کامپایلر
پویشگر
از عبارت منظم تا پویشگر

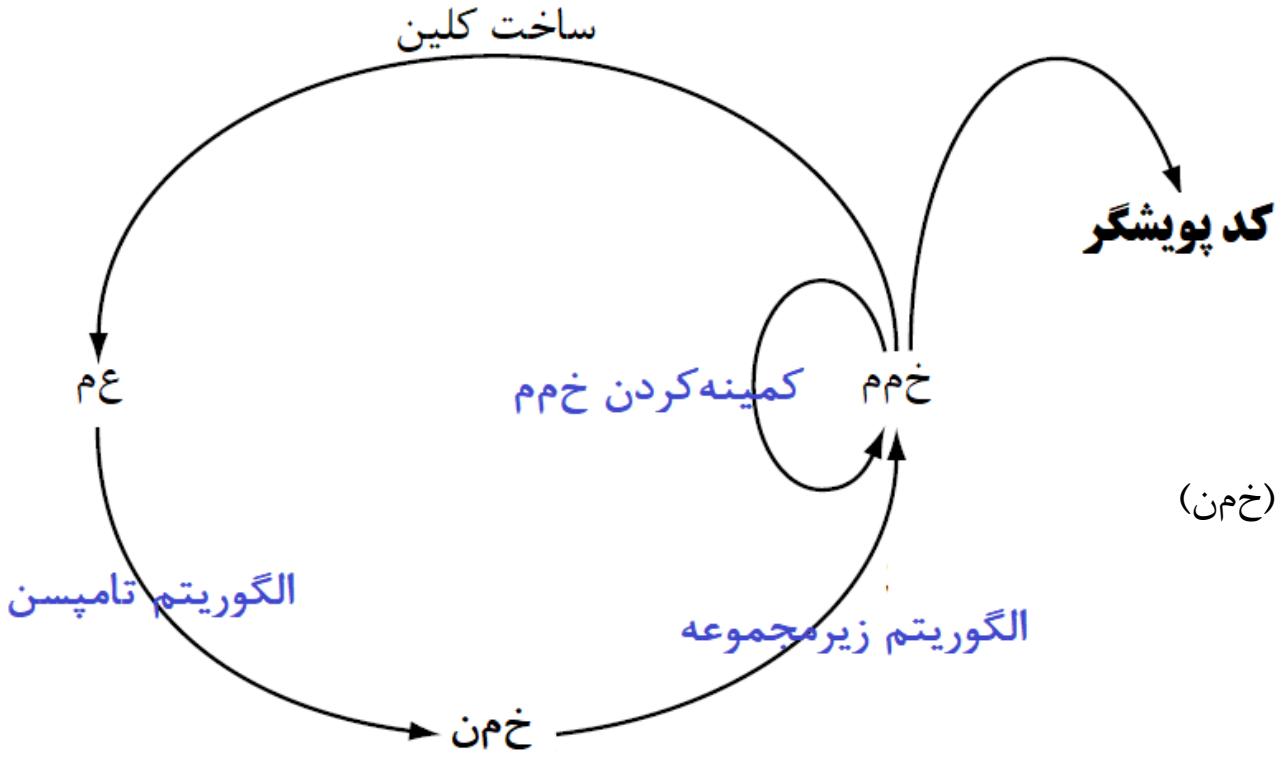
محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

عبارت منظم و پویشگر

دلیل استفاده از خم

- خودکارسازی تولید پویشگر اجراپذیر از عم
- پس ابتدا تعریف عم
- جهت دستیابی به مقصود نیاز به تعریف خودکاره متناهی نامعین (خمن)

عبارت منظم و پویشگر



دلیل استفاده از خ‌م

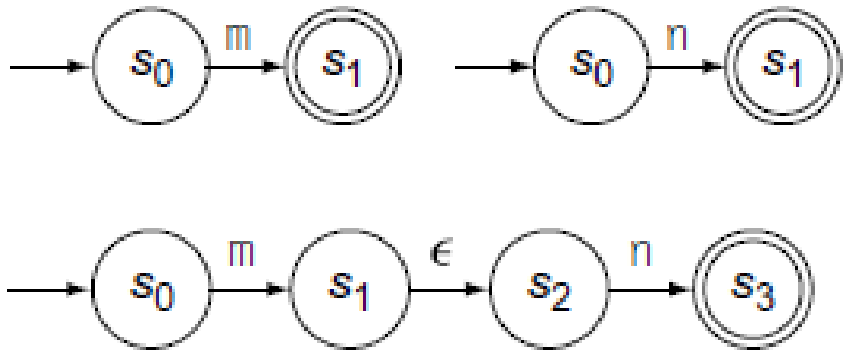
- خودکارسازی تولید پویشگر اجراپذیر از ع‌م
- پس ابتدا تعریف ع‌م
- جهت دستیابی به مقصود نیاز به تعریف خودکاره متناهی نامعین (خ‌م‌ن)

خودکاره متناهی نامعین (خ‌م‌ن)

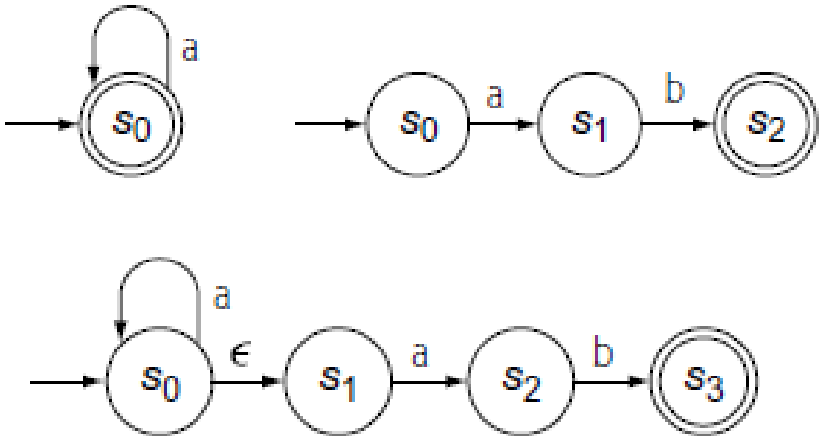
- پنج‌تائی (S,Σ,δ,s_0,F)
- S مجموعه متناهی از حالت‌های تشخیص‌گر (دارای حالت خطا s_e)
- Σ مجموعه متناهی از الفباء مورداستفاده تشخیص‌گر
- δ تابع انتقال خروجی تابع انتقال مجموعه توانی از حالات است.
  - در هر حالت، می‌توان با هیچ یا یک ورودی از مجموعه الفبا به حالت یا حالات دیگر رفت.
- s_0 حالت آغاز
- F مجموعه حالات پذیرش (نهائی) زیرمجموعه‌ای از S. اگر با خواندن تمامی رشته ورودی، خودکاره در یکی از حالات پذیرش باشد، رشته پذیرفته شده است.

