LING/C SC/PSYC 438/538

Lecture 20

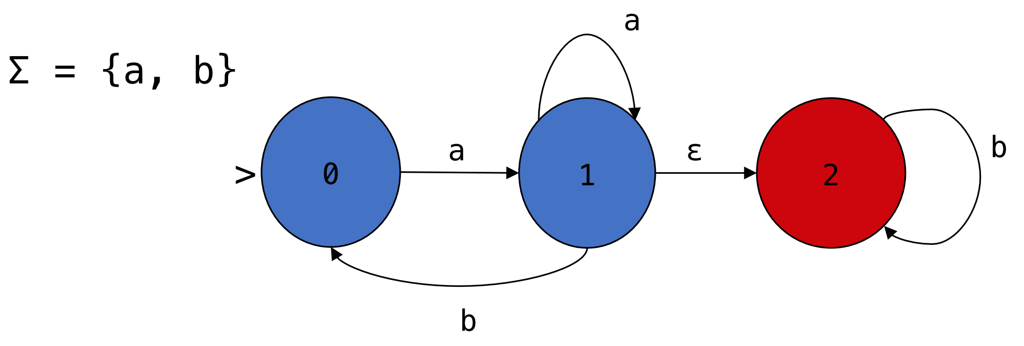
Sandiway Fong

Today's Topics

- Homework 10 review
- An example where a machine is (perhaps) easier to build than a regex.
- The state bypass method: converting a FSA into a regex algorithmically

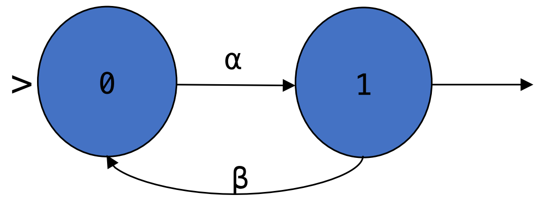
Homework 10

1. Give an equivalent Perl regex for the FSA shown below.
2. Convert the NDFSA to a (deterministic) FSA. Draw the machine.
3. Give the implementation of the FSA in Perl.
4. Run your two Perl programs and give examples:
   - your Perl regex should accept and reject (*) same strings as the Perl FSA
   - a, *b, aa, ab, *ba, aaab, abaabb, *abba, *abaabbaaabbb

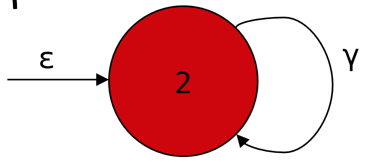


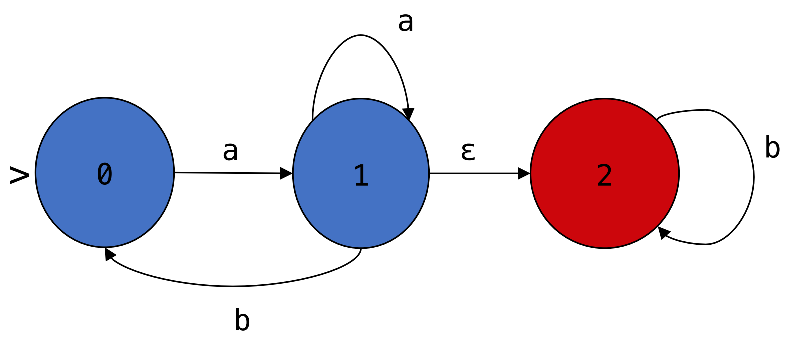
Homework 10 Review

- Regex, part by part
  - Part 1: α or αβα or αβαβα etc.
  - α(βα)*



- Part 2: ε or γ or γγ etc.
- γ*



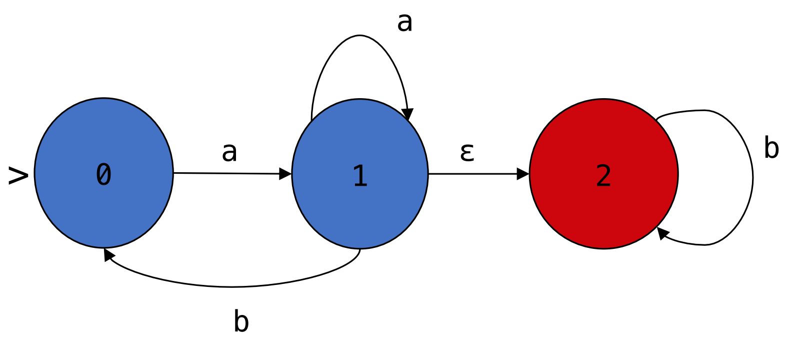


- Part 1 + Part 2 is:
  - α(βα)*γ*
- Apply to our FSA:
  - α = aa* = a+
  - β = b
  - γ = b
  - a+(ba+)*b*

