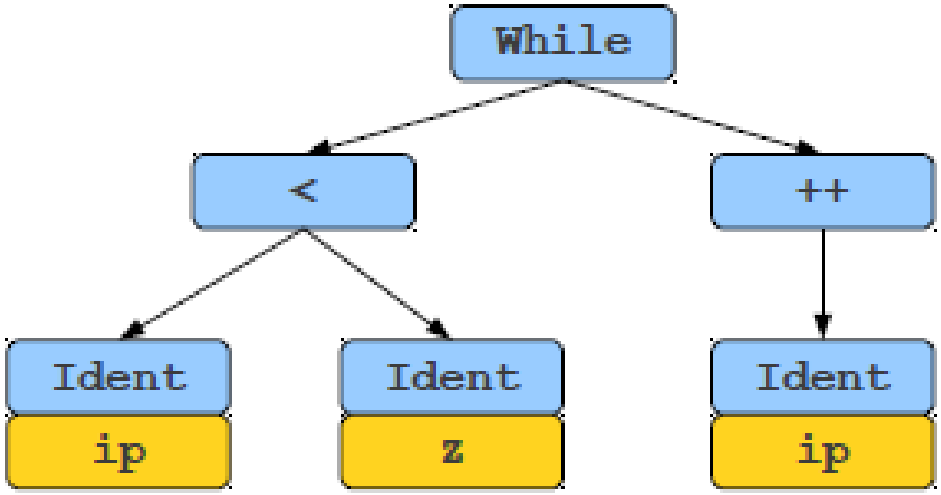
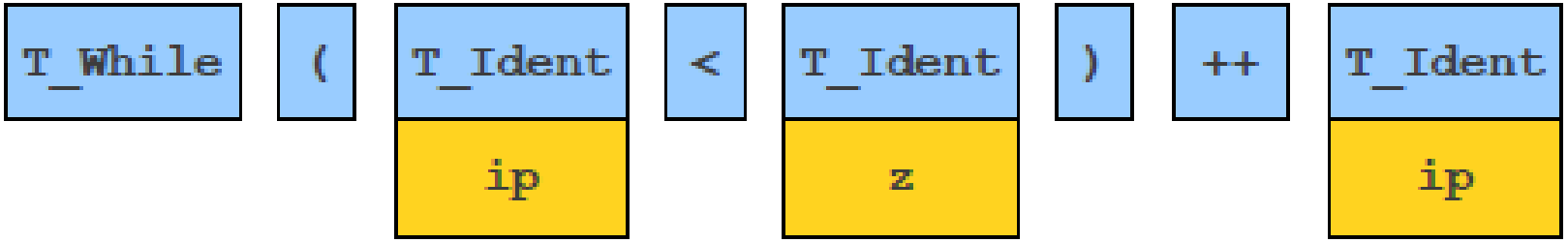
طراحی کامپایلرها

تحلیل لغوی (بخش 1)

قمرناز تدین تبریزی





w	h	i	l	e		(	i	p		<		z	)	\n	\t	+	+	i	p	;