پیکربندی

خودکاره متناهی نامعین - *ادامه*

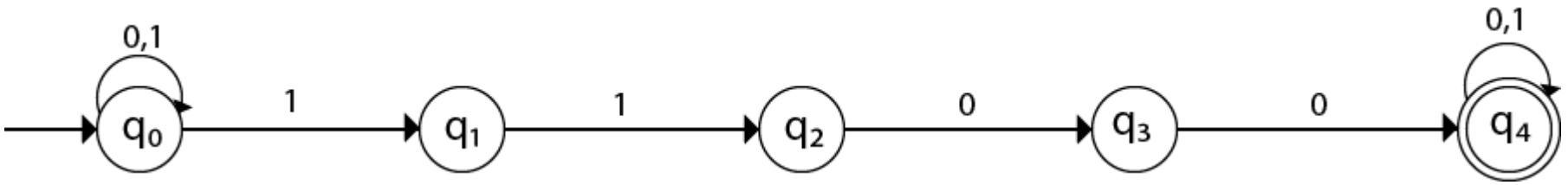


خودکاره متناهی نامعین - *ادامه*



خودکاره متناهی نامعین - *ادامه*

؟



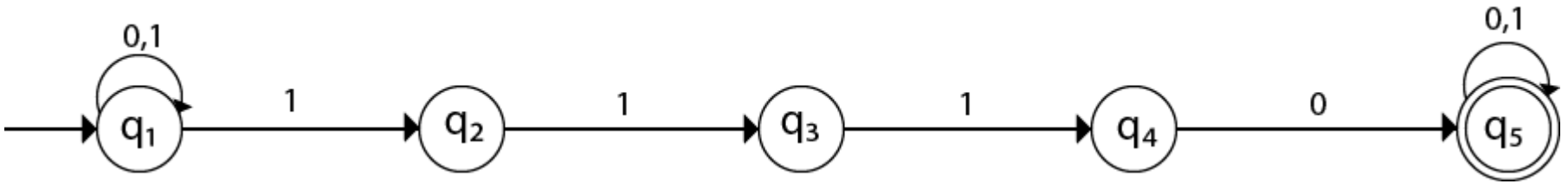
https://www.javatpoint.com/examples-of-non-deterministic-finite-automata

خودکاره متناهی نامعین – *ادامه*

خ‌من جهت تشخیص جملات با زیررشتهٔ 1110؟

خودکاره متناهی نامعین - *ادامه*

خ‌من جهت تشخیص جملات با زیررشتهٔ 1110؟



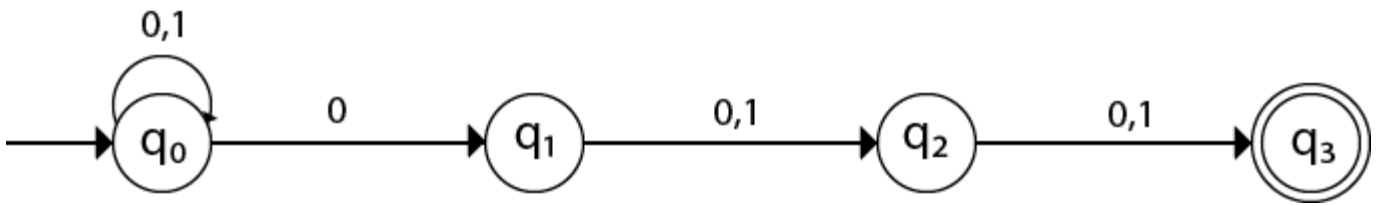
https://www.javatpoint.com/examples-of-non-deterministic-finite-automata

خودکاره متناهی نامعین - *ادامه*

خ‌من جهت تشخیص جملاتی که سومین عضو از سمت راست همیشه برابر 0؟

خودکاره متناهی نامعین - *ادامه*

خ‌من جهت تشخیص جملاتی که سومین عضو از سمت راست همیشه برابر 0؟



https://www.javatpoint.com/examples-of-non-deterministic-finite-automata

تناظر خودکاره معین و نامعین

خم و خمن دارای قدرت بیان یکسان

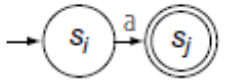
هر خم حالت خاصی از خمن

هر خمن را می‌توان با خم نشان داد

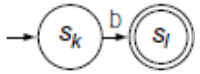
هر خمن با n حالت دارای حداکثر $|\sum|^n$ حالت است

تبدیل عبارت منظم به خودکاره نامعین (تامپسن)

