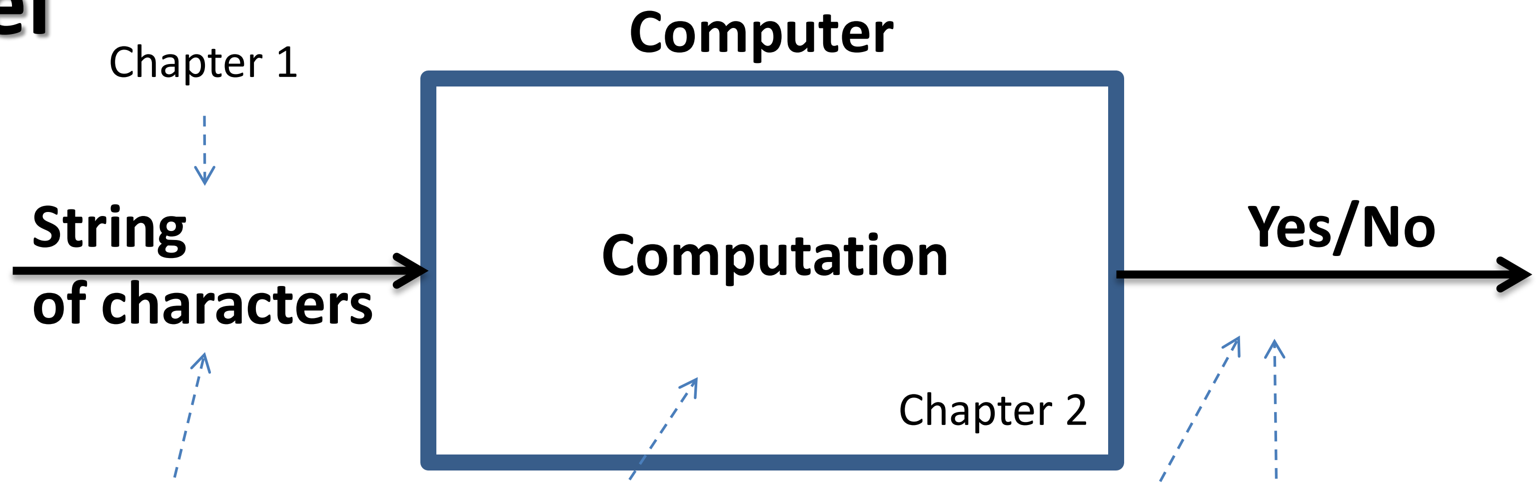
Chapter 2

Finite Automata and the Languages They Accept

Model

Chapter 1



This device plays a role of a language acceptor

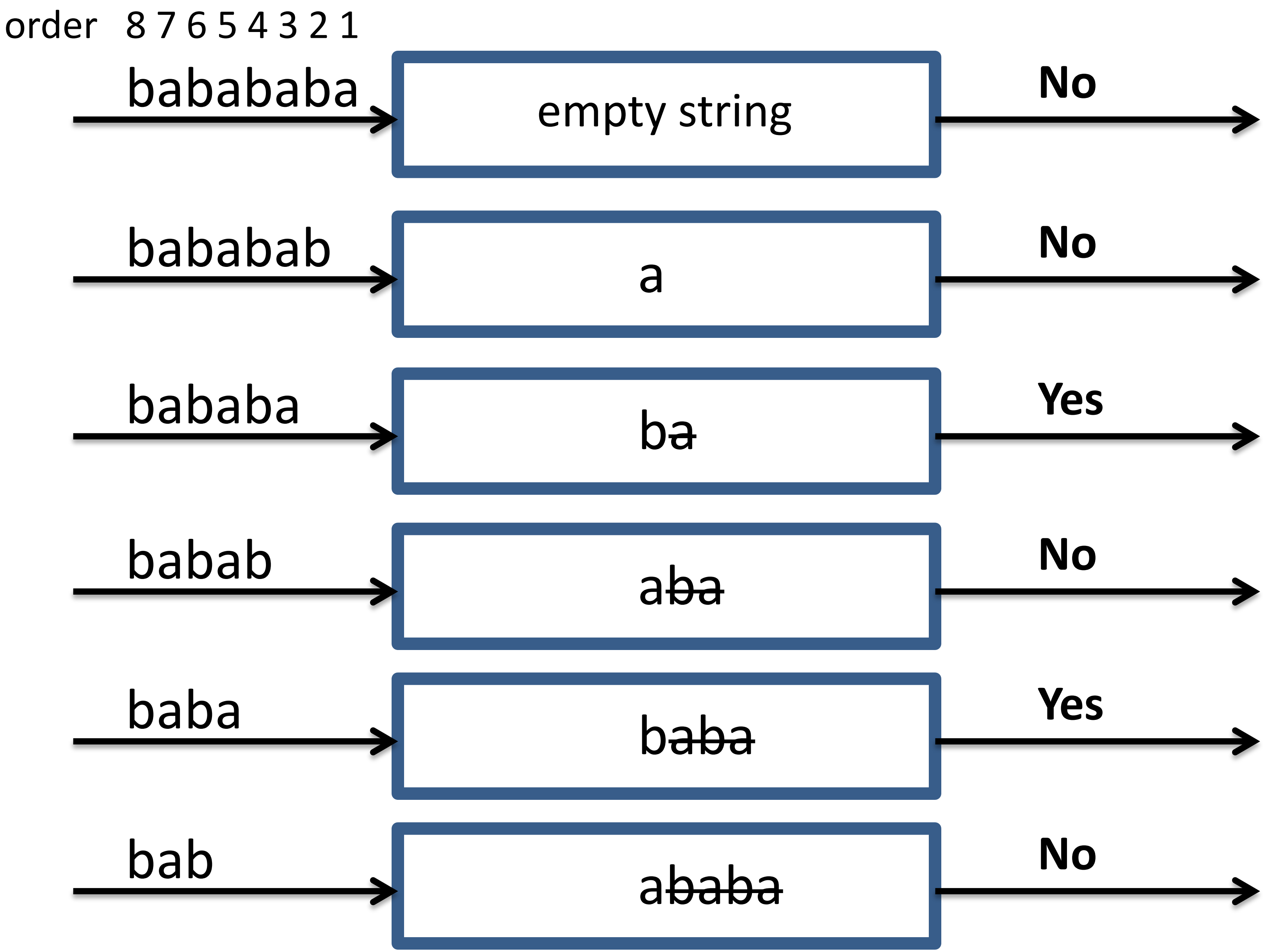
```
my $mystring;

$mystring = "Hello world!";

if($mystring =~ m/world/) { print "Yes"; } else { print “No”;}
```

Code in Perl

Intuition about finite automaton model requirements

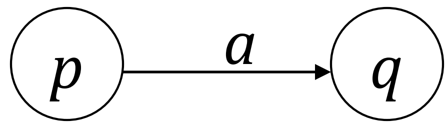
- A *finite automaton* is a simple type of computer
  - Its output is limited to “yes” or “no”
  - It has very primitive memory capabilities
- Our primitive computer that answers yes or no acts as a *language acceptor*
- For this model, consider that:
  - The input comes in the form of a string of individual input symbols
  - The computer gives an answer for the *current* string (the string of symbols that have been read so far)

Finite automaton that accepts language $L = \{(ab)^n \mid n \in N_{\geq 1}\}$.

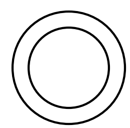
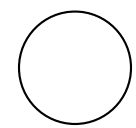


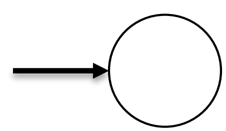
- A finite automaton (FA) or *finite state machine* is always in one of a finite number of *states*

- At each step FA makes a move (from state to state) that depends only on the **current state** and the **input symbol**



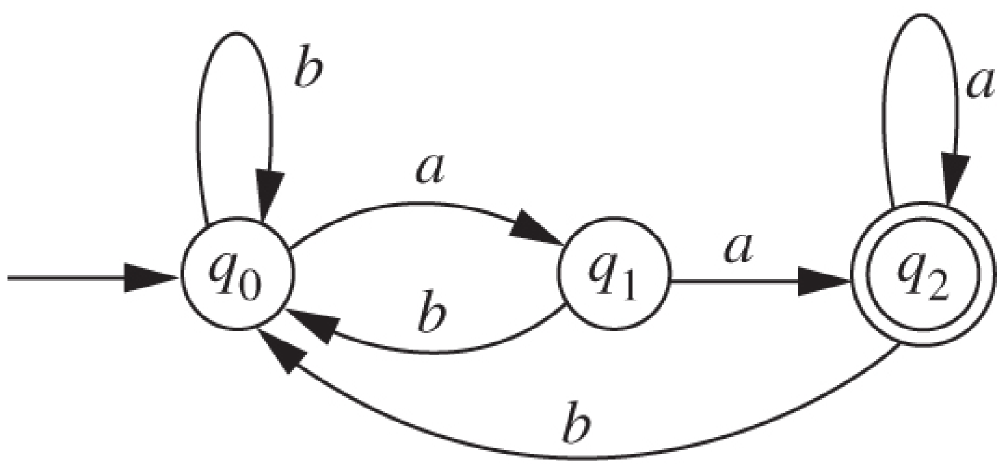
- The move is to enter a particular state (possibly the same as the one it was already in)

