Lecture 4: Finite Automata

st select an open discussion section to be enrolled. Please
iately. You will not get off the waitlist until you do.

uld now have a Unix instructional account. If you don't,

ve teams formed by Friday, if possible, but next Monday
.

ork due next Wednesday (31 Jan).

Public-Service Announcement

erimental Social Science Laboratory (Xlab) invites
cipate in social science studies! Experiments con-
lab (located in Hearst Gym, Suite 2) are computer-
n-making studies such as tasks, surveys, and games.
sionally offer remote online and mobile studies that
leted anywhere. Participants earn $15/hour on av-
time they participate. For more information, visit
y.edu. To sign up, visit berkeley.sona-systems.com.

Some Abbreviations

rm from the last slide is good for formal analysis, but
ng.

llow some abbreviations that are obviously exandable
c forms:

Abbreviation	Meaning
$\cdots \mid \mathcal{R}_n$	$A: \mathcal{R}_1$ $\vdots$ $A: \mathcal{R}_n$
$\mathcal{R})\cdots$	$B: \mathcal{R}$ $A: \cdots B \cdots$
$_1$" | $\cdots$ | "c_n" other character classes)	$[c_1 \cdots c_n]$

rnative Style for Describing Languages

giving a single pattern, we can give a set of rules of the

$$A: \ \alpha_1\alpha_2\cdots\alpha_n, \ \ n \geq 0,$$

mbol that is intended to stand for a language (set of
a *metavariable* or *nonterminal symbol*.
s either a literal character (like "a") or a nonterminal

tation of this rule is

o form a string in $L(A)$ (the language denoted by A) is
enate one string each from $L(\alpha_1),\ L(\alpha_2), \ldots$.

') is just the language $\{"c"\}$).

us-Naur Form (BNF). A set of rules is a *grammar*. One
rminals is designated as its *start symbol* denoting the
cribed by the grammar.

see that ':' written many different ways, such as ': :=',
e'll just use the same notation our tools use.

A Big Restriction (for now)

being, we'll also add a restriction. In each rule:

$$A: \ \alpha_1\alpha_2\cdots\alpha_n, \ \ n \geq 0,$$

that if α_i is a nonterminal symbol, then either

les for that symbol have to occured before all the rules

is the last item) and α_n is A.

a restricted grammar a *Type 3* or *regular* grammar. s definable by regular grammars are called *regular lan-*

gular languages are exactly the ones that can be de- egular expressions.

Some Technicalities

finition, each nonterminal in a grammar defines a lan- en, we are interested in just one of them (the *start* the others are auxiliary definitions.

on of what a rule means ("One way to form a string in leaves open the possibility that there are other ways to h $L(A)$ than covered in the rule.

t freedom in order to allow multiple rules for A, but we want to include strings that aren't covered by some rule.

athematical definitions throw in sentences like:

r defines the *minimal* languages that contain all strings fy the rules.

Proof of Claim (I)

regular expression, $\mathcal{R}$, and make a (possibly not yet

(preceding) rule for each parenthesized expression.

e just the constructs '$X*$', '$X+$', and '$X?$'. What do we ?

Proof of Claim (II)

uct. . .	. . . with Q, where
	`Q :` `Q : R Q`
	`Q : R` `Q : R Q`
	`Q :` `Q : R`

Side Note: The Empty Language

r for the empty language is a bit non-intuitive:

trings, Q, satisfies this rule.

e implicit rule that we choose the *smallest* solution that rules, Q represents the empty set.

Example

regular expression `("+"|"-")?("0"|"1")+`

`?("0"|"1")+` *replace with . . .*

```
: "+" | "-"
: "0" | "1"
 Q1? Q2+
```

replace with . . .

```
Q3: ϵ | Q1
Q4: Q2 | Q2 Q4
R: Q3 Q4
```


Review: FA operation

ph whose nodes are states (of memory) and whose edges nsitions. There are a finite number of nodes.

the designated start state.

of the nodes are final states.

ion is labeled with a set of symbols (characters, etc.) or

izes a string $c_1c_2\cdots c_n$ if there is a path (sequence of the start state to a final state such that the labels in sequence, aside from ϵ edges, respectively contain

leaving any node have disjoint sets of characters and no ϵ nodes, FA is a DFA, else an NFA.

cal Pattern-Matching Implementation

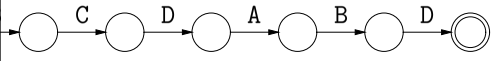
s, can generally make do with "classical" regular expres-

le using *finite(-state) automata* or *FAs*. ("Finite state" mory").

struction:

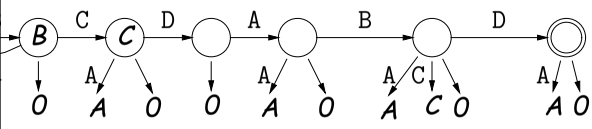
xpression ⇒ nondeterministic FA (NFA)
inistic FA (DFA) ⇒ table-driven program.

ple: What does this NFA recognize?



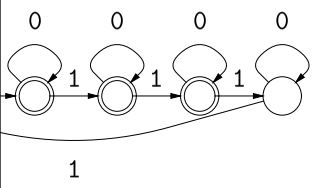
Strings of capitals ending in ABCDABD.

plest equivalent DFA you can think of?