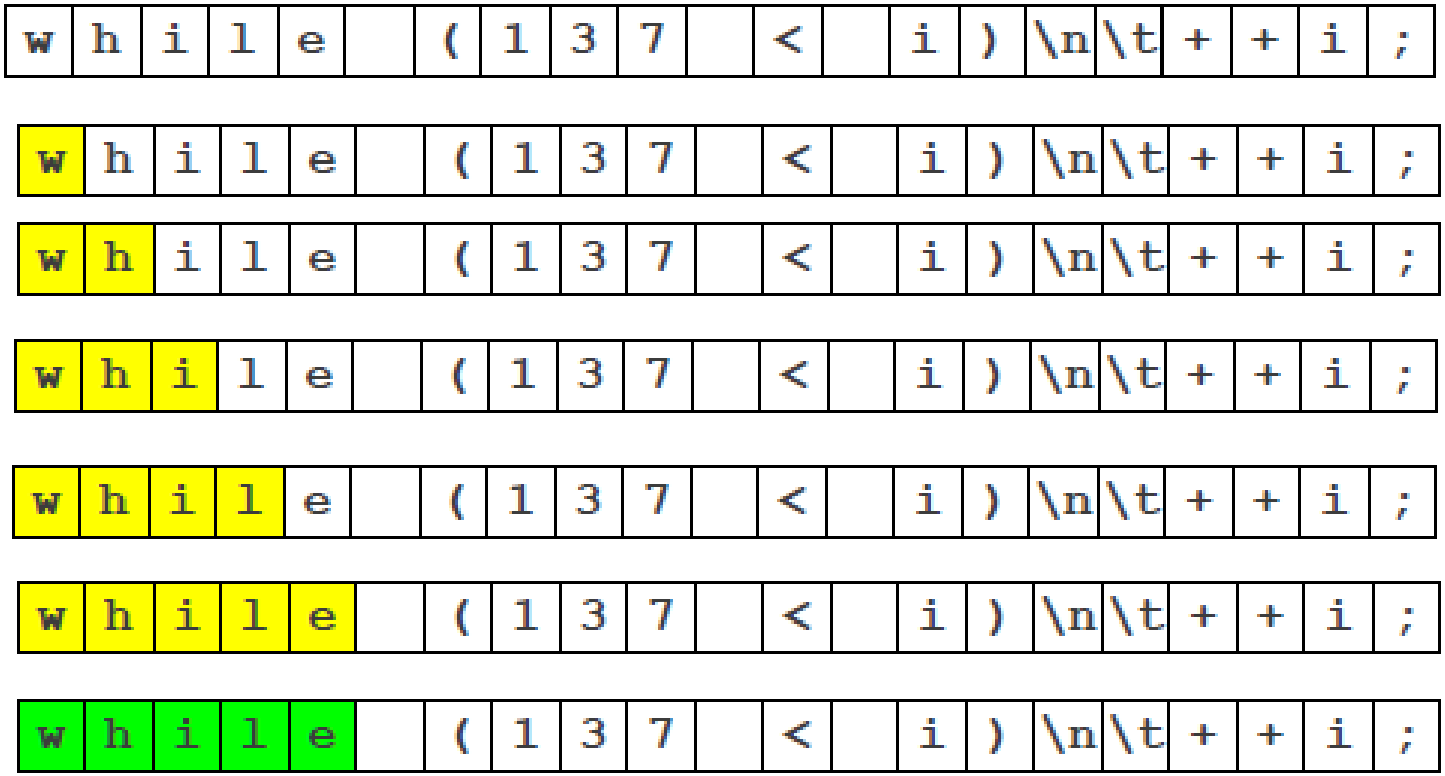
```
while (ip < z)
++ip;
```

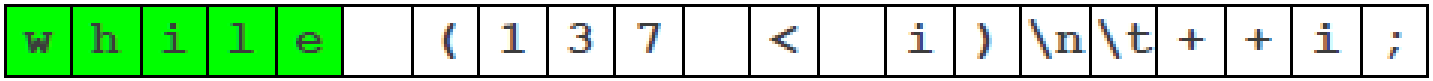
T_Do	[	T_For	]	=	T_New	T_IntConst
						0

d	o	[	f	o	r	]		=		n	e	w		0	;

```
do[for] = new 0;
```

پویش فایل مبدأ

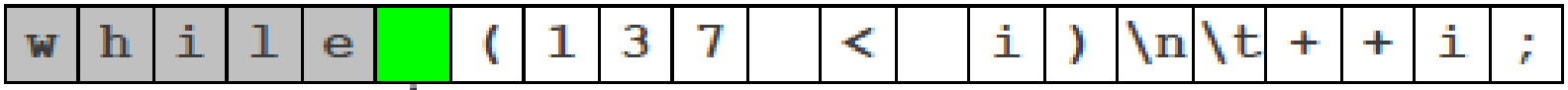




- این بخش از برنامه اصلی، که توکن بر اساس آن ساخته می شود، لغت (Lexeme) گفته می شود