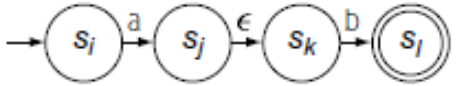
خمن برای a



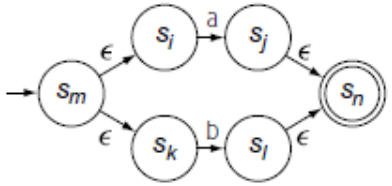
خمن برای b



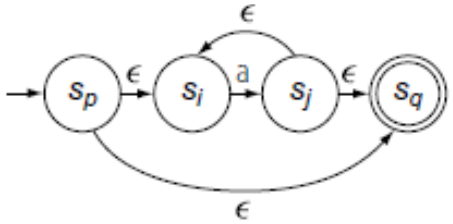
خمن برای ab



خمن برای a|b

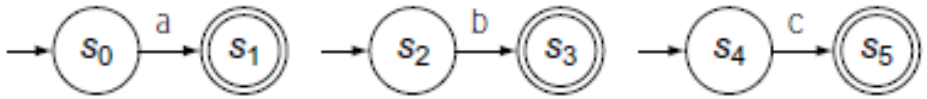


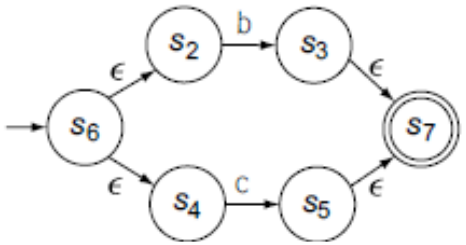
خمن برای a^*



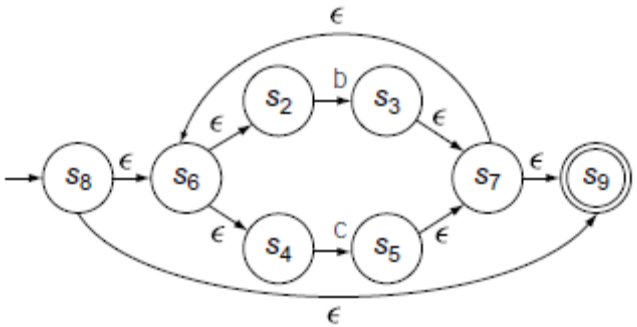
روش تامپسن- مثال

a(b|c)*

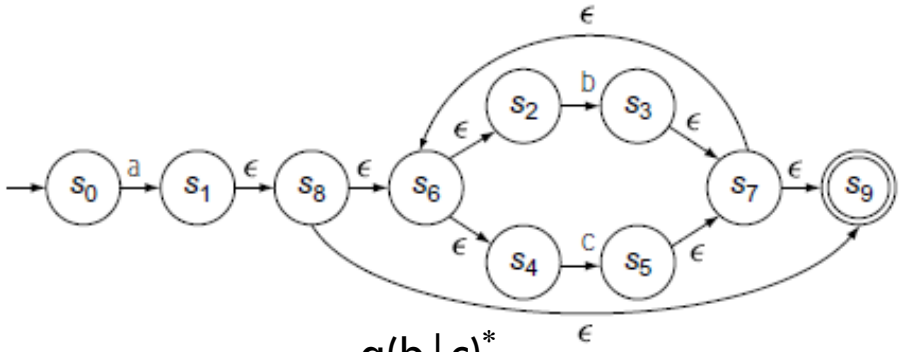




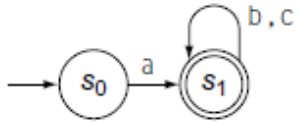
b | c



(b | c)*

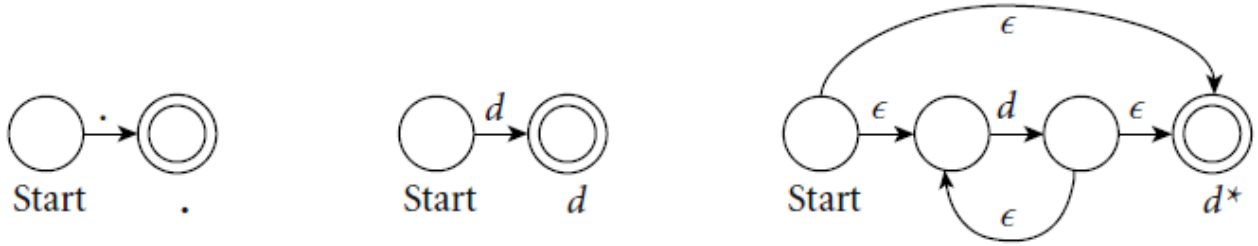


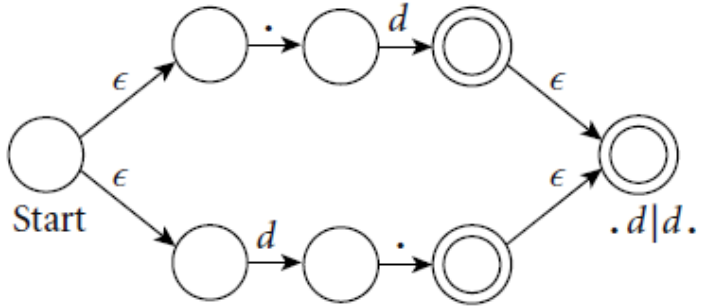
a(b | c)*

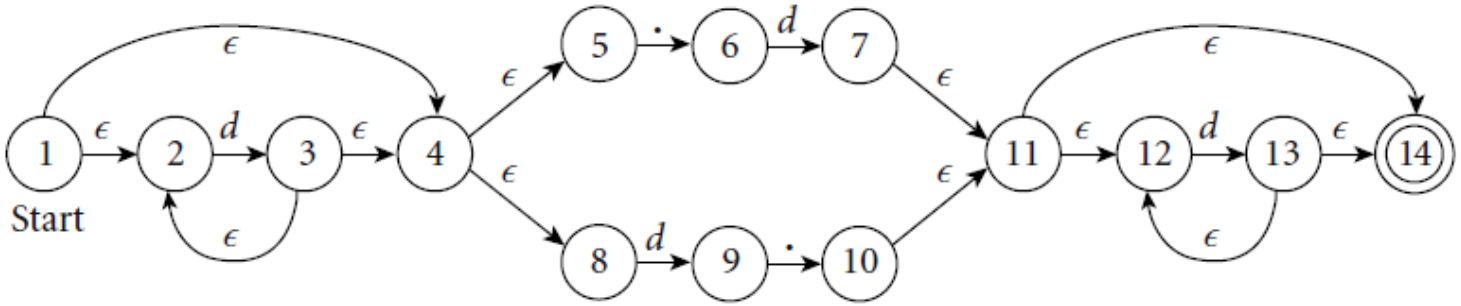


مثال ۲ –

$d\,*(\,.d \mid d.\,)\,d\,*$



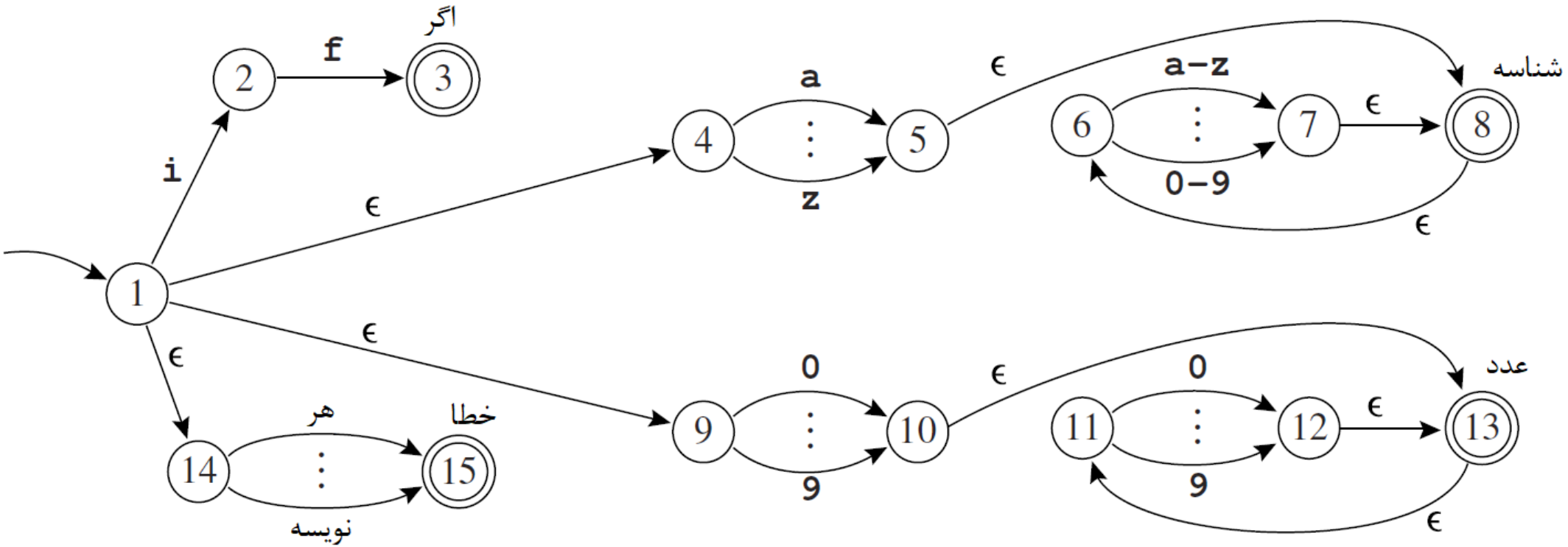




روش تامپسن – مثال۳

```
if                                              {return IF;}
[a-z][a-z0-9]*                                  {return ID;}
[0-9]+                                          {return NUM;}
([0-9]+"."[0-9]*)|([0-9]*"."[0-9]+)             {return REAL;}
