به نام خدا

درس نظریه زبان ها

تمرین اول

نیمسال 03_1402

استاد
نرگس السادات بطحائیان

دانشجو
دانیال کشاورز

مهلت ارسال
26 فروردین 1401

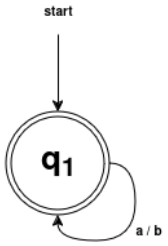
مقدمه و شرح تمرین

با توجه به صلاحدید استاد سوال 24 از فصل 2 در سکشن 1 کتاب An introduction to formal language and automata به عنوان تکلیف به بنده محول شد در این تمرین با فرض اینکه $L = \{awa : w \in \{a, b\}*\}$ باشد خواسته شده regular بود زبان $L*$ را به اثبات برسانیم

حل مسئله

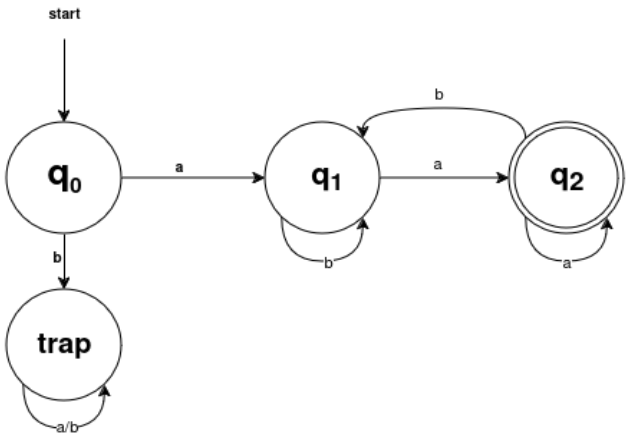
در گام نخست مجموعه $\{a, b\}*$ را تحلیل میکنیم واضح است که این مجموعه برابر می شود با

$\{a, b\}*$ = { λ , a , b, aa , ab , ba , bb , ...}



گام دوم به ساخت زبان L منتهی می شود با توجه به تعریف زبان صرفا با concat حرف a به اول و آخر تمام رشته های گام نخست میتوان مجموعه L را ساخت

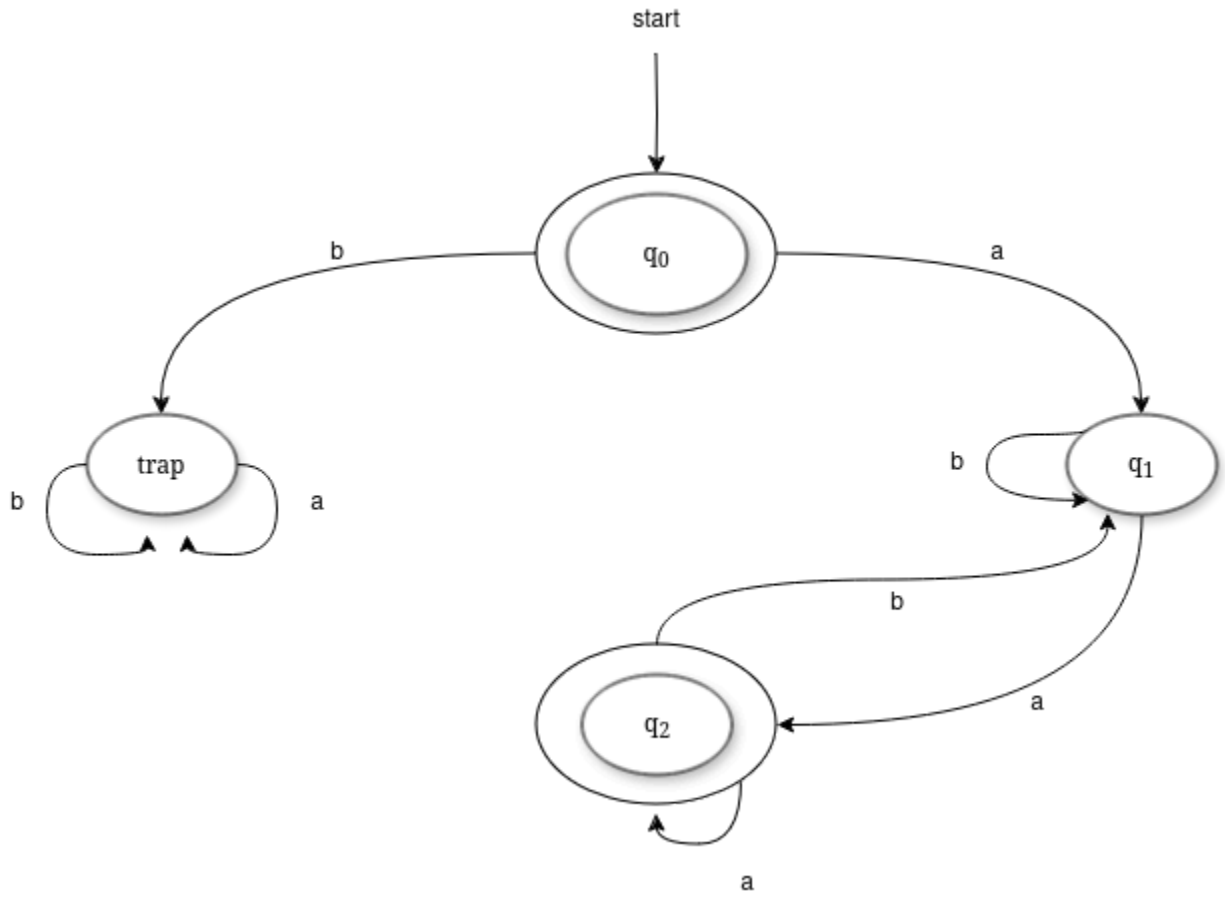
L = { aa , aaa , aba , $aaaa$, $aaba$, $abaa$, $abba$, ...}