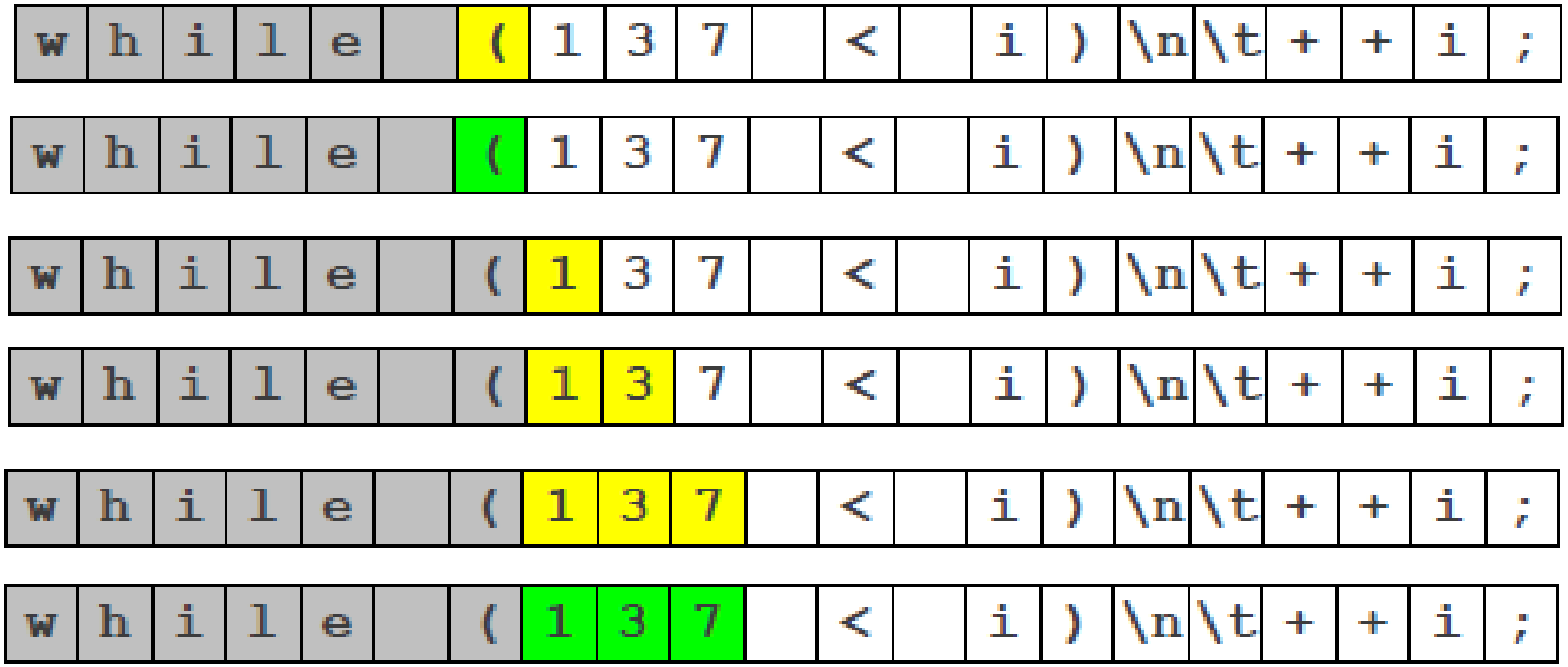
T_While

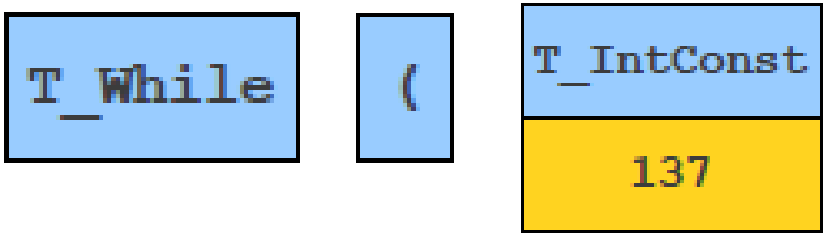
- توکن: یک نوع عددی شده است که نشان میدهد چه موجودیت منطقی از کد مبدأ خوانده شده است



- گاهی اوقات یک لغت ذخیره نمیشود و ندیده گرفته میشود. فضای خالی هیچ معنایی برای برنامه ندارد بنابراین نادیده گرفته می شود.

T_While





- برخی از توکن ها خصیصه ای دارند که اطلاعات اضافه درباره توکن را نگهداری میکند.

هدف از تحلیل لغوی

- تبدیل از توصیف فیزیکی برنامه به دنباله ای از توکنها
  - هر توکن یک بخش منطقی از فایل مبدأ را نشان میدهد مثل یک کلمه کلیدی، نام متغیر و ...
- هر توکن با یک لغت مرتبط است.
  - متن واقعی توکن مثل 137, "int", ...
- هر توکن میتواند خصیصه ای داشته باشد که از متن به دست می آید مثل مقدار عدد
- دنباله توکن ها در مرحله بعد در پارسر به کار می رود.

