اصول طراحی کامپایلر

حسین کارشناس

دانشکده مهندسی کامپیوتر

ترم اول ۹۸ – ۹۷

تحلیل واژه‌ای (پویش)

رئوس مطالب

- مثال‌های اولیه
  - آشنایی با مفاهیم مرتبط با تحلیل واژه‌ای
- توصیف واژه‌ها
  - عبارت‌های منظم
- شناسایی واژه‌ها
  - ماشین‌های متناهی

تحلیل واژه‌ای

- اولین مرحله از فرآیند کامپایل
- هدف: دریافت برنامه ورودی و تفکیک به اجزای شناخته شده
- مثال: برنامه ساده به زبان C

```
if (i == j)
    Z = 0;
else
    Z = 1;
```

- ورودی به شکل یک رشته پیوسته از نویسه‌ها (characters) دریافت می‌شود

```
\tif (i == j)\n\t\tz = 0;\n\telse\n\t\tz = 1;
```

تحلیل واژه‌ای

- وظایف اصلی

خواندن دنباله نویسه‌های برنامه ورودی

گروه‌بندی نویسه‌ها به شکل واژه‌ها

- استفاده از الگوی واژه‌های هر دسته نماد

تولید دنباله نمادهای شناسایی شده

- حذف توضیحات (comments) و فاصله‌های خالی (whitespaces)
- شناسایی خطاهای واژه‌ای

تحلیل واژه‌ای

- مفاهیم مورد استفاده
  - واژه (lexeme)
    - یک دنباله از نویسه‌های منطبق با یک الگوی مشخص (مصداق آن الگو)
  - الگو (pattern)
    - توصیف اشکال ممکن برای واژه‌های یک دسته نماد
  - نماد (token)
    - نمایانگر یک دسته از واژه‌های به هم مرتبط
    - نمایش به صورت یک چندتایی (tuple) شامل یک نام (شناسه) و تعدادی ویژگی
      - نام تعیین کننده دسته (نوع) نماد
      - ویژگی‌ها حاوی اطلاعات اضافی در مورد نماد

تحلیل واژه‌ای

- هر دسته نماد شامل مجموعه‌ای از واژه‌ها (رشته‌ها)
- برخی از انواع دسته نمادهای قابل شناسایی در برنامه‌ها
  - شناسه‌ها (identifiers)
  - کلیدواژه‌ها (keywords)
  - عملگرها (operators)
  - اعداد (numerical literals)
  - رشته‌ها (string literals)
  - نشانه‌های ویژه (مثلاً ;)
- مثال: دسته نماد اعداد صحیح در زبان‌های متداول برنامه‌نویسی
  - 0، 123، 4-، 07، ...

تحلیل واژه‌ای

- مثال: شناسایی دسته‌های نماد برای واژه‌های برنامه‌ها

```
\tif (i == j)\n\t\tz = 0;\n\telse\n\t\tz = 1;
```

```
<W> <K> <W> <(> <I> <W> <O> <W> <I> <)> <W>
<I> <W> <=> <W> <N> <;> <W> <K> <W> <I> <W>
<=> <W> <N> <;>
```

W: Whitespace
K: Keyword
I: Identifier
O: Operator
N: Number

(
)
=
;

تحلیل واژه‌ای

- سوال: از هر دسته نماد چه تعداد واژه در برنامه زیر وجود دارد؟

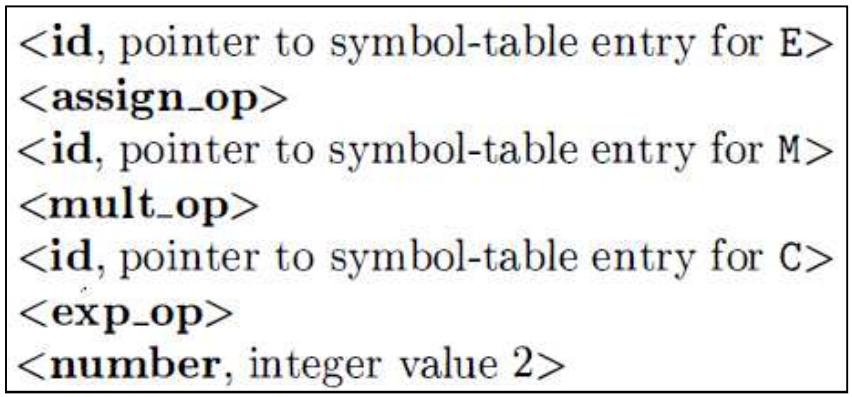
```
x = 0;\n\twhile (x < 10) {\n\tx++;\n}
```

- W = 9; K = 1; I = 3; N = 2; O = 9
- W = 11; K = 4; I = 0; N = 2; O = 9
- W = 9; K = 4; I = 0; N = 3; O = 9
- W = 11; K = 1; I = 3; N = 3; O = 9

W: Whitespace
K: Keyword
I: Identifier
N: Number
O: Other Tokens:
{ } ( ) < ++ ; =

تحلیل واژه‌ای

- ویژگی‌های یک نماد: اطلاعات خاص هر واژه
  - مثال: در زبان Fortran



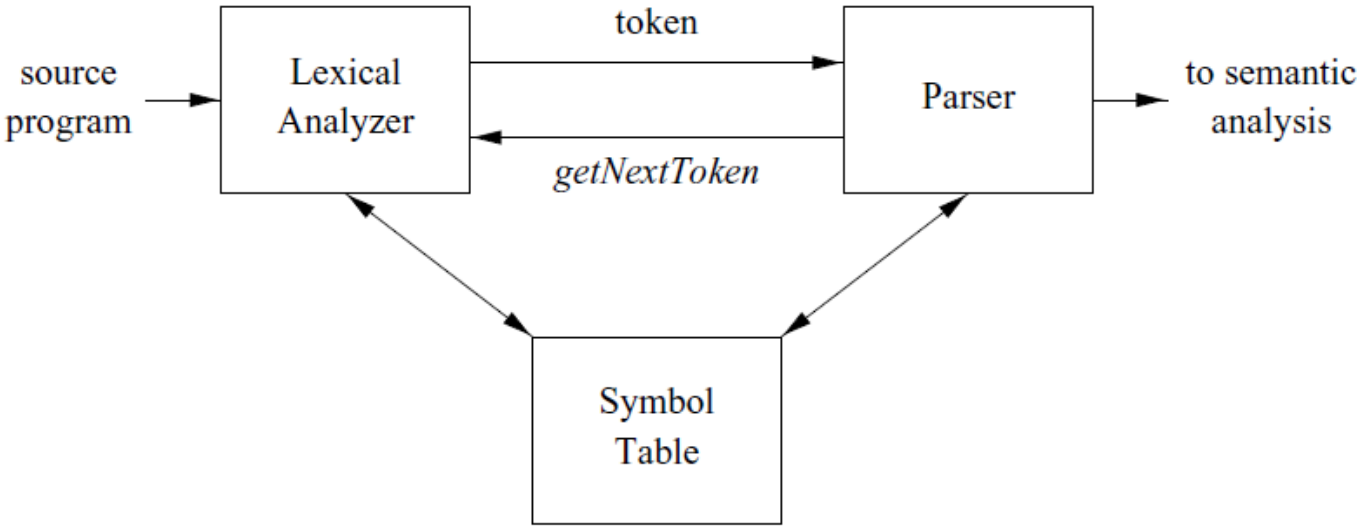

```
E = M * C ** 2
```

- پر کردن جدول نشانه‌ها (symbol table)
  - افزودن شناسه‌های برنامه ورودی به جدول
    - جستجوی جدول برای شناسه دیده شده در ورودی
    - اگر شناسه قبلاً در جدول نباشد به آن اضافه می‌شود

1	position	···
2	initial	···
3	rate	···

تحلیل واژه‌ای

- ارتباط با تجزیه‌گر (parser)



- معمولاً تحلیلگر واژه‌ای توسط تجزیه‌گر فراخوانی می‌شود
  - مثلاً با رویه getNextToken()

تحلیل واژه‌ای

- بکارگیری حافظه میانگیر (buffer) برای ورودی
- نیاز به خواندن ورودی‌های جلوتر (lookahead) برای شناسایی واژه‌ها
  - مثال: شناسایی پایان نام یک شناسه
  - مثال: تفکیک عملگرهای **-**، **=** و **>** از عملگرهای **->**، **==** و **>=**
  - مثال: امکان استفاده از فاصله‌های خالی در بین حروف شناسه‌ها در Fortran

```
DO 5 I = 1,25          DO 5 I = 1.25
```