("--"[a-z]*"\n")|(" "|"\n"|"\t")+               {  /* do nothing */ }
.                                               {error();}
```

روش تامپسن – مثال ۳ - *ادامه*



ویژگی‌های روش تامپسن

هر خ‌من دارای یک حالت آغاز و یک حالت پذیرش

حالت آغاز دارای هیچ انتقالی به جز انتقال آغازین

حالت پذیرش بدون هیچ حالت خروجی به حالت دیگر

انتقال-اپسیلون دو حالت را متصل می‌کند

- حالت آغاز و حالت پذیرش دو خ‌من قبلی

هر حالت حداکثر دارای دو ورودی انتقال-اپسیلون و دو خروجی حالت اپسیلون

تمامی موجب ساده‌کرد نمایش و کار با خ‌من

تبدیل خودکاره نامعین به خودکاره معین

نیاز به جستجوی تمام پیکربندی‌های ممکن

- موازی
- یا پس‌رو

پیچیده و زمان‌گیر

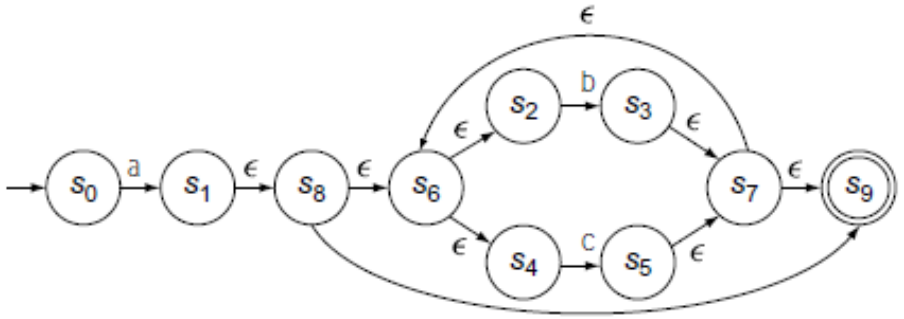
ساخت زیرمجموعه

بستار-ϵ حالت آغاز(q_0).

مرحله اول $\epsilon(\delta(q_0,a_i))$ و $a_i \in \Sigma$

در مراحل بعد حالات شکل گرفته جدید را پی می‌گیرد

تبدیل خمن به خمم



a(b | c)*

نام	معادل	a	b	c
q0	s0	{s1,s8,s6, 2,s4,s9} =q1	ϕ	ϕ
q1	{s1,s8,s6, 2,s4,s9}	ϕ	{s3,s7,s9, 6,s2,s4} =q2	{s5,s7,s9, 6,s2,s4} =q3
q2	{s3,s7,s9,	ϕ	q2	q3
q3	{s5,s7,s9,			