- States are either *accepting*  or *nonaccepting* 
  - Entering an accepting state means answering “yes”
  - Entering a nonaccepting state means “no”

- An FA has an initial state 

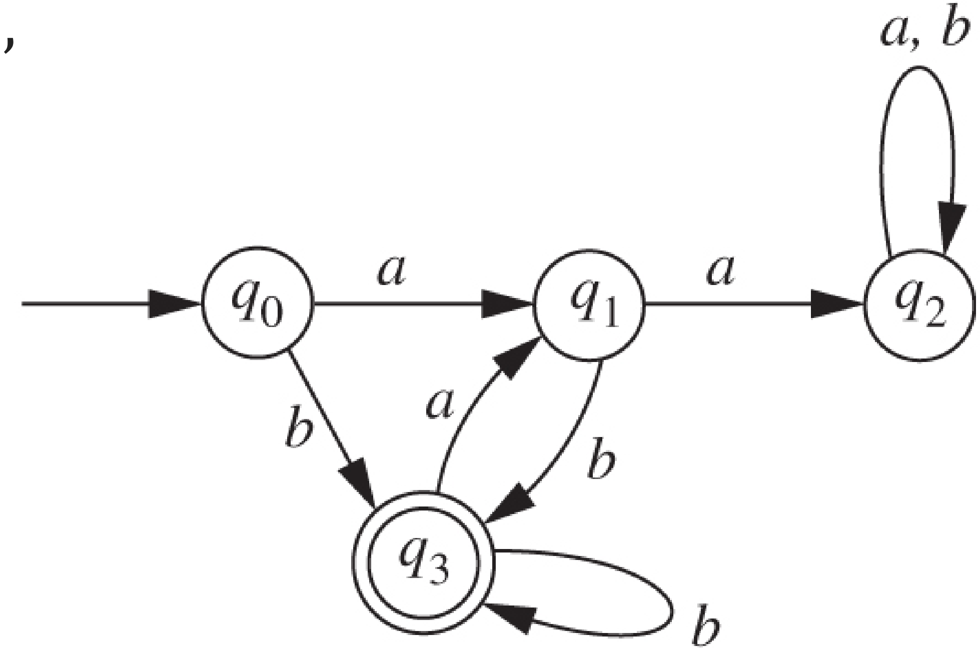
Finite Automata: Example

- This FA accepts the language of strings that end in *aa*
  - The three states represent strings that end with no *a*’s, one *a*, and two *a*’s, respectively
  - From each state, if the input is anything but an *a*, go back to the initial state, because now the current string doesn’t end with *a*