  - مثال: عدم وجود کلیدواژه‌های از قبل رزرو شده در PL/1

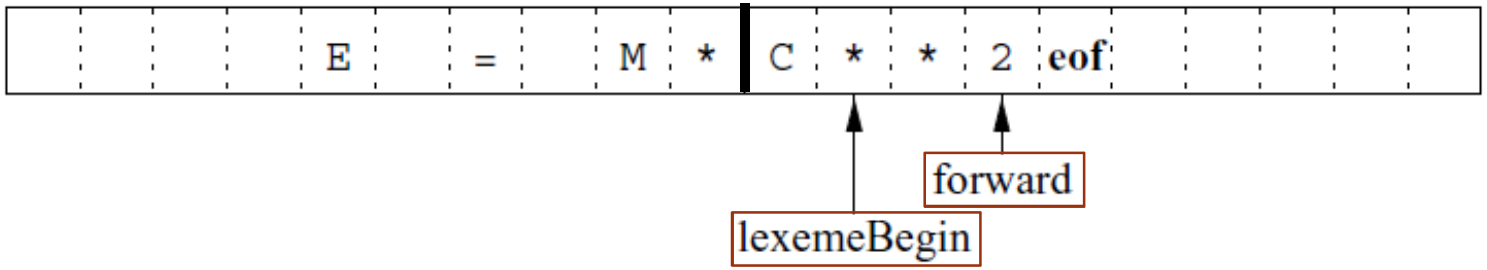
```
DECLARE (ARG1,. . ., ARGN)
```

  - مثال: شباهت نوع‌های الگویی (template) تودرتو و عملگرهای جریانی در C++

```
Add<Bar<Car>>
```

تحلیل واژه‌ای

- بکارگیری حافظه میانگیر (ادامه)
  - افزایش سرعت پردازش برنامه ورودی
    - نیاز به پردازش حجم انبوهی از نویسه‌ها در برنامه‌های بزرگ
  - راهکار مؤثر: استفاده از بافرینگ مضاعف (double buffering)
    - بکارگیری دو بافر بصورت همزمان
    - در هر بار خواندن، به اندازه ظرفیت بافر (یک بلاک) داده‌ها از دیسک منتقل می‌شوند
    - با پردازش محتویات هر بافر، بخش دیگری از ورودی در آن بارگذاری می‌شود
    - استفاده از دو نشانه‌رو (pointer) برای تعیین محدوده مورد بررسی در ورودی



رئوس مطالب

- مثال‌های اولیه
  - آشنایی با مفاهیم مرتبط با تحلیل واژه‌ای
- توصیف واژه‌ها
  - عبارت‌های منظم
- شناسایی واژه‌ها
  - ماشین‌های متناهی

توصیف واژه‌ها

عبارت‌های منظم

- نیاز به یک روش رسمی (formal) برای توصیف الگوی واژه‌ها
  - راهکار اصلی: استفاده از عبارت‌های منظم (regular expressions)
    - توان بیان کافی برای توصیف الگوی واژه‌ها را دارند

- مفاهیم مورد استفاده
  - الفبا: مجموعه‌ای از نشانه‌ها (symbols)
    - مثال: {0, 1}، ASCII، Unicode
  - رشته: یک دنباله محدود از نشانه‌های یک الفبا
    - مثال: aab، 2ad#$%، banana
    - طول رشته (|s|): تعداد نشانه‌های موجود در رشته
    - رشته تهی: ϵ

عبارت‌های منظم

- مفاهیم مورد استفاده (ادامه)
  - زبان: مجموعه‌ای قابل شمارش از رشته‌های یک الفبا
    - مثال: مجموعه اعداد صحیح، $\{\epsilon\}$، $\emptyset$
  - زیررشته (پیشوندی، پسوندی)، زیردنباله
  - عملیات‌های الحاق (concatenation) و توان روی رشته‌ها
    - برای مثال اگر s و t دو رشته باشند: st، ts، s^3، t^0
  - عملیات اجتماع، الحاق و بستار (closure) روی زبان‌ها
    - برای مثال اگر L و D دو زبان باشند:
      - $L \cup D$، LD، L^*
      - عملیات بستار مثبت (L^+)

عبارت‌های منظم

- تعریف استقرایی عبارت منظم روی یک الفبا
  - **ϵ** یک عبارت منظم است
  - هر نشانه از الفبا یک عبارت منظم است
  - اگر **r** یک عبارت منظم باشد آنگاه **(r)** نیز یک عبارت منظم است
  - اگر **r** و **q** عبارت‌های منظم باشند آنگاه **r | q** نیز یک عبارت منظم است
  - اگر **r** و **q** عبارت‌های منظم باشند آنگاه **rq** نیز یک عبارت منظم است
  - اگر **r** عبارت منظم باشد آنگاه $\mathbf{r}^*$ نیز یک عبارت منظم است

- زبان منظم: قابل تعریف توسط عبارت‌های منظم
  - دو عبارت منظمی که زبان یکسانی را تعریف کنند معادل (equivalent) هستند

عبارت‌های منظم

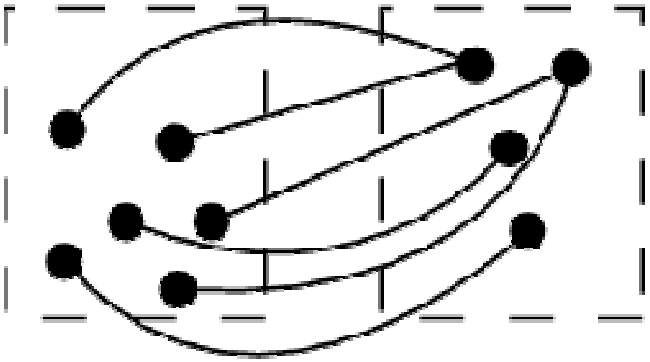
- تابع معنا (meaning function)
  - تابعی که ساختار (syntax) را به معنا (semantics) می‌نگارد:

$$L(r) = M$$

  - مثال: زبان منظمی که مجموعه رشته‌های منطبق با یک عبارت منظم را تعیین می‌کند

$$\text{Reg. Lan.} : \text{Reg. Expr.} \rightarrow \{\text{strings}\}$$

- امکان بررسی مجزای ساختار و معنا
  - مثلاً تغییر ساختار نشان دهنده یک معنا
- ساختار و معنا دارای رابطه یگانی نیستند
  - وجود ساختارهای متفاوت برای یک معنا
  - فلسفه بهینه‌سازی در کامپایلرها



عبارت‌های منظم

- برابری‌های جبری عبارت‌های منظم (*r*, *s*, and *t* are any regular expressions)

1.	$r \mid s = s \mid r$	(commutativity for alternation)
2.	$r \mid (s \mid t) = (r \mid s) \mid t$	(associativity for alternation)
3.	$r \mid r = r$	(absorption for alternation)
5.	$r(st) = (rs)t$	(associativity for concatenation)
6.	$r(s \mid t) = rs \mid rt$	(left distributivity)
7.	$(s \mid t)r = sr \mid tr$	(right distributivity)
8.	$r\epsilon = \epsilon r = r$	(identity for concatenation)
9.	$r^*r^* = r^*$	(closure absorption)
10.	$r^* = \epsilon \mid r \mid rr \mid \ldots$	(Kleene closure)
11.	$(r^*)^* = r^*$	
12.	$rr^* = r^*r$	
13.	$(r^* \mid s^*)^* = (r^*s^*)^*$	
14.	$(r^*s^*)^* = (r \mid s)^*$	
15.	$(rs)^*r = r(sr)^*$	
16.	$(r \mid s)^* = (r^*s)^*r^*$	

عبارت‌های منظم

- سوال
  - کدام یک از عبارت‌های منظم زیر معادل عبارت منظم $(0 \mid 1)^*1(0 \mid 1)^*$ است؟

$\Sigma = \{ 0, 1 \}$

- ☐ $(01 \mid 11)^*(0 \mid 1)^*$
- ☐ $(0 \mid 1)^*(10 \mid 11 \mid 1)(0 \mid 1)^*$
- ☐ $(1 \mid 0)^*1(1 \mid 0)^*$
- ☐ $(0 \mid 1)^*(0 \mid 1)(0 \mid 1)^*$

عبارت‌های منظم

- تعریف منظم
- نامگذاری عبارت‌های منظم برای استفاده ساده‌تر در سایر عبارت‌ها
- مثال: تعریف منظم برای شناسه‌ها در زبان C

$$
\begin{aligned}
\mathit{letter\_} &\rightarrow \texttt{A} \mid \texttt{B} \mid \cdots \mid \texttt{Z} \mid \texttt{a} \mid \texttt{b} \mid \cdots \mid \texttt{z} \mid \texttt{\_} \\
\mathit{digit} &\rightarrow \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{9} \\
\mathit{id} &\rightarrow \mathit{letter\_} \; (\; \mathit{letter\_} \mid \mathit{digit} \;)^*
\end{aligned}
$$

- مثال: تعریف منظم برای اعداد بدون علامت (unsigned)

$$
\begin{aligned}
\mathit{digit} &\rightarrow \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{9} \\
\mathit{digits} &\rightarrow \mathit{digit} \; \mathit{digit}^* \\
\mathit{optionalFraction} &\rightarrow \texttt{.} \; \mathit{digits} \mid \epsilon \\
\mathit{optionalExponent} &\rightarrow (\; \texttt{E} \; (\; \texttt{+} \mid \texttt{-} \mid \epsilon \;) \; \mathit{digits} \;) \mid \epsilon \\
\mathit{number} &\rightarrow \mathit{digits} \; \mathit{optionalFraction} \; \mathit{optionalExponent}
\end{aligned}
$$