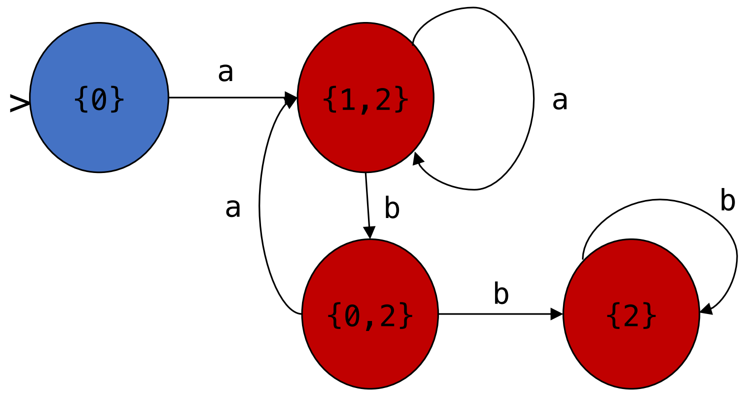
Homework 10 Review

Step by step over Σ

- State 0:
  - see a: {1, 2}
  - see b: {}
- State 1:
  - see a: {1, 2}
  - see b: {0}
- State 2:
  - see a: {}
  - see b: {2}

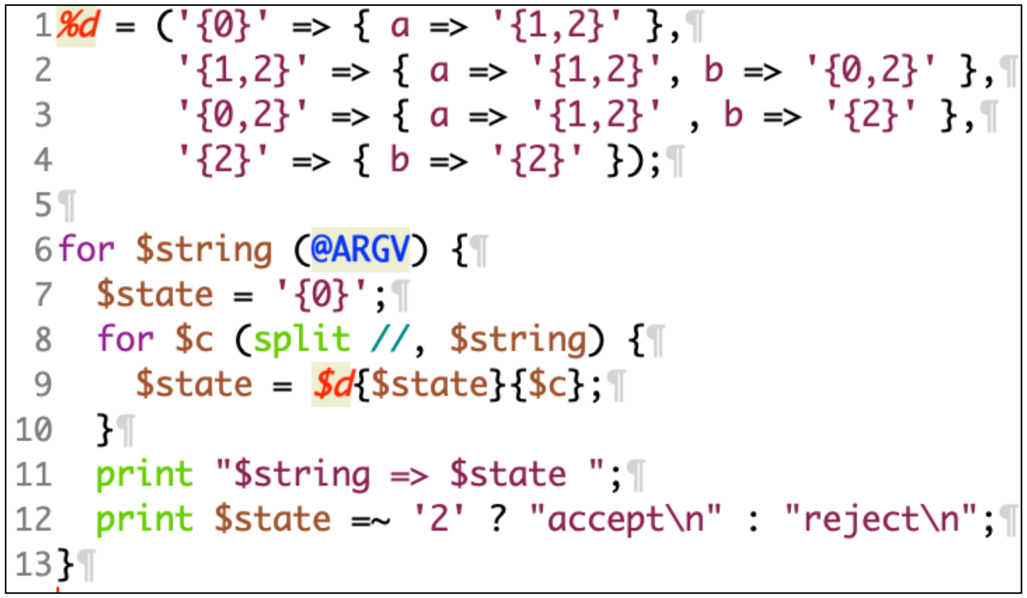


- Deterministic machine:

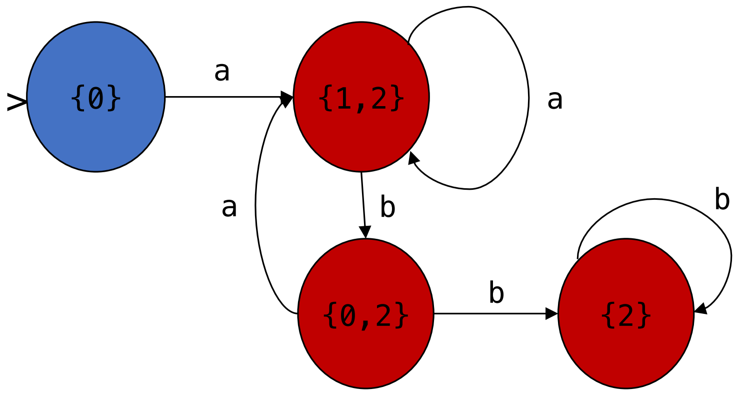


Homework 10 Review

- FSA code in Perl:



```perl
%d = ('{0}' => { a => '{1,2}' },
      '{1,2}' => { a => '{1,2}', b => '{0,2}' },
      '{0,2}' => { a => '{1,2}' , b => '{2}' },
      '{2}' => { b => '{2}' });

for $string (@ARGV) {
  $state = '{0}';
  for $c (split //, $string) {
    $state = $d{$state}{$c};
  }
  print "$string => $state ";
  print $state =~ '2' ? "accept\n" : "reject\n";
}
```



$$\begin{tikzcd}
\arrow[r] & \{0\} \arrow[r, "a"] & \{1,2\} \arrow[loop right, "a"] \arrow[d, "b"] & \\
& & \{0,2\} \arrow[u, bend left, "a"] \arrow[r, "b"] & \{2\} \arrow[loop right, "b"]
\end{tikzcd}$$