انتخاب توکن

- چه توکنهایی مفید هستند؟

```
for (int k = 0; k < myArray[5]; ++k) {
cout << k << endl;
}
```

For {
int }
<< ;
= <
([
)]
++
Identifier
IntegerConstant

انتخاب توکن های خوب

- بسیار وابسته به زبان است.
- معمولا:
  - کلمات کلیدی، توکن خودشان را دارند.
  - هر نماد نقطه گزاری، توکن خودش را دارد.
  - لغتهای نمایشگر شناسه ها، ثابتهای عددی، رشته ها و غیره با هم گروه بندی میشوند.
  - اطلاعات نامربوط کنارگذاشته می شوند (فضای خالی، توضیحات)

پیچیدگیها

- در fortran همه فضاهای خالی حذف می شوند
- **DO 5 I = 1,25**
- **DO5I = 1.25**

- تعیین این که ورودی کجا تفکیک شود مشکل است.

پیچیدگیها

- در زبان PL/1 کلمات کلیدی می توانند به عنوان شناسه استفاده شوند.
- IF THEN THEN THEN = ELSE; ELSE ELSE = IF
- **IF THEN THEN THEN = ELSE; ELSE ELSE = IF**
  - برچسب گزاری لغات مشکل است

چالش ها

- چگونه تعیین کنیم کدام لغات با هر توکن مرتبط هستند.
- اگر چند روش برای پویش ورودی وجود دارد چگونه بدانیم کدام را انتخاب کنیم.
- چگونه میتوان موارد فوق را با کارایی درنظر گرفت

ارتباط لغات و توکنها

- توکنها روشی برای گروه بندی لغات فراهم میکنند.
- برخی توکنها ممکن است فقط با یک لغت مرتبط شوند.
  - توکنها برای کلمات کلیدی مثل if و While میتوانند جزء این دسته باشند.
- بعضی ت.کنها ممکن است با لغات زیادی مرتبط شوند:
  - نام متغیرها، اعداد، رشته ها و...

مجموعه لغات

- ارتباط لغات با هر توکن
- توکن "number" با مجموعه {0,1,2, … ,10,11,12,…}
- توکن "string" با مجموعه {" ","a","b", … }
- توکن While با مجموعه {while}

Formal Languages

- زبانهای رسمی، مجموعه ای از رشته ها هستند.
- بسیاری از زبانهای نامتناهی دارای تعریف متناهی هستند.
  - تعریف زبان با استفاده از اتوماتا
  - تعریف زبان با استفاده از گرامر
  - تعریف زبان با استفاده از عبارت باقاعده

عبارت باقاعده

- مجموعه ای از توصیفها برای تعریف زبانهای معینی است.
- توصیف فشرده و قابل فهمی از زبان
- اساسی برای بسیاری سیتمهای نرم افزاری مثل Flex که بعدا بررسی خواهد شد

عبارات باقاعده اتمیک

- نماد □ یا ؐ رشته تهی را نشان میدهد.
- برای هر نماد a، a یک عبارت باقاعده است که با کاراکتر a منطبق می شود.

عبارات باقاعده ترکیبی

- اگر R1 و R2 عبارت باقاعده باشند، R1R2(الحاق)، R1|R2 (اجتماع) و R1* بستار ستاره ای آنها هم باقاعده است.
- اولویت عملگرها:

$$(R)$$

$$R^*$$

$$R_1 R_2$$

$$R_1 \mid R_2$$

مثال

- الفبای 0و1:
- رشته های شامل 00 :(0|1)*00(0|1)*

11011100101
0000
1111101111110011111

مثال

- الفبای 0و1:
- رشته های با طول 4:

$(0|1)(0|1)(0|1)(0|1)$ $(0|1)\{4\}$

0000
1010
1111
1000

مثال

- الفبای 0و1:
- رشته هایی که حداکثر یک صفر دارند:

$1^*(0 \mid \varepsilon)1^*$ $1^*0?1^*$

11110111
111111
0111
0

مثال کاربردی

- فرض کنید الفبا شامل a , @ و . باشد که a به معنی هر حرف الفبا باشد.
- عبارت باقاعده برای آدرس ایمیل:

$$aa^* \ (.aa^*)^* \ @ \ aa^*.aa^* \ (.aa^*)^*$$

$$a^+(.a^+)^*@a^+(.a^+)^+$$

cs143@cs.stanford.edu
first.middle.last@mail.site.org
barack.obama@whitehouse.gov

مثال کاربردی

- الفبا: همه کاراکترهای اسکی
- عبارت باقاعده برای اعداد زوج

(+|-)?(0|1|2|3|4|5|6|7|8|9)*(0|2|4|6|8)