عبارت‌های منظم

- برخی عملگرهای تکمیلی در عبارت‌های منظم
  - عملگر یک یا بیشتر از یک عبارت منظم: $\mathbf{r^+}$ (بستار مثبت)
  - عملگر صفر یا یک از یک عبارت منظم: **r?**
  - عملگر یک دسته از نشانه‌های الفبا:

$$\mathbf{a_1 \mid a_2 \mid \ldots \mid a_n \rightarrow [a_1 a_2 \ldots a_n]}$$

  - عملگر یک دسته پشت سر هم (دنباله) از نشانه‌های الفبا:

$$\mathbf{a \mid b \mid \ldots \mid z \rightarrow [a\text{-}z]}$$

- مورد استفاده در ابزارهای تولید تحلیلگر واژه‌ای

عبارت‌های منظم

- توصیف الگوی واژه‌ها
- مثال: برخی نمادها در زبان برنامه‌نویسی Pascal

$$
\begin{array}{rcl}
digit & \rightarrow & \texttt{[0-9]} \\
digits & \rightarrow & digit^{+} \\
number & \rightarrow & digits\ (\texttt{.}\ digits)?\ (\ \texttt{E}\ \texttt{[+-]}?\ digits\ )? \\
letter & \rightarrow & \texttt{[A-Za-z]} \\
id & \rightarrow & letter\ (\ letter \mid digit\ )^{*} \\
if & \rightarrow & \texttt{if} \\
then & \rightarrow & \texttt{then} \\
else & \rightarrow & \texttt{else} \\
relop & \rightarrow & \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=} \mid \texttt{=} \mid \texttt{<>}
\end{array}
$$

- توصیف فاصله‌های خالی

$$ws \rightarrow (\ \textbf{blank} \mid \textbf{tab} \mid \textbf{newline}\ )^{+}$$

عبارت‌های منظم

- توصیف الگوی واژه‌ها (ادامه)
- مثالی از دسته نمادهای مرتبط با هر واژه و ویژگی‌های آنها

LEXEMES	TOKEN NAME	ATTRIBUTE VALUE
Any *ws*	–	–
if	**if**	–
then	**then**	–
else	**else**	–
Any *id*	**id**	Pointer to table entry
Any *number*	**number**	Pointer to table entry
<	**relop**	LT
<=	**relop**	LE
=	**relop**	EQ
<>	**relop**	NE
>	**relop**	GT
>=	**relop**	GE

عبارت‌های منظم

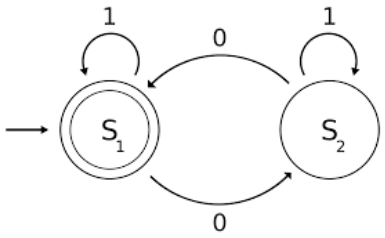
- سوال: کدام یک از عبارت‌های منظم، زبان زیر را نشان می‌دهد؟
- زمان به شکل "04:13PM"، که در آن دقیقه همیشه شامل دو رقم ولی ساعت می‌تواند تک رقمی باشد

- ☐ (0 | 1)?[0-9]:[0-5][0-9](AM | PM)
- ☐ ((0 | ε)[0-9] | 1[0-2]):[0-5][0-9](AM | PM)
- ☐ (0*[0-9] | 1[0-2]):[0-5][0-9](AM | PM)
- ☐ (0?[0-9] | 1(0 | 1 | 2):[0-5][0-9](A | P)M

رئوس مطالب

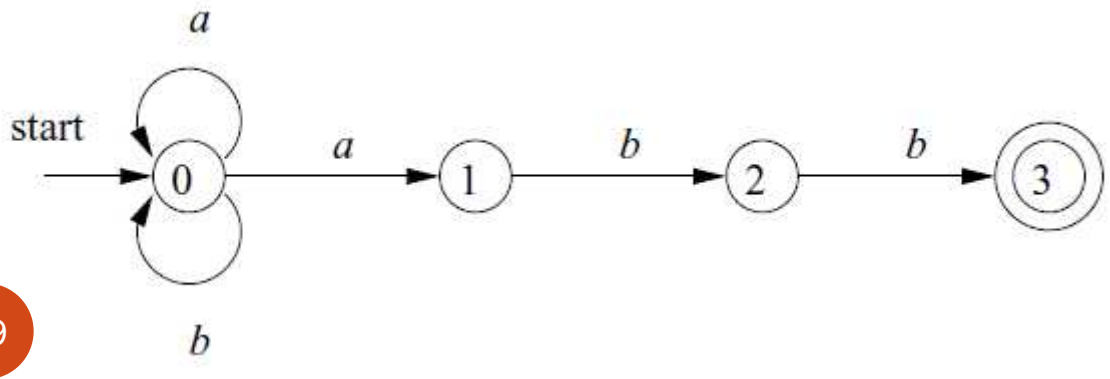
- مثال‌های اولیه
  - آشنایی با مفاهیم مرتبط با تحلیل واژه‌ای
- توصیف واژه‌ها
  - عبارت‌های منظم
- شناسایی واژه‌ها
  - ماشین‌های متناهی

شناسایی واژه‌ها



ماشین‌های متناهی

- نیاز به ساز و کاری برای شناسایی واژه‌ها در دنباله نویسه‌های ورودی
  - عبارت‌های منظم برای توصیف واژه‌ها بکار می‌روند
  - استفاده از **ماشین‌های متناهی** (finite automata)
  - تعریف ماشین‌های متناهی با پنج مؤلفه
    - الفبای ورودی
    - مجموعه متناهی از حالات
    - حالت اولیه
    - مجموعه حالات نهایی
    - تابع انتقال
  - نحوه نمایش

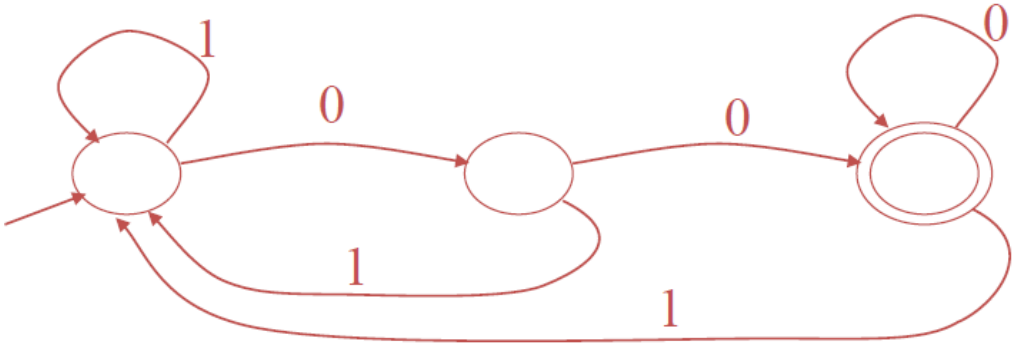


ماشین‌های متناهی

- نحوه کار ماشین‌های متناهی
  - وضعیت یک ماشین متناهی: «حالت فعلی، ورودی فعلی»
  - انتقال از یک حالت به حالت دیگر با دیدن یک ورودی
  - شرط پذیرش یک رشته
    - با رسیدن به انتهای رشته در یکی از حالات نهایی باشیم (پیدا کردن یک مسیر)
    - عدم پذیرش رشته در صورت عدم امکان پردازش رشته ورودی
- زبان ماشین متناهی
  - مجموعه تمام رشته‌های پذیرفته شده توسط آن ماشین
  - یک زبان منظم
    - برابر با زبان عبارت‌های منظم

ماشین‌های متناهی

- سوال: زبان کدامیک از عبارت‌های منظم زیر برابر با زبان ماشین متناهی داده شده است؟



- (0 | 1)*
- (1* | 0)(1 | 0)
- 1* | (01)* | (001)* | (000*1)*
- (0 | 1)*00

ماشین‌های متناهی

- ماشین متناهی معین (DFA)
  - به ازای هر نشانه الفبا فقط یک انتقال از هر حالت ممکن است
  - امکان انتقال فقط با دیدن نشانه‌های ورودی (حرکت تهی [ϵ] وجود ندارد)
  - وجود مسیر یکتا در انتقال حالات ماشین با دیدن نشانه‌های ورودی
    - ماشین در هر زمان فقط در یک حالت است

- پیاده‌سازی بصورت جداول جستجو (lookup tables)
  - سطرها متناظر با حالت‌ها
  - ستون‌ها متناظر با نشانه‌های الفبا
  - هر یک از خانه‌های جدول نتیجه تابع انتقال