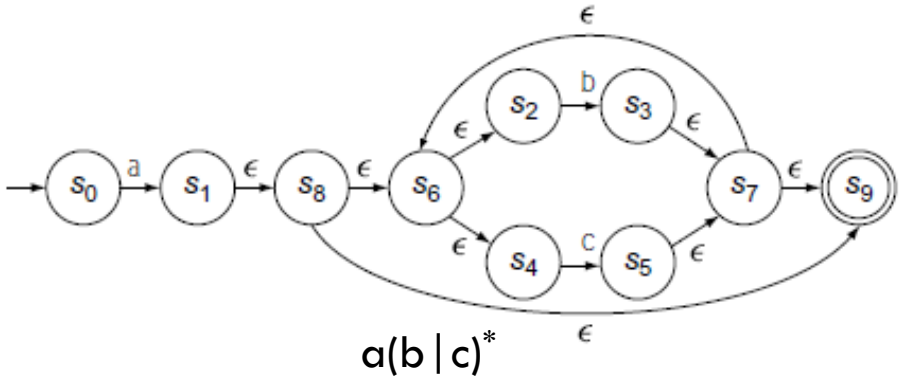
- $\epsilon(s0) = \{s0\} = q0$

- $\epsilon(\delta(q0, a)) = \epsilon(s1) = \{s1, s8, s6, s2, s4, s9\} = q1$
- $\epsilon\big(\delta(q0, b)\big) = \epsilon(\phi) = \phi$
- $\epsilon(\delta(q0, c)) = \epsilon(\phi) = \phi$

- $\epsilon(\delta(q1, a)) = \epsilon(\delta(\{s1, s8, s6, s2, s4, s9\}, a)) = \epsilon(\phi) = \phi$
- $\epsilon(\delta(q1, b)) = \epsilon(\delta(\{s1, s8, s6, s2, s4, s9\}, b)) = \epsilon(s3) = \{s3, s7, s9, s6, s2, s4\} = q2$
- $\epsilon(\delta(q1, c)) = \epsilon(\delta(\{s1, s8, s6, s2, s4, s9\}, c)) = \epsilon(s5) = \{s5, s7, s9, s6, s2, s4\} = q3$

- $\epsilon(\delta(q2, a)) = \epsilon(\delta(\{s3, s7, s9, s6, s2, s4\}, a)) = \epsilon(\phi) = \phi$
- $\epsilon(\delta(q2, b)) = \epsilon(\delta(\{s3, s7, s9, s6, s2, s4\}, b)) = \epsilon(s3) = \{s3, s7, s9, s6, s2, s4\} = q2$
- $\epsilon(\delta(q2, c)) = \epsilon(\delta(\{s3, s7, s9, s6, s2, s4\}, c)) = \epsilon(s5) = \{s5, s7, s9, s6, s2, s4\} = q3$

تبدیل خمن به خمم

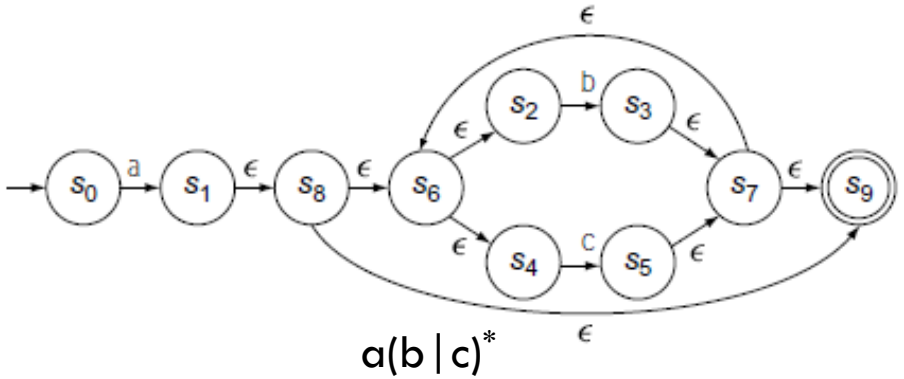


$a(b \mid c)^*$

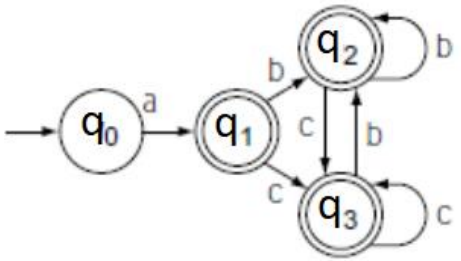
- $\epsilon(\delta(q3,a)) = \epsilon(\delta(\{s3,s7,s9,s6,s2,s4\},a)) = \epsilon(\phi) = \phi$
- $\epsilon(\delta(q3,b)) = \epsilon(\delta(\{s3,s7,s9,s6,s2,s4\},b)) = \epsilon(s3) = \{s3,s7,s9,s6,s2,s4\} = q2$
- $\epsilon(\delta(q3,c)) = \epsilon(\delta(\{s3,s7,s9,s6,s2,s4\},c)) = \epsilon(s5) = \{s5,s7,s9,s6,s2,s4\} = q3$

نام	معادل	a	b	c
q0	s0	{s1,s8,s6,	ϕ	ϕ
q1	{s1,s8,s6, 2,s4,s9}	ϕ	{s3,s7,s9, 6,s2,s4} =q2	{s5,s7,s9, 6,s2,s4} =q3
q2	{s3,s7,s9,	ϕ	q2	q3
q3	{s5,s7,s9,	ϕ	q2	q3

تبدیل خمن به خمم



$a(b \mid c)^*$



نام	معادل	a	b	c
q0	s0	{s1,s8,s6,	ϕ	ϕ
q1	{s1,s8,s6, 2,s4,s9}	ϕ	{s3,s7,s9, 6,s2,s4} =q2	{s5,s7,s9, 6,s2,s4} =q3
q2	{s3,s7,s9,	ϕ	q2	q3
q3	{s5,s7,s9,	ϕ	q2	q3