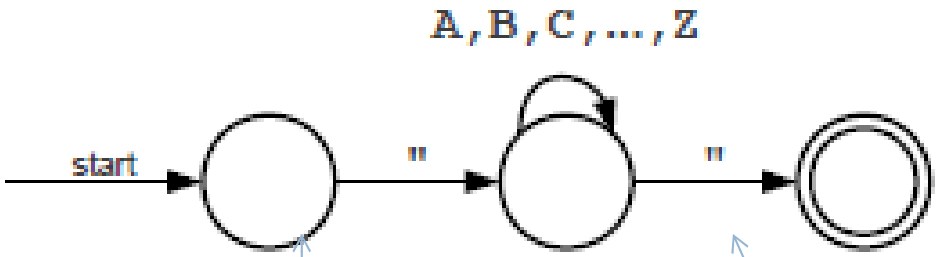
(+|-)?[0-9]*[02468]

42
+1370
-3248
-9999912

پیاده سازی عبارتهای باقاعده

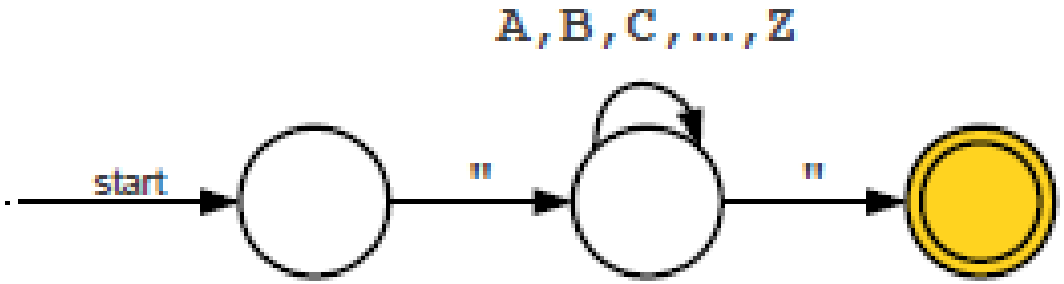
- عبارات باقاعده با اتوماتای متنهای قابل پیاده سازی هستند.
- دو نوع اصلی:
  - NFA(Nondeterministic Finite Automata)
  - DFA(Deterministic Finite Automata)

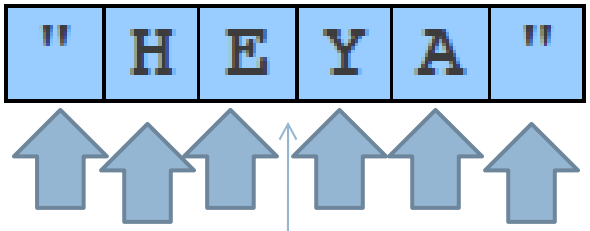
یک اتوماتای ساده



کمانها تغییر وضعیت هستند	هر دایره یک وضعیت اتوماتا است.

یک اتوماتای ساده، ادامه....