STATE	a	b
A	B	A
B	B	D
D	B	E
E	B	A

ماشین‌های متناهی

- شبیه‌سازی DFA (simulation)
  - الگوریتم بسیار سریع و کارآمد

```
s = s0;
c = nextChar();
while ( c != eof ) {
        s = move(s, c);
        c = nextChar();
}
if ( s is in F ) return "yes";
else return "no";
```

جستجو در جدول

- دارای پیچیدگی زمانی $O(|x|)$ برای رشته x
  - در صورتی که زمان جستجو در جدول ثابت فرض شود

ماشین‌های متناهی

- ماشین متناهی نامعین (NFA)
  - امکان وجود چندین انتقال به ازای هر نشانه الفبا در هر حالت
  - امکان انتقال بدون دیدن نشانه‌های ورودی (حرکت تهی [ϵ])
  - امکان وجود چندین مسیر انتقال با دیدن نشانه‌های ورودی
    - ماشین در هر زمان ممکن است در چندین حالت باشد (حق انتخاب)
  - زبان ماشین‌های معین و نامعین برابر است
    - افزودن حرکت ϵ فقط باعث سادگی بیان می‌شود

- مفهوم بستار ϵ (ϵ-closure) برای یک زیرمجموعه از حالات
  - تمام حالاتی که با یک یا بیشتر حرکت ϵ از این زیرمجموعه قابل دستیابی است

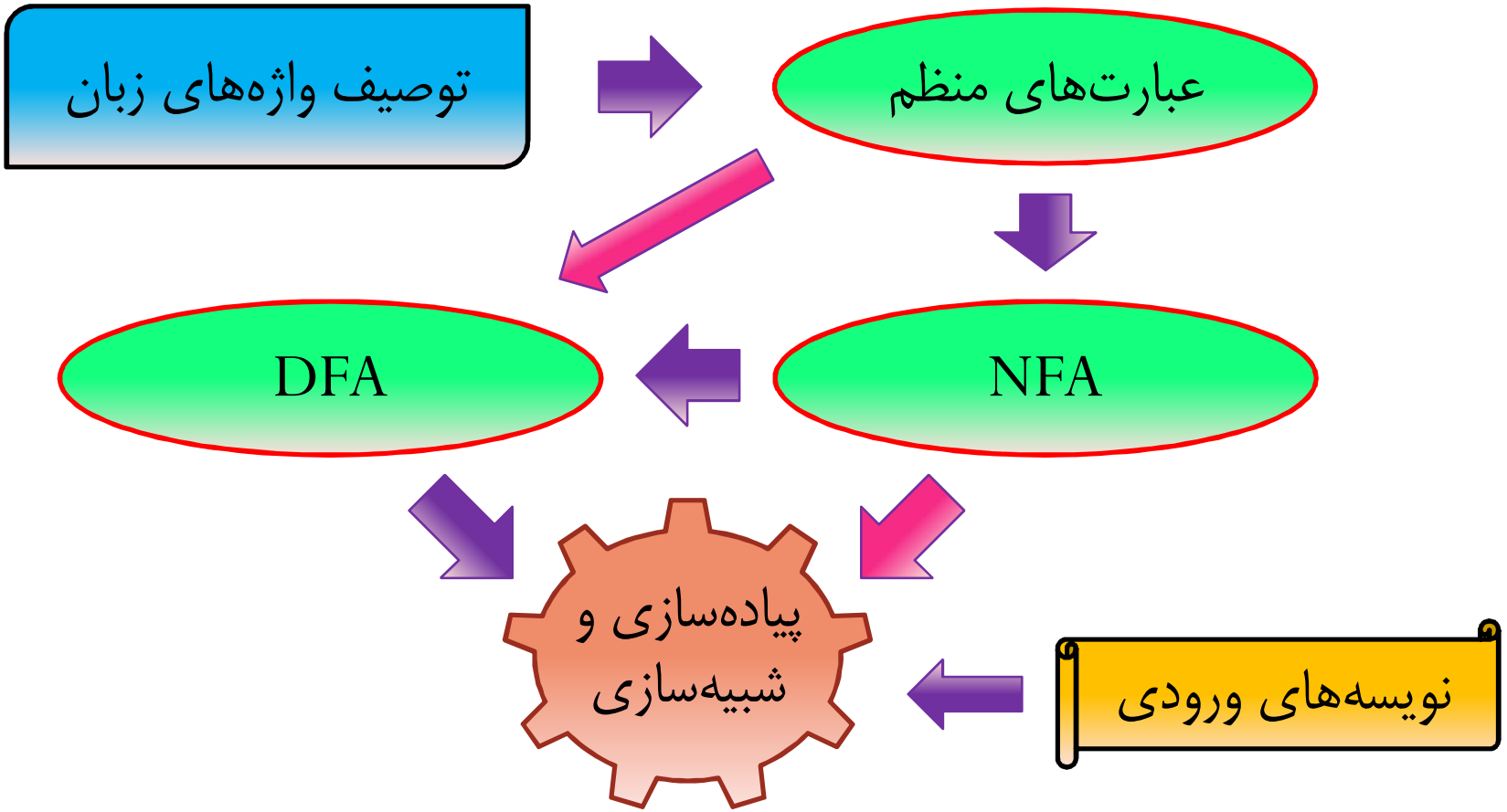
ماشین‌های متناهی

```
S = ϵ-closure(s0);
c = nextChar();
while ( c != eof ) {
        S = ϵ-closure(move(S, c));
        c = nextChar();
}
if ( S ∩ F != ∅ ) return "yes";
else return "no";
```

- شبیه‌سازی NFA
  - نیاز به محاسبه بستار ϵ
  - نیاز به نگهداری مجموعه حالات فعلی
- دارای پیچیدگی زمانی O(|x|(n+m)) برای رشته x
  - n تعداد حالات ماشین و m تعداد کل انتقال‌ها (یال‌ها) است
- نسبت به شبیه‌سازی DFA دارای سرعت کمتری است
  - اما تعداد حالت‌های DFA می‌تواند بیشتر از NFA معادل آن باشد
  - نوعی مصالحه بین زمان و فضا در ماشین‌های معین و نامعین

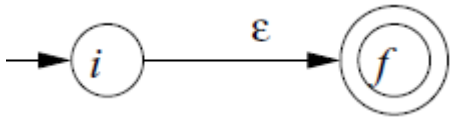
شناسایی واژه‌ها

- روند شناسایی واژه‌ها با استفاده از ماشین‌های متناهی

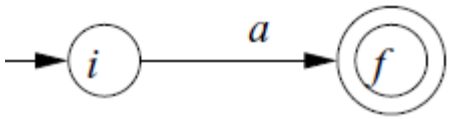


شناسایی واژه‌ها

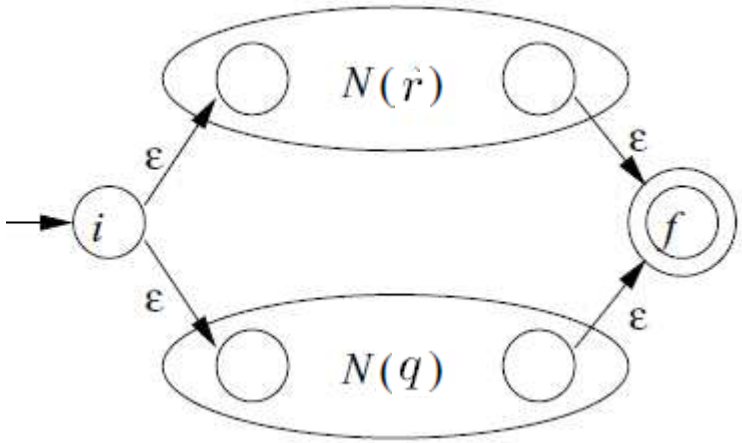
- تبدیل عبارت منظم به NFA
- الگوریتم بازگشتی McNaughton – Yamada – Thompson (MYT)
  - NFA معادل برای ϵ



  - NFA معادل برای نشانه‌های الفبا (a)

