcceptOrReject

اگر یکی از اعضا پایانی در compute branches وجود داشته باشد Return ture میکند (رشته پذیرفته شده است) در غیر اینصورت Return false (رشته پذیرفته نمیشود)

```
compute(w) {
  let computeBranches = new Set(this.startState.valueOf());
  this.handleEpsiloneTransitions(computeBranches, this.startState.valueOf())

  console.log(computeBranches);
  for (const letter of w) {
    let futureBranches = new Set();
    computeBranches.forEach(branch => {
      const objectBranch = this.findStateByName(branch);


      const newBranches = new Set(objectBranch.transitions[letter]);

      newBranches.forEach(newBranch => {
        futureBranches.add(newBranch);
        this.handleEpsiloneTransitions(futureBranches, newBranch);
      });
    });

    computeBranches = futureBranches;
  }

  const acceptOrReject = this.checkForAcceptOrReject(computeBranches);
  return acceptOrReject;

}
```

:newBranch

بعد از خواندن هر ورودی وارد شاخه جدید محاسباتی میشویم

و رشته مرحله قبل رو در قسمت futureBranch ذخیره میکنیم

که یعنی محاسبات بعدی رو این شاخه نیاز است که پردازش شود

HandleEpsilonTransion: برای حالت اپلسیون که جز الفبا NFA هست پیاده سازی شده است.

اگر ما در new branch باشیم در واقع نیاز هست که هم شاخه جدید رو اضافه کنیم به future branchهم گره بعد رو

حال اگر همین گره بعدی با اپسیلون به گره بعدی بره نیاز هست که این گره هم به future branch افزوده بشه

حال در ادامه درصورتی که هیچ اپسیلونی تعریف نشده باشد return میکنیم(در واقع هیچی برنمیگرده)

و از طرفی اگر گره جدید باشه که اپسیلون داشته باشه نیازه که این تابع دوباره فراخوانی بشه که ما در اینجا تابع رو به صورت بازگشتی تعریف نموده ایم

نمونه خروجی برنامه:

نمونه ورودی اول:

```
a,b,0,1
a,c,0
b,d,0,1
d,b,1
d,a,0
c,d,$
```

نمونه خروجی اول:

States

Provide a comma separated list of state names.

a,b,c,d

Alphabet

Provide a comma separated list of strings to serve as the input alphabet.
Each symbol must be exactly one character.

0,1 Apply clear

Transitions Function

Enter a separate line for each transition in this format: origin_state, target, inputs

```
a,b,0,1
a,c,0
b,d,0,1
d,b,1
```

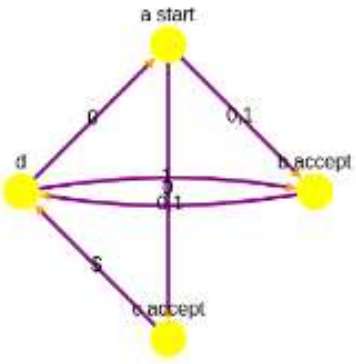
Start & Accept States

Start States: a b c d

Accep States: a b c d

generate NFA

Enter a word to compute: 1100 compute ACCEPTED



نمونه ورودی دوم:

```
b,d,$,1
c,d,$
d,b,1
d,a,0
```

نمونه خروجی دوم:

States

Provide a comma separated list of state names.

a,b,c,d

Alphabet

Provide a comma separated list of strings to serve as the input alphabet.
Each symbol must be exactly one character.

0,1 Apply clear

Transitions Function

Enter a separate line for each transition in this format: origin_state, target, inputs

```
b,d,$,1
c,d,$
d,b,1
d,a,0
```

Start & Accept States

Start States: a b c d

Accep States: a b c d

generate NFA

Enter a word to compute: 00 compute REJECTED