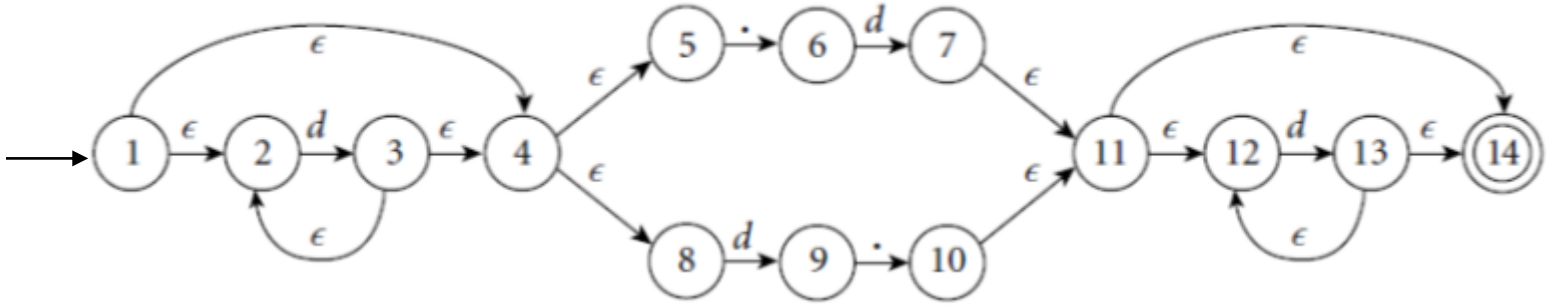
تبدیل خودکاره نامعین به خودکاره معین

ساخت زیرمجموعه

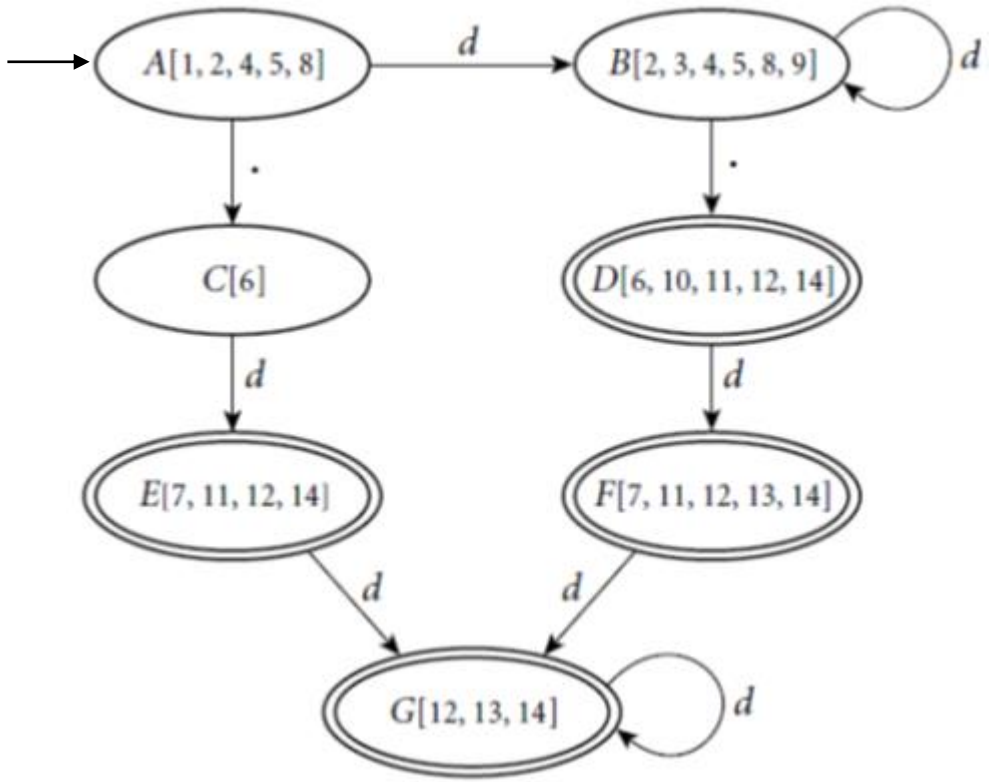
```
q0 ← ε-closure({n0});
Q ← q0;
WorkList ← {q0};

while (WorkList ≠ ∅ ) do
   remove q from WorkList;
   for each character c ∈ Σ do
      t ← ε-closure(Delta(q,c));
      T[q,c] ← t;
      if t ∉ Q then
         add t to Q and to WorkList;
   end;
end;
```

مثال


1
2
3
4
5
6
7
8
9
10
11
12
13
14
ϵ
d
.

$d\ *(\ .d\ |\ d.\ )\ d\ *$


A[1, 2, 4, 5, 8]
B[2, 3, 4, 5, 8, 9]
C[6]
D[6, 10, 11, 12, 14]
E[7, 11, 12, 14]
F[7, 11, 12, 13, 14]
G[12, 13, 14]
d
.

تبدیل خ‌م‌م به خ‌م‌م کمینه (هاپکرافت)

تولید خ‌م‌م با تعداد زیادی حالت

بی‌تاثیر بر زمان اجرا و خواندن رشته

اشغال حافظه زیاد

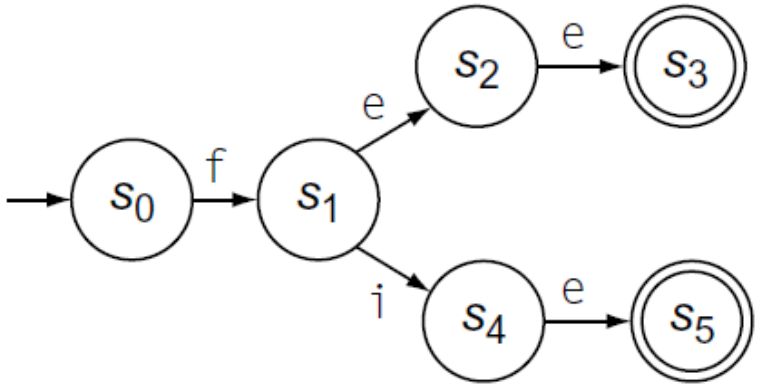
نیاز به روشی جهت تشخیص دو حالت همسان

- رفتار یکسان به ازای هر ورودی

تبدیل خ‌م‌م به خ‌م‌م کمینه (هاپکرافت)

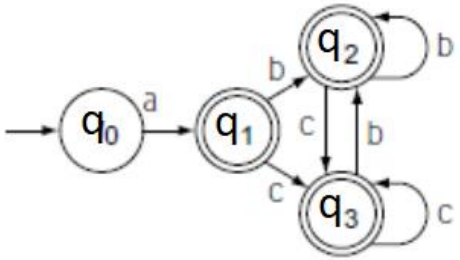
```
T ← {D_A,  {D − D_A}};
P ← ∅
while (P ≠ T) do
    P ← T;
    T ← ∅;
    for each set p ∈ P do
        T ← T ∪ Split(p);
    end;
end;
```

```
Split(S) {
    for each c ∈ Σ do
        if c splits S into s_1 and s_2
            then return {s_1, s_2};
    end;
    return S;
}
```