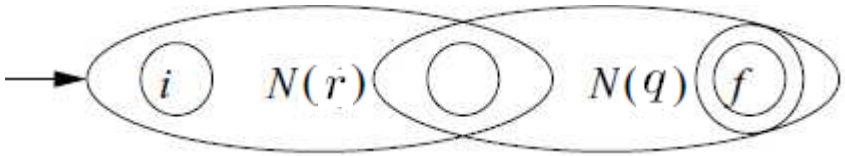


  - NFA معادل برای عبارت منظم r | q

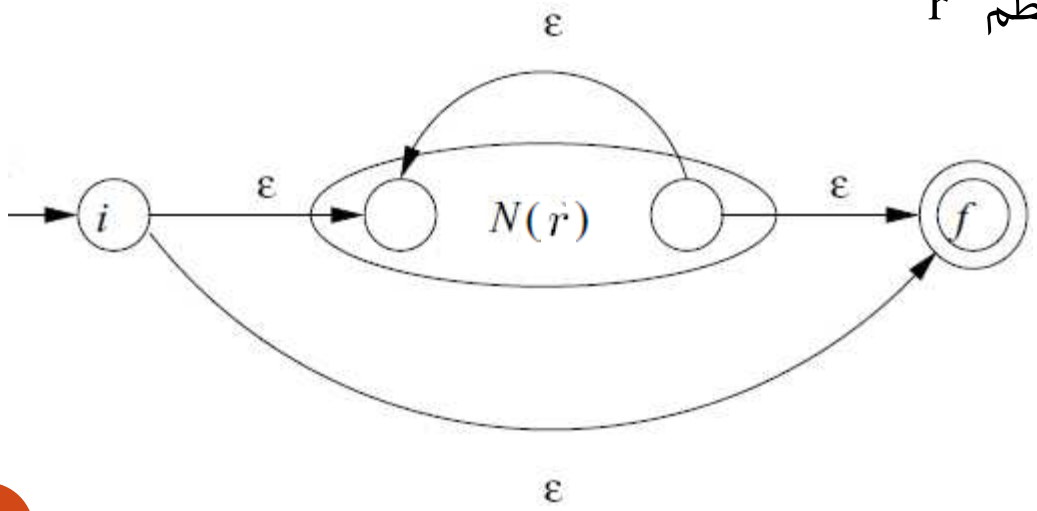


شناسایی واژه‌ها

- تبدیل عبارت منظم به NFA
- الگوریتم بازگشتی MYT (ادامه)
  - NFA معادل برای عبارت منظم rq