Homework 10 Review

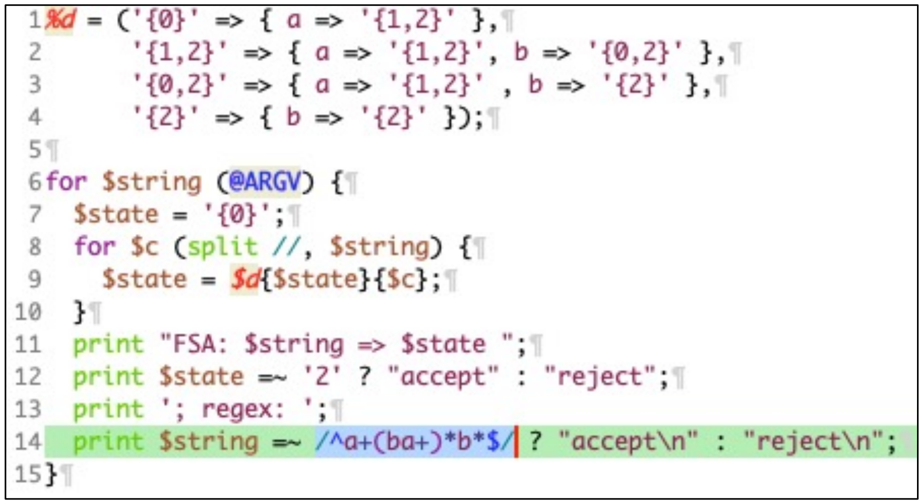
- your Perl regex should accept and reject (*) same strings as the Perl FSA
- a, *b, aa, ab, *ba, aaab, abaabb, *abba, *abaabbaaabbb

- Example:

```
perl hw10.perl a b aa ab ba aaab abaabb abba abaabbaaabbb
a => {1,2} accept
b =>  reject
aa => {1,2} accept
ab => {0,2} accept
ba =>  reject
aaab => {0,2} accept
abaabb => {2} accept
abba =>  reject
abaabbaaabbb =>  reject
```

```
perl hw10.perl ''
 => {0} reject
```

Homework 10 Review

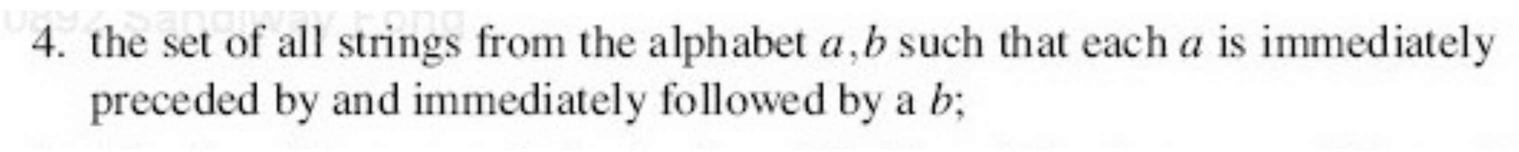


```perl
%d = ('{0}' => { a => '{1,2}' },
      '{1,2}' => { a => '{1,2}', b => '{0,2}' },
      '{0,2}' => { a => '{1,2}' , b => '{2}' },
      '{2}' => { b => '{2}' });

for $string (@ARGV) {
  $state = '{0}';
  for $c (split //, $string) {
    $state = $d{$state}{$c};
  }
  print "FSA: $string => $state ";
  print $state =~ '2' ? "accept" : "reject";
  print '; regex: ';
  print $string =~ /^a+(ba+)*b*$/ ? "accept\n" : "reject\n";
}
```

```
perl hw10.perl '' a b aa ab ba aaab abaabb abba
abaabbaaabbb

FSA:  => {0} reject; regex: reject

FSA: a => {1,2} accept; regex: accept

FSA: b =>  reject; regex: reject

FSA: aa => {1,2} accept; regex: accept

FSA: ab => {0,2} accept; regex: accept

FSA: ba =>  reject; regex: reject

FSA: aaab => {0,2} accept; regex: accept

FSA: abaabb => {2} accept; regex: reject

FSA: abba =>  reject; regex: reject

FSA: abaabbaaabbb =>  reject; regex: reject
```

Regex from FSA

Textbook Exercise: find a regex for


4. the set of all strings from the alphabet a, b such that each a is immediately preceded by and immediately followed by a b;

Examples (* denotes string not in the language):

*ab *ba

b**a**b

λ (empty string)