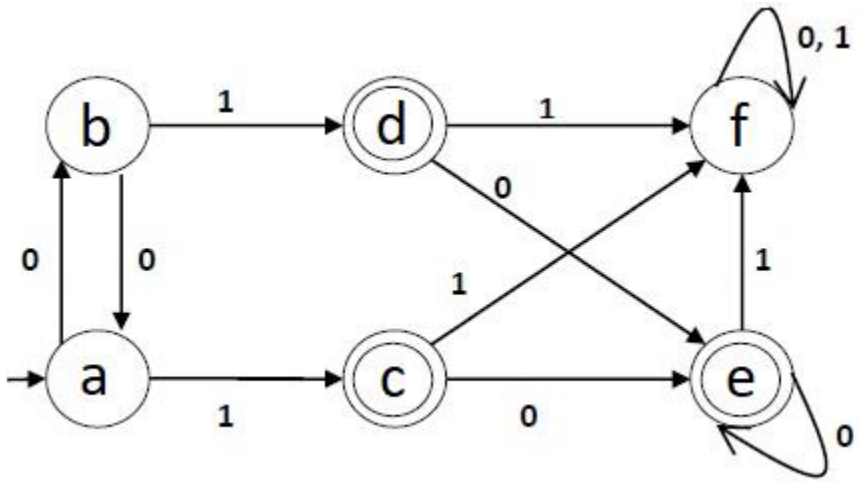


قدم	تقسیم‌بندی فعلی			
		مجموعه	نویسه	عمل
۰	{{s3,s5},{s0,s1,s2,s4}}	-	-	-
۱	{{s3,s5},{s0,s1,s2,s4}}	{s3,s5}	همه	هیچ
۲	{{s3,s5},{s0,s1,s2,s4}}	{s0,s1,s2,s4}	e	جداسازی {s2,s4}
۳	{{s3,s5},{s0,s1},{s2,s4}}	{s0,s1}	f	جداسازی {s1}
۴	{{s3,s5},{s0},{s1},{s2,s4}}	همه	همه	هیچ

مثال

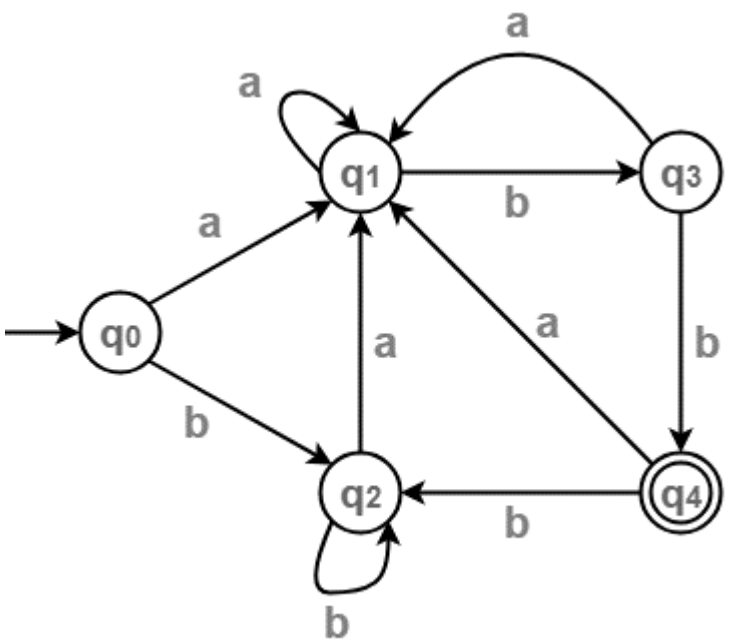


مثال



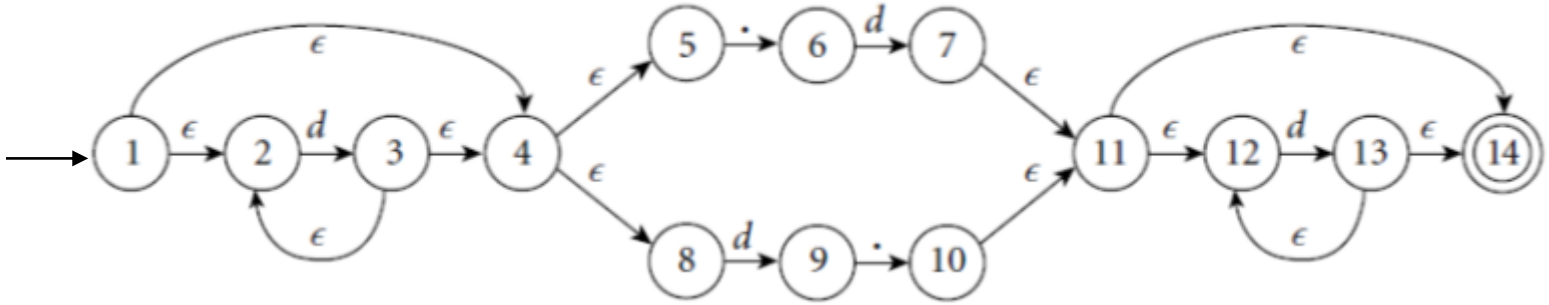
https://www.tutorialspoint.com/automata_theory/dfa_minimization.htm