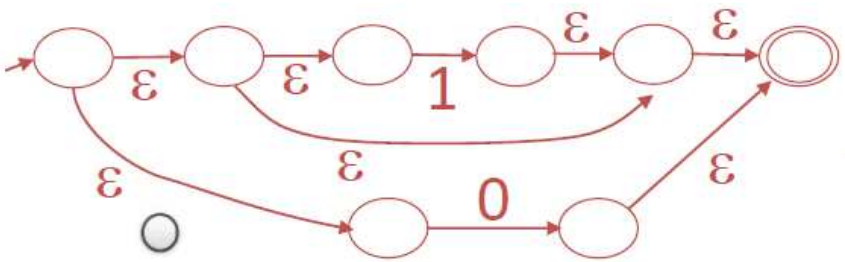


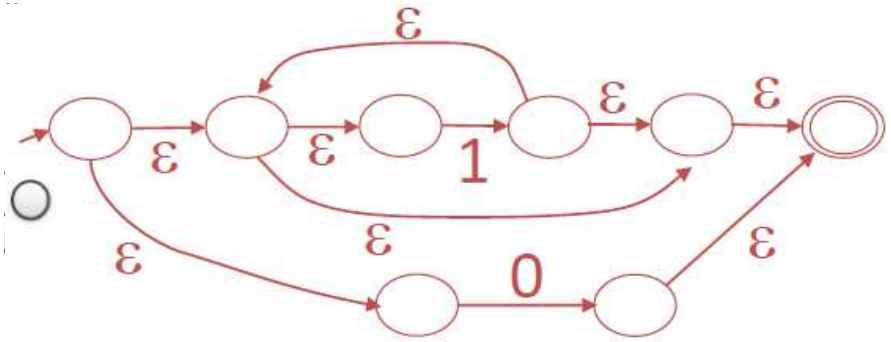
  - NFA معادل برای عبارت منظم r^*

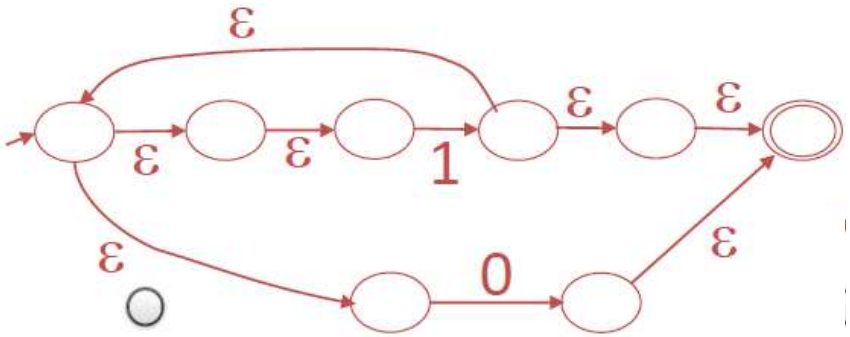


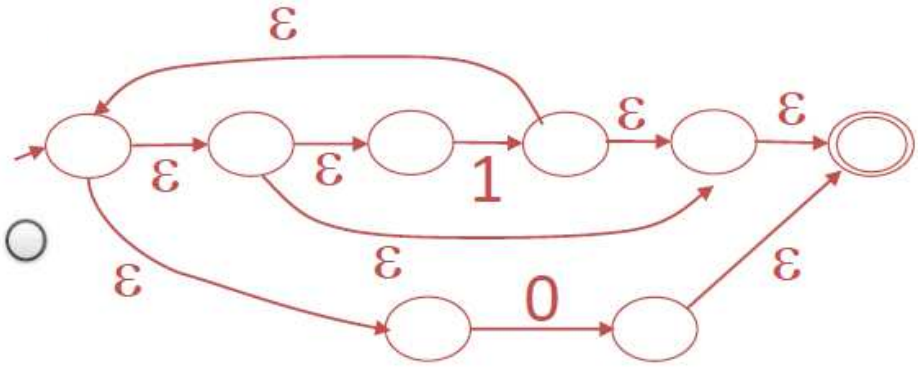
شناسایی واژه‌ها

- سوال: کدامیک از NFAهای زیر معادل عبارت منظم $0 | 1^*$ است؟



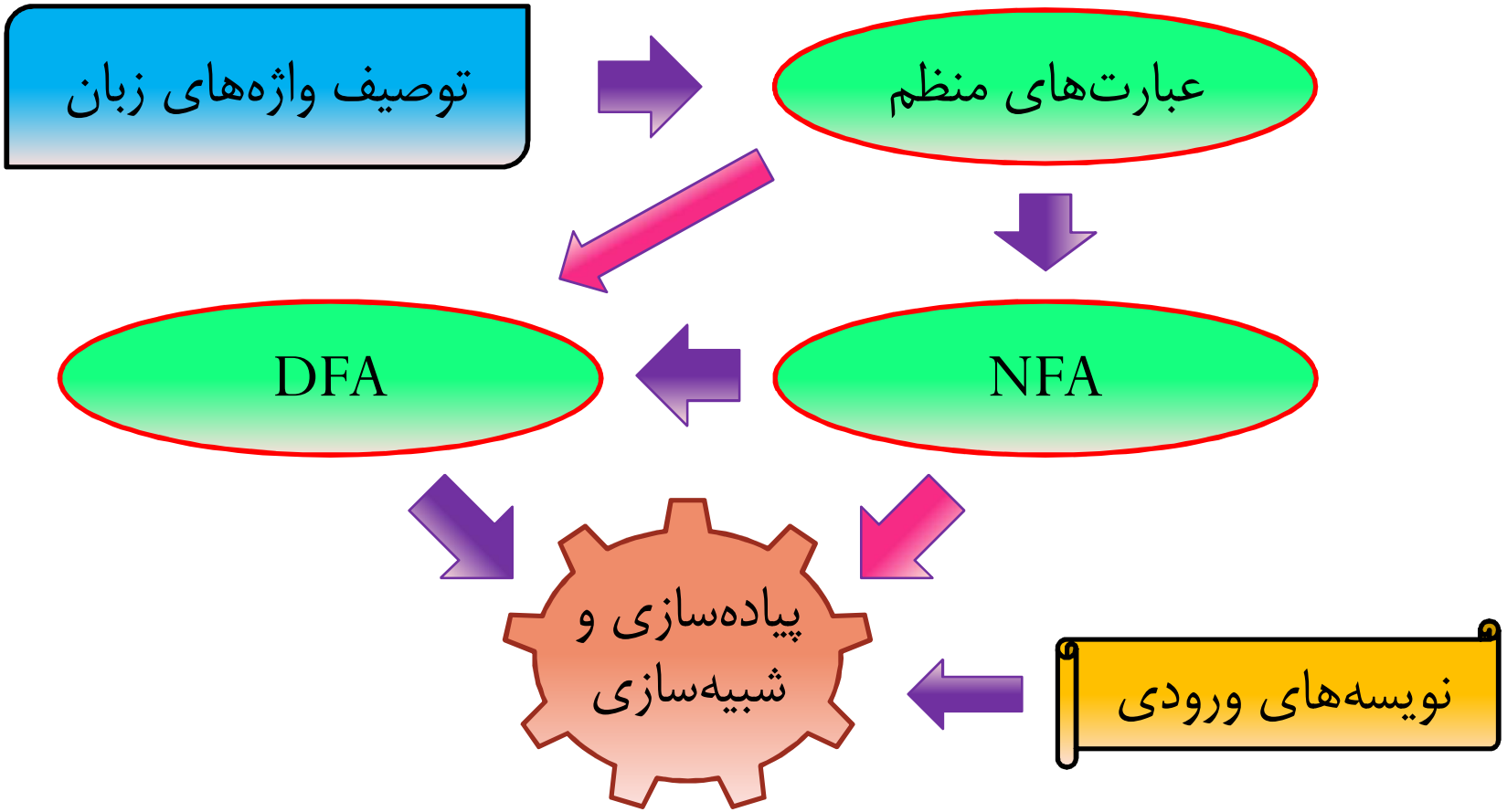






شناسایی واژه‌ها

- روند شناسایی واژه‌ها با استفاده از ماشین‌های متناهی



شناسایی واژه‌ها

- تبدیل NFA به DFA
  - NFA می‌تواند به صورت همزمان در چندین حالت باشد
    - هر وضعیت ماشین دربردارنده زیرمجموعه‌ای از حالات است
    - تعداد زیرمجموعه‌های ممکن (غیرتهی) برای n حالت: 2^n-1
      - یک مجموعه با اندازه نمایی ولی متناهی
    - زیرمجموعه حالات فعلی با یک نشانه ورودی، به زیرمجموعه‌ای یکتا نگاشته می‌شود
      - عملکرد معین
  - هر حالت از DFA معادل زیرمجموعه‌ای از حالات NFA در نظر گرفته می‌شود
    - برابر با بستار ϵ زیرمجموعه‌ای از حالات NFA

شناسایی واژه‌ها

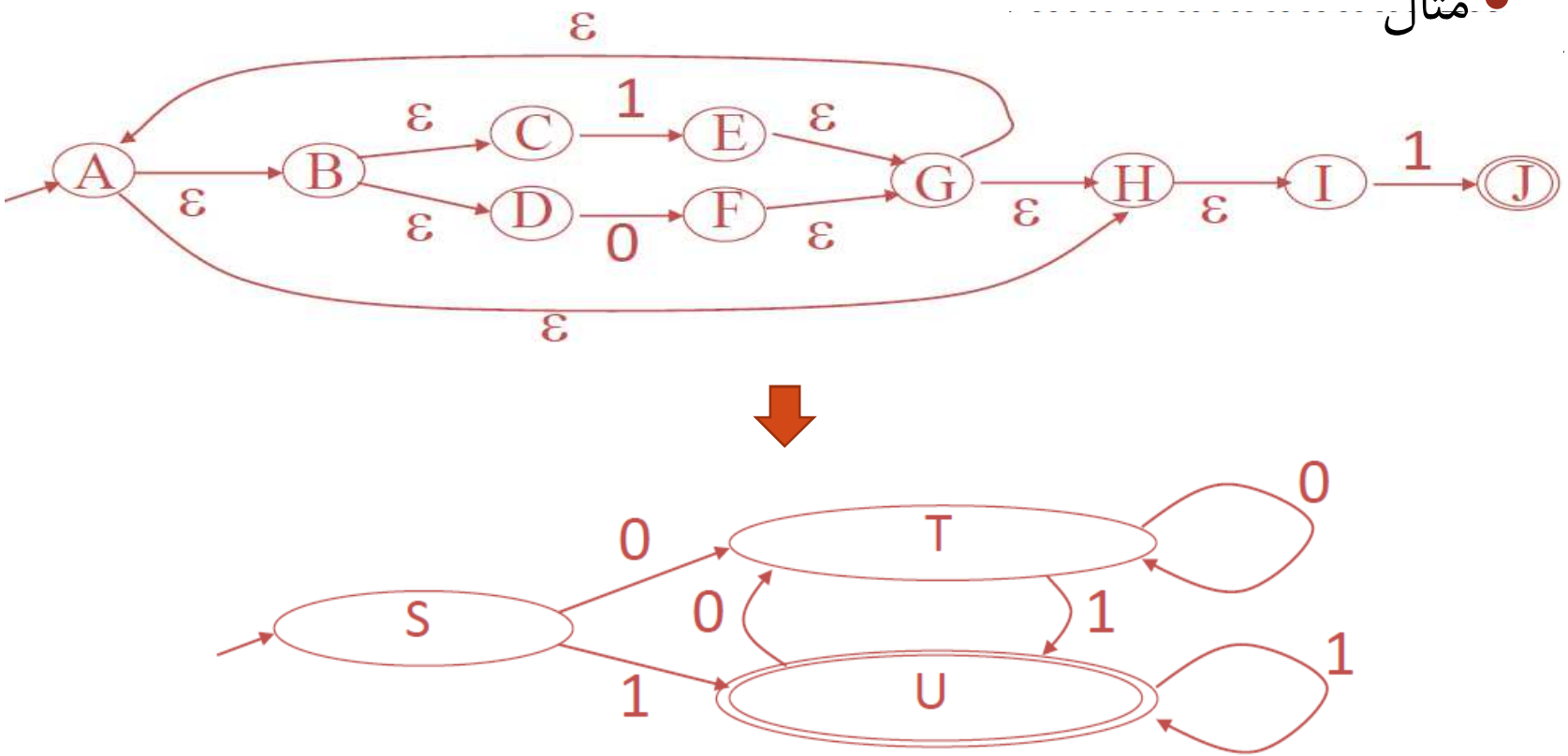
- تبدیل NFA به DFA (ادامه)
  - حالت اولیه DFA: بستار ϵ حالت اولیه NFA
  - نحوه ساخت DFA معادل

```
initially, ϵ-closure(s0) is the only state in Dstates, and it is unmarked;
while ( there is an unmarked state T in Dstates ) {
        mark T;
        for ( each input symbol a ) {
                U = ϵ-closure(move(T, a));
                if ( U is not in Dstates )
                        add U as an unmarked state to Dstates;
                Dtran[T, a] = U;
        }
}
```

  - حالت نهایی DFA: هر حالتی که دارای حداقل یک حالت نهایی NFA باشد

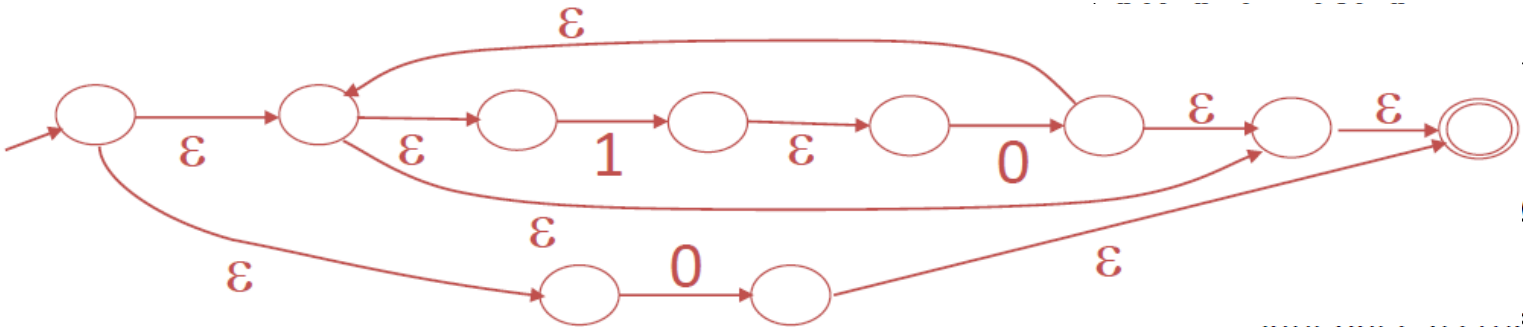
شناسایی واژه‌ها

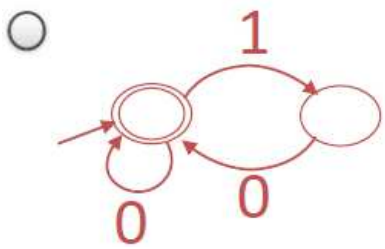
- تبدیل NFA به DFA (ادامه)
  - مثال

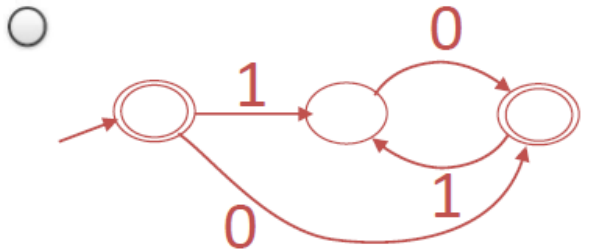


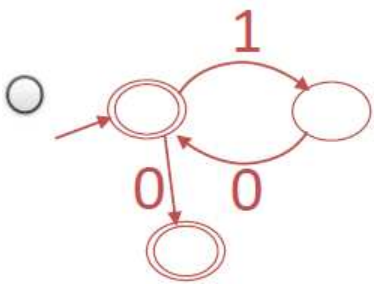
شناسایی واژه‌ها

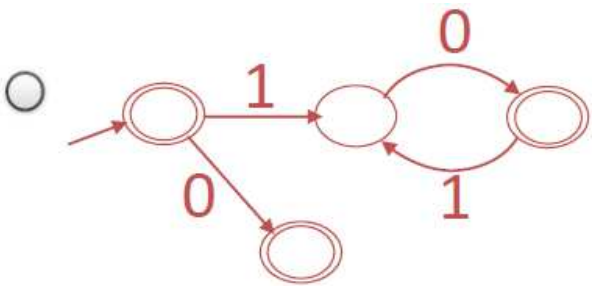
- سوال: کدامیک از DFAهای زیر معادل NFA داده شده است؟