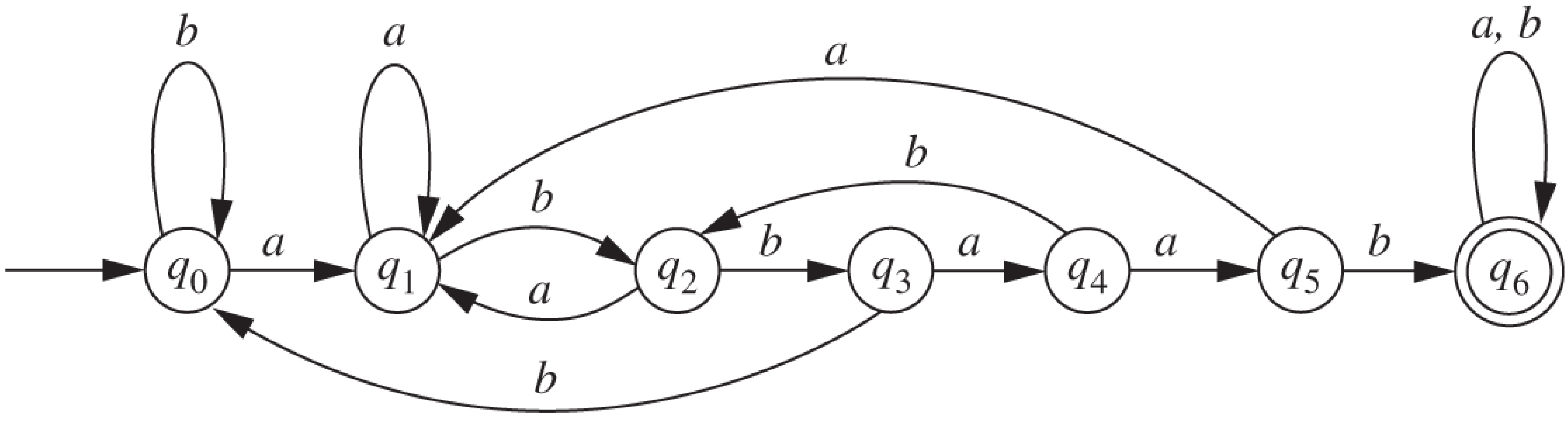
Finite Automata: Example

- This FA accepts the strings ending with *b* and not containing *aa*
  - The idea is to go to a permanently-non-accepting state if you ever read two *a*'s in a row
  - Go to an accepting state if you see a *b* (and haven't read two *a*'s),



Finite Automata: Example

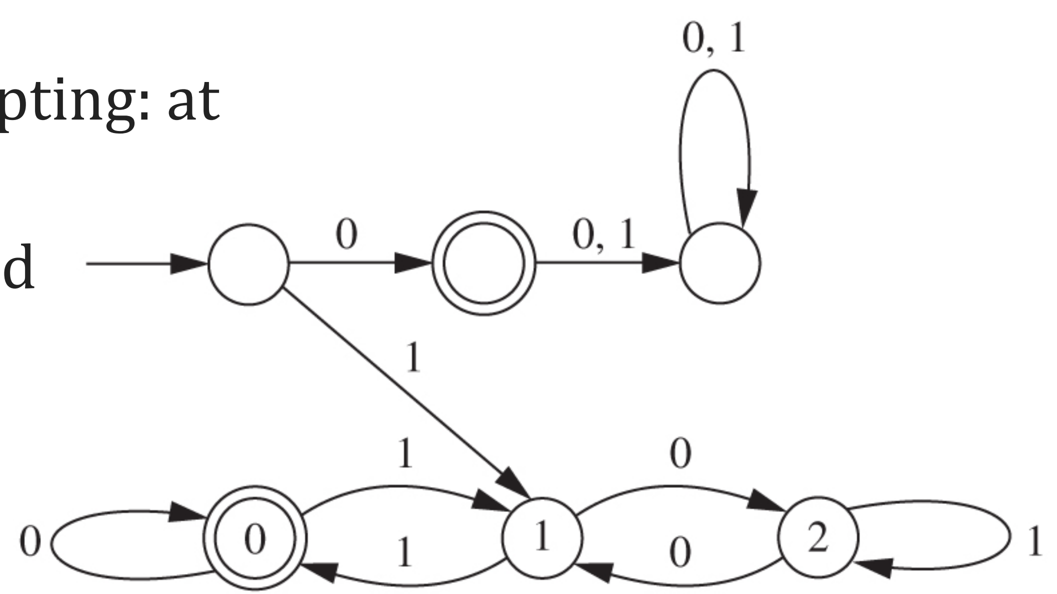
- This FA accepts strings that contain *abbaab*
- What do we do when a prefix of *abbaab* has been read but the next symbol doesn't match?
  - Go back to the state representing the longest prefix of *abbaab* at the end of the new current string
  - Example: If we've read *abba* and the next symbol is *b,* go to q_2, because *ab* is the longest prefix at the end of *abbab*



Finite Automata: the language of strings that are the binary representations of natural numbers divisible by 3.

If *x* represents *n*, and *n mod 3* is *r*, then what are *2n mod 3* and *(2n + 1) mod 3*? It is almost correct that the answers are *2r* and *2r + 1*; the only problem is that these numbers may be *3* or bigger, and in that case we must do another *mod 3* operation.

- States 0, 1, and 2 represent the current "remainder"
- The initial state is non-accepting: at least one bit is required
- Leading zeros are prohibited
- Transitions represent multiplication by two, then addition of the input bit



n	bin	r	n	bin	r
0	0	0	16	10000	1
1	1	1	17	10001	2
2	10	2	18	10010	0
3	11	0	19	10011	1
4	100	1	20	10100	2
5	101	2	21	10101	0
6	110	0	22	10110	1
7	111	1	23	10111	2
8	1000	2	24	11000	0
9	1001	0	25	11001	1
10	1010	1	26	11010	2
11	1011	2	27	11011	0
12	1100	0	28	11100	1
13	1101	1	29	11101	2
14	1110	2	30	11110	0
15	1111	0	31	11111	1