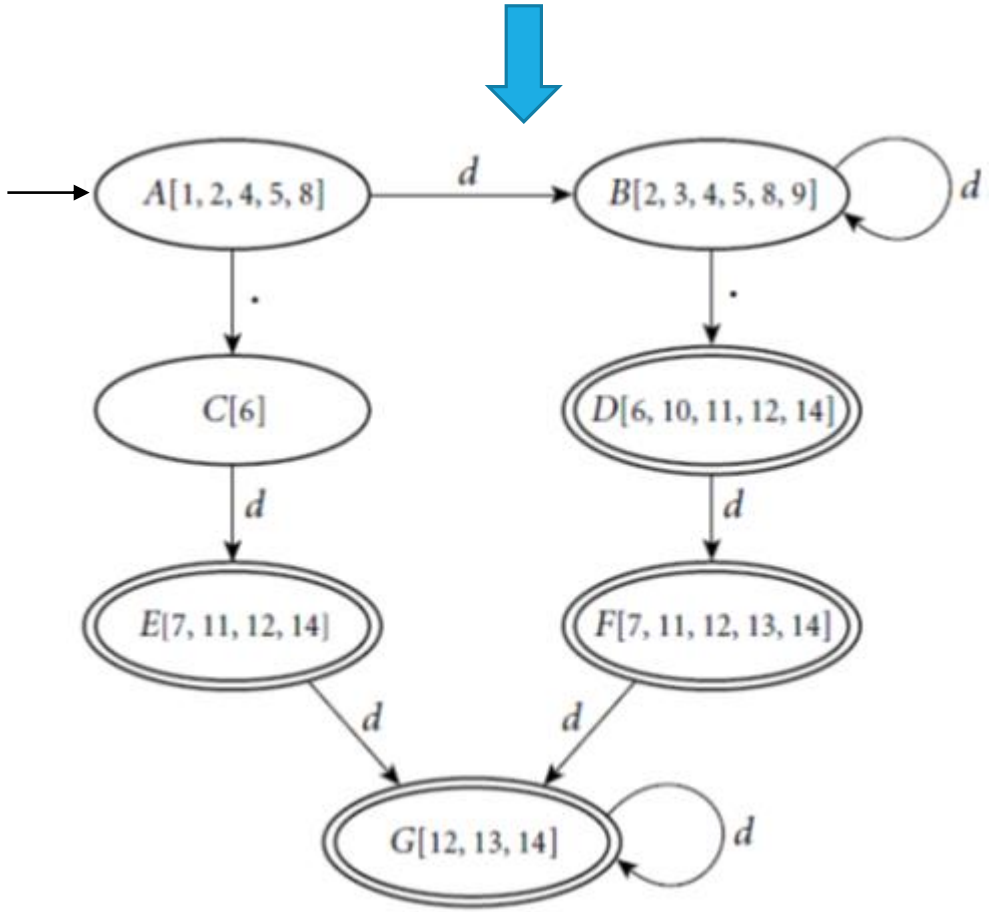
مثال

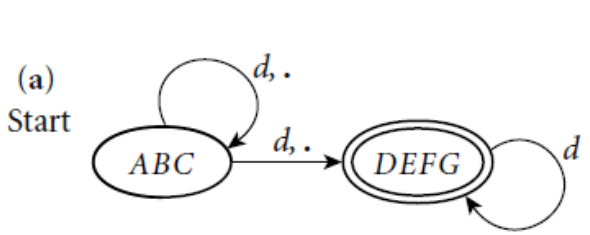


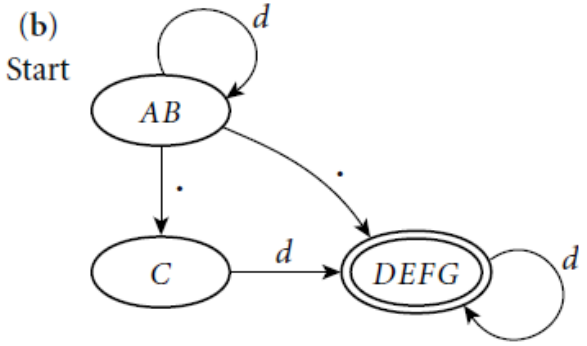
مثال

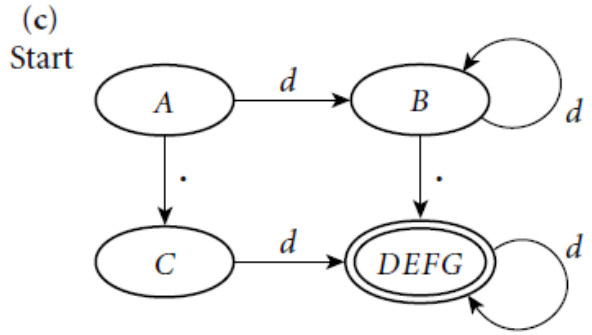
$d\,*(\,.d \mid d.\,)\,d\,*$











خودکاره معین به مثابه تشخیص‌گر

تاکنون توجه به یک عبارت منظم و ساخت خم‌م از آن

نیاز به تشخیص تمامی رده‌های نحوی

نیاز به تشخیص‌گری که تمامی عبارت‌های منظم ریزنحو زبان را مدیریت کند.

فرض که تمامی عبارت‌ها شامل

- $r1, r2, r3, \ldots, rk$
- آن‌گاه عبارت منظمی از تمامی آن‌ها
- $(r1 \mid r2 \mid r3 \mid \ldots \mid rk)$

خودکاره معین به مثابه تشخیص‌گر

ایجاد خ‌م‌ن از عبارت واحد

تبدیل به خ‌م‌م و کمینه‌کردن آن

تبدیل به کد اجرائی

مشخص کردن هر رشته و برگرداندن رده نحوی آن

قانون یافتن طولانی‌ترین کلمه

- منطبق بر عبارات منظم
- ادامه تا رسیدن به جایی که انتقال به حالت دیگری ممکن نباشد.

خودکاره معین به مثابه تشخیص‌گر

قانون یافتن طولانی‌ترین کلمه

- منطبق بر عبارات منظم
- ادامه تا رسیدن به جایی که انتقال به حالت دیگری ممکن نباشد.

سه وضعیت

- یا حالت فعلی حالت پذیرش
- حالت فعلی حالت پذیرش نیست
  - گذر خ‌م از یک یا بیشتر از چند از حالت پذیرش
  - طولانی‌ترین حالت پذیرفته می‌شود
  - خ‌م از هیچ حالت پذیرشی نگذشته است.
  - پیغام خطا

اولویت

- Lex: بر اساس ترتیب

منابع

[ببر سبز]

[اژدرها]

[کوپر]

[فیشر]

[نیسان]