شناسایی واژه‌ها

- هزینه استفاده از انواع ماشین‌های متناهی
  - $|r|$: تعداد عملگرها و عملوندهای موجود در عبارت منظم r
  - حداکثر تعداد حالات NFA بدست آمده از الگوریتم MYT محدود است
    - $n \leq 2|r|$ و $m \leq 4|r|$
  - هزینه ساخت DFA معادل برای یک NFA: $O(|r|^2 s)$
    - s تعداد حالت‌های DFA است

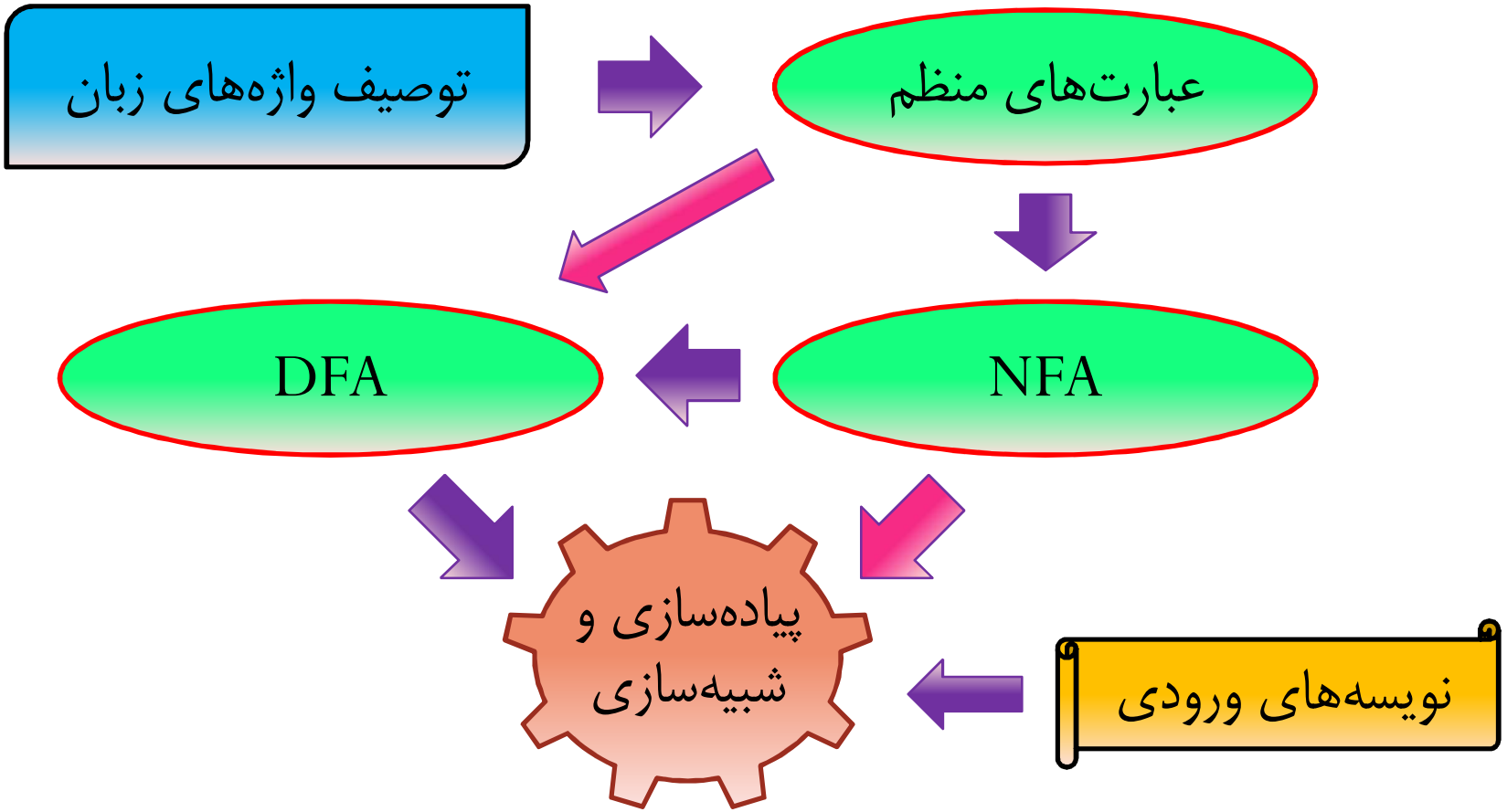
AUTOMATON	INITIAL	PER STRING
NFA	$O(\|r\|)$	$O(\|r\| \times \|x\|)$
DFA typical case	$O(\|r\|^3)$	$O(\|x\|)$
DFA worst case	$O(\|r\|^2 2^{\|r\|})$	$O(\|x\|)$

شناسایی واژه‌ها

- کدام ماشین را استفاده کنیم؟
  - در NFA تعداد عملیات قابل انجام به ازای هر نویسه ورودی زیاد است
  - در بدترین وضعیت تعداد حالت‌های DFA از NFA معادل آن به صورت نمایی بیشتر است
  - پرداخت هزینه تبدیل NFA به DFA؟
    - در صورت بکارگیری زیاد ماشین تبدیل به DFA مقرون به صرفه خواهد بود
      - مثلاً در کامپایلرها
    - در صورت بکارگیری نادر ماشین، شبیه‌سازی مستقیم NFA منطقی‌تر است
      - مثلاً در جستجو برای فایل‌ها با استفاده از یک الگو
  - بکارگیری رویکرد ترکیبی؟

شناسایی واژه‌ها

- روند شناسایی واژه‌ها با استفاده از ماشین‌های متناهی



پیاده‌سازی تحلیلگر واژه‌ای

تحلیلگر واژه‌ای

- مؤلفه‌های اصلی در پیاده‌سازی تحلیلگر واژه‌ای
  - روش بافرینگ برای خواندن ورودی
  - ماشین متناهی حاوی پیاده‌سازی توصیف واژه‌های ممکن در ورودی
  - تعیین عملیات متناظر با هر یک از حالت‌های پایانی

تحلیلگر واژه‌ای

- مراحل پیاده‌سازی تحلیلگر واژه‌ای

1) توصیف الگوی واژه‌های مرتبط با هر دسته نماد
   - استفاده از عبارت‌های منظم (r_i)
2) توصیف کلی واژه‌های زبان
   - ترکیب فصلی تمام الگوها در یک عبارت منظم: $r \longrightarrow r_1 \mid r_2 \mid r_3 \mid \ldots$
3) بررسی تعلق پیشوند دنباله نویسه‌های ورودی به زبان
   - اگر $x_1 \ldots x_n$ دنباله نویسه‌های ورودی باشد نیاز به بررسی: $x_1 \ldots x_i \in L(r)$
   - اعلام دسته نماد مرتبط با r_i برای پیشوند شناسایی شده (و انجام عملیات تکمیلی)
4) حذف پیشوند شناسایی شده و تکرار مرحله قبلی برای باقیمانده دنباله ورودی
   - تکرار تا رسیدن به انتهای دنباله نویسه‌های ورودی

تحلیلگر واژه‌ای

- برخی نکات مطرح در پیاده‌سازی تحلیلگر واژه‌ای
  - تعلق پیشوندهایی با اندازه‌های متفاوت به زبان
    - در نظر گرفتن گرفتن طولانی‌ترین پیشوند قابل شناسایی
      - maximal munch
    - مثال: در نظر گرفتن یک نماد ’==‘ بجای دو نماد ’=‘

$$x_1 \ldots x_i \in L(r)$$
$$x_1 \ldots x_i \ldots x_j \in L(r)$$

  - تعلق پیشوند به بیش از یک الگو
    - مثال: کلیدواژه‌ها هم با الگوی دسته نماد خود و هم با الگوی شناسه‌ها تطبیق دارند
    - اولویت‌بندی الگوی دسته نمادها و اعلام دسته نمادی که دارای اولویت بیشتر است
      - مثلاً دسته نمادی که توصیف الگوی آن اول بیان شده باشد
  - عدم تعلق هیچ پیشوندی از ورودی به زبان
    - بروز خطا