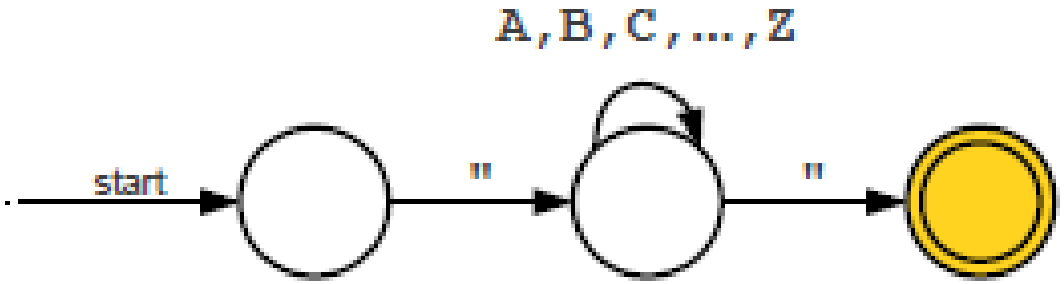


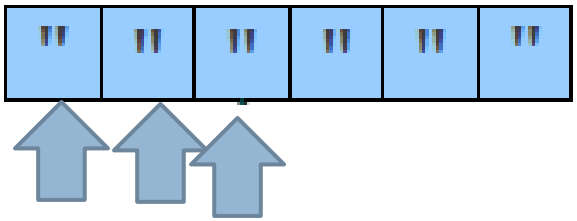


حالت پدیرش

رشته ورودی که اتوماتا تصمیم میگیرد آن را بپذیرد یا خیر

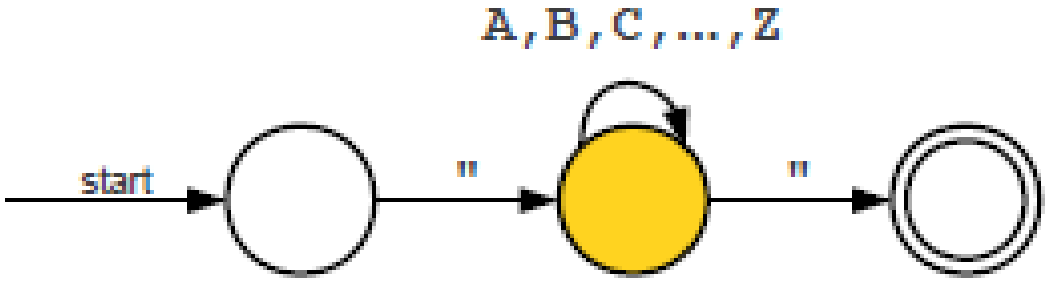
یک اتوماتای ساده، ادامه....

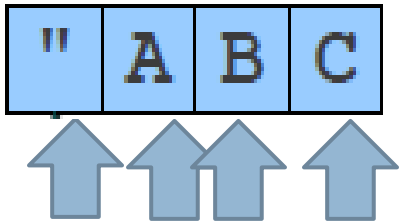




هیچ تغییر حالتی وجود ندارد. ورودی رد می شود.

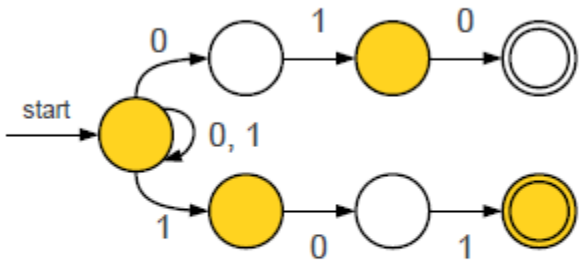
یک اتوماتای ساده، ادامه....





این جالت، پذیرش نیست پس اتوماتا رشته را رد میکند.

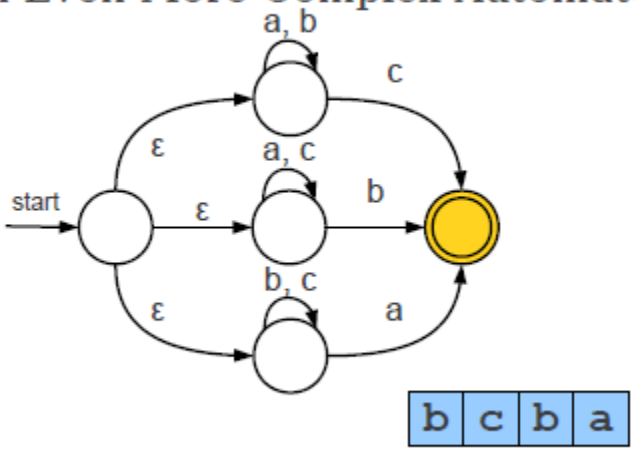
A More Complex Automaton



0	1	1	1	0	1

پذیرش

An Even More Complex Automaton



گذر ε به صورت خودکار
و بدون استفاده از
ورودی دنبال می شود

شبیه سازی NFA

- وضعیتها دنبال می شوند. پردازش ازوضعیت شروع و هر وضعیتی که با تغییرحالت ε قابل دسترسی باشد آغاز می شود.
- برای هر کاراکتر ورودی:
  - مجموعه ای از وضعیتهای بعدی نگهداری می شوند. (مقدار اولیه تهی است)
  - برای هر وضعیت جاری:
    - همه انتقالهایی که برچسب آنها حرف جاری است دنبال می شوند.
    - این وضعیتها به مجموعه وضعیتهای جدید اضافه می شوند.
  - هر وضعیتی که با حرکت ε قابل دسترسی باشد به مجموعه وضعیت های بعدی اضافه میشود.
- پیچیدگی: $O(mn^2)$ برای رشته هایی به طول m و اتوماتایی با n وضعیت

جمع بندی

- هر عبارت باقاعده با طول n را می توان به یک NFA با O(n) وضعیت تبدیل کرد.
- در زمان $O(mn^2)$ میتوان تعیین کرد که آیا رشته ای با طول m با عبارت باقاعده با طول n منطبق می شود یا خیر.
- بعدا نشان میدهیم چگونه این زمان را به O(m) کاهش میدهیم.