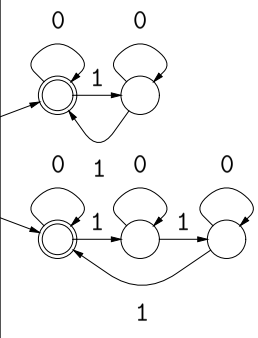
labels mean "any character not covered by another edge.")

ple: What does this DFA recognize?

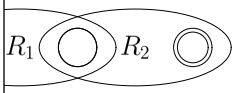


Bit strings with # of 1's divisible by 2 or 3.

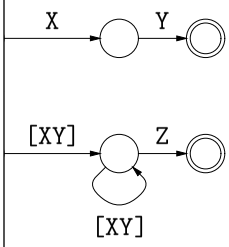
plest equivalent NFA you can think of?



lassical Regular Expressions to NFAs (I)



ple: What does this NFA recognize?



plest equivalent DFA you can think of?

Extensions?

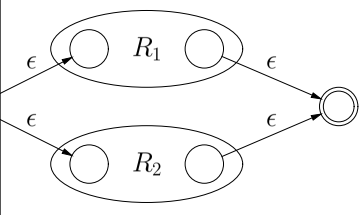
ou translate ϕ (the empty language, containing no strings)

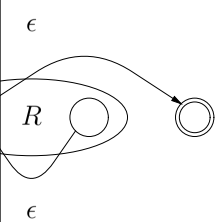
ou translate 'R?' into an NFA?

ou translate 'R+' into an NFA?

ou translate '$R_1|R_2|\cdots|R_n$' into an NFA?

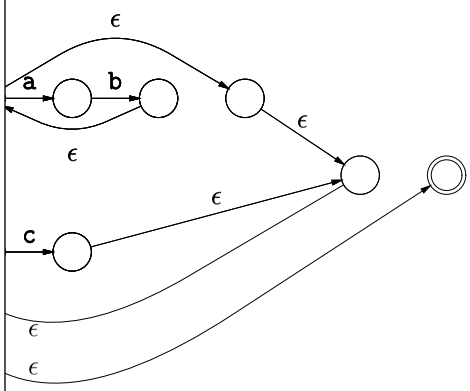
lassical Regular Expressions to NFAs (II)



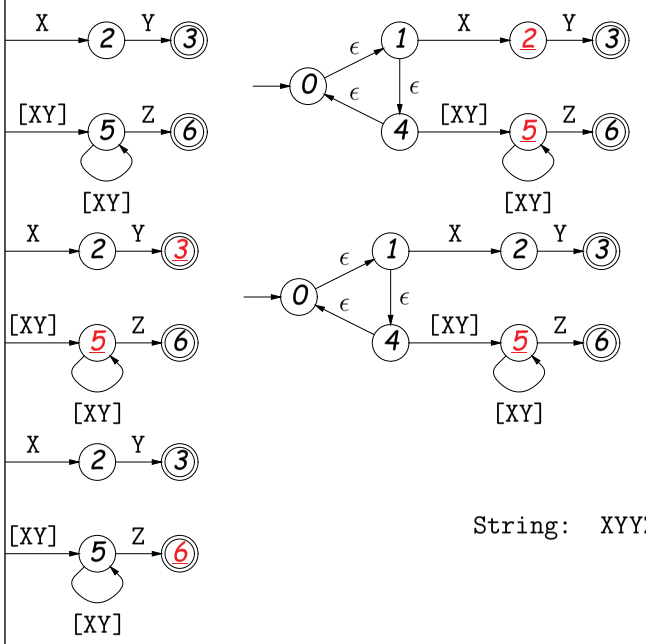


Example of Conversion

translate `((ab)*|c)*` into an NFA (using the construc-



bstract Implementation of NFAs



DFAs as Programs

)FA in program with control structure:

```
ITIAL;
nput; *s != '\0'; s += 1) {
(state):
ITIAL:
*s == 'a') state = A_STATE; break;
STATE:
*s == 'b') state = B_STATE; else state = INITIAL; break;


te == FINAL1 || state == FINAL2;
```

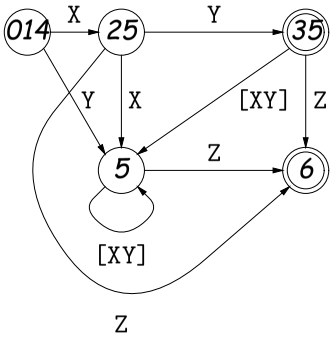
structure (table driven):

```
ITIAL;
nput; *s != '\0'; s += 1)
 transition[state][s];
inal[state];
```


Review: Converting to DFAs

ON: The set of states that are marked (colored red) each character in a way that depends only on the set acter.

ds, machine on previous slide acted like this DFA:



How Do They Do It?

use a DFA to recognize longest match?

use DFA to act on first of equal-length matches?

use a DFA to handle the R_1/R_2 pattern (matches just f followed by R_2, like R_1(?=R_2) in Python)?

What Flex Does

n specification is giant regular expression of the form , where none of the R_i match ϵ.

ate labeled with some action.

y previous methods, into a table-driven DFA.

rticular DFA is used to recognize *prefixes* of the (re- t: initial portions that put machine in a final state.

state(s) we end up in determine action. To deal with ons:

gest prefix ("maximum munch").

are multiple matches, apply *first* rule in order.