شناسایی واژه‌ها

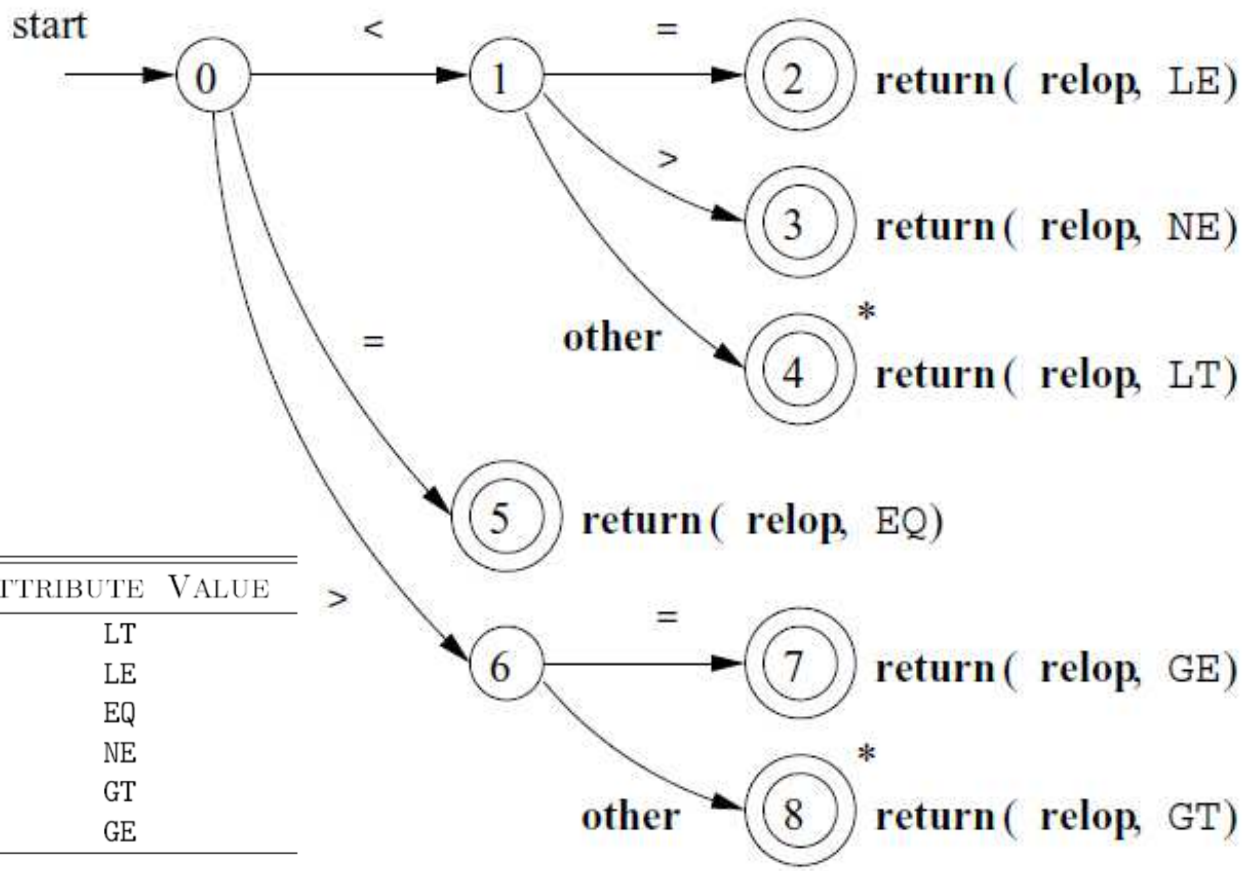
- مراحل شناسایی با استفاده از ماشین‌های متناهی

1) خواندن نویسه‌های ورودی و دنبال کردن انتقال حالات در ماشین متناهی
   - نویسه‌ها با مدیریت نشانه‌روهای مربوطه از بافر خوانده می‌شوند
2) بازگشت به آخرین حالت نهایی با توقف ماشین متناهی
   - توقف ماشین: رسیدن به بن‌بست در DFA یا زیرمجموعه تهی در NFA
   - بازگرداندن نویسه‌های جلوتر (lookahead) در بافر
3) اجرای عملیات مرتبط با حالت نهایی بدست آمده
   - اعلام دسته نماد
   - بروز رسانی جدول نشانه‌ها
   - مدیریت خطا
   - ...

تحلیلگر واژه‌ای

- مثال: شناسایی عملگرهای مقایسه‌ای

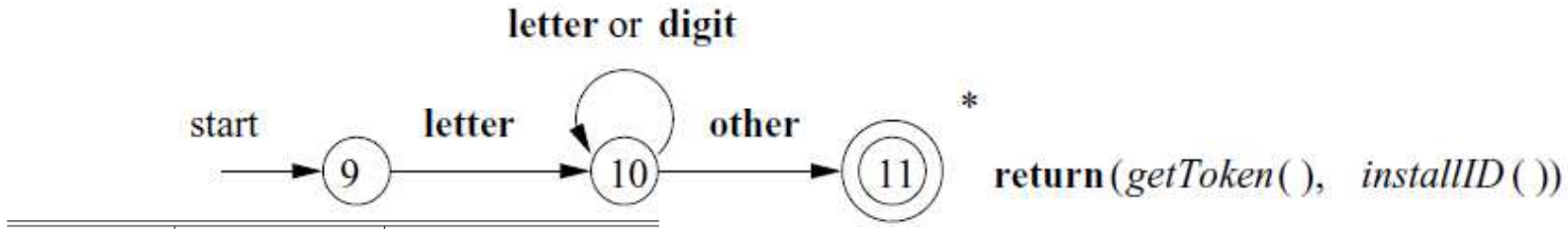
نمودارهای انتقال



LEXEMES	TOKEN NAME	ATTRIBUTE VALUE
<	relop	LT
<=	relop	LE
=	relop	EQ
<>	relop	NE
>	relop	GT
>=	relop	GE

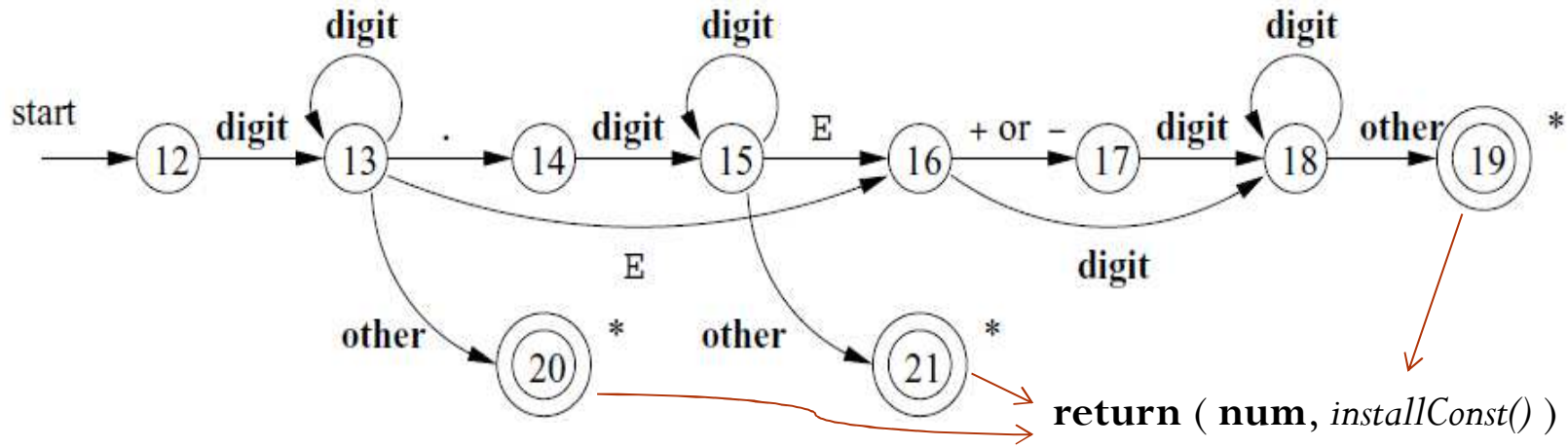
تحلیلگر واژه‌ای

- مثال: شناسایی شناسه‌ها و کلیدواژه‌ها



LEXEMES	TOKEN NAME	ATTRIBUTE VALUE
Any *ws*	–	–
if	if	–
then	then	–
else	else	–
Any *id*	id	Pointer to table entry
Any *number*	number	Pointer to table entry

- دو راهکار برای شناسایی از کلمات رزرو شده
- مثال: شناسایی اعداد بدون علامت



جمع‌بندی

- معرفی تحلیلگر واژه‌ای و وظایف آن
  - واژه‌ها، دسته‌های نماد، الگوها
- توصیف واژه‌ها با استفاده از عبارت‌های منظم
- شناسایی واژه‌ها با استفاده از ماشین‌های متناهی
  - شبیه‌سازی ماشین‌های معین و نامعین
  - تبدیل عبارت منظم به ماشین نامعین
  - تبدیل ماشین نامعین به معین
- طرح کلی پیاده‌سازی تحلیلگر واژه‌ای