در آخرین گام با توجه با $L*$ تمام رشته ها را می سازیم

$L* = \{\lambda\ ,\ aa\ , aaa\ , aaaba\ , aabaaaba\ ,\ ...\}$

حال برای اثبات regular بود این زبان کافیست DFA آن را رسم کنیم که به شرح زیر است



پذیرش و عدم پذیرش رشته ها

رشته abba

در ابتدا برنامه به حالت اول یعنی q میرود با ورودی a به حالت q1 میرود با b اول و دوم در همان حالت میماند د در نهایت به حالت q2 می رود حال با اتمام رشته ما در حالتی final هستیم پس رشته مورد پذیرش واقع میشود

رشته baabaa

در این رشته به روال همیشگی از حالت پایه یعنی q0 شروع میکنیم با اولین ورودی به حالت trap می رویم این حالت همانطور که از اسمش مشخص است که با هر کدام یک از اعضای الفبای DFA در همان حالت باقی خواهیم ماند در نهایت با پیان رشته داخل همین حالت گرفتار شدیم که final نیست و DFA آن را نمی پذیرد

رشته abab

ابتدا وارد حالت q0 شده سپس با a به q1 میرویم سپس با b در همین حالت می مانیم سپس با a به حالت q2 می رویم و با آخرین حرف رشته دوباره به حالت q1 باز میگردیم که جزو حالات final نیست و پذیرفته نمی شود

کد و پیاده سازی

برای پیاده سازی سعی شده از حالت معمول صرف نظر کرده و با روش دیگر DFA را پیاده سازی کرد

واضح است که DFA یک گراف است و ما با هر مقدار عضو الفبا تعریف شده در DFA می توانیم به یک راس جدید منتقل شویم پس کلاس DFA را به روشی شبیه پیاده سازی گراف وزن دار می توانیم پیاده کنیم و اینگونه کلاس ما می تواند هر DFA متفاوت را نشان دهد

```
class DFA {
    friend std::ostream& operator<< (std::ostream& out , const DFA& obj) ;

    public:
        DFA();
        DFA(std::string);

        void readFile   (std::string);
        void addVertex  (std::string);
        void addEdge    (std::string,std::string,std::string);
        bool checkWord  (std::string);
        void setFinals  (std::unordered_set<std::string>);
        void setAlphabet(std::unordered_set<std::string>);
        void setFinals  (std::vector<std::string>);
        void setAlphabet(std::vector<std::string>);
        void setInitialState(std::string);

    private:
        bool traceWord(std::string,int,std::string);
        std::vector<std::string> splitString(std::string,char spliter);

        std::unordered_map<std::string,std::unordered_map<std::string,std::string>> graph;
        std::unordered_set<std::string> finalsStates;
        std::unordered_set<std::string> alphabet;
        std::string initialState ;
```

برای ساخته شدن یک DFA تابع سازنده این کلاس فراخوانی میشود این تابع فایل txt را باز کرده و با توجه با اطلاعات موجود آن یک DFA را درست میکند به طور مثال در فایل زیر در خط اول تمام حالت آماده است سپس الفبای مقبول ماشین مهیا شده و در نهایت در خصوص بدی نشان داده شده اگر در یک حالت باشیم با هر کدام از اعضای الفبا به کدام حالت خواهیم رفت
در دو خط آخر حالت پایه و سپس حالات final به ترتیب آماده اند

ansewer2_1_24.txt

```
q0 q1 q2 trap
a b
q0 a q1
q0 b trap
q1 a q2
q1 b q1
q2 a q2
q2 b q1
trap a trap
trap b trap
q0
q0 q2
```

در نهایت می توانیم در main برنامه کلاس خود را چک کنیم

```
int main () {
    DFA answer("ansewer2_1_24.txt");

    cout << answer.checkWord("baababa") << "\n\n";
    cout << answer.checkWord("aababa" ) << "\n\n";
    cout << answer.checkWord("abbbbba") << "\n\n";

    return 0;
}
```

که خروجی کد نشان می دهد ماشین رشته اول را نمی پذیرد اما رشته دوم و سوم را می پذیرد که کاملا مطابق با هدف ماشین اصلی ماست این ماشین با پیچیدگی زمانی پذیرش یک رشته به طول n در هر ماشین فارغ از تعداد حالات و الفبای آن برابر است با O(n)
که کلاس بتواند بهینگی خود را ، حتی در ماشین ها با سایز بزرگتر حفظ کند
برای این مقصود در کلاس از ساختار های متکی به هش استفاده شده لینک گیت هاب این پروژه ضمیمه شده

https://github.com/DKeshavarz/formal-anguage-

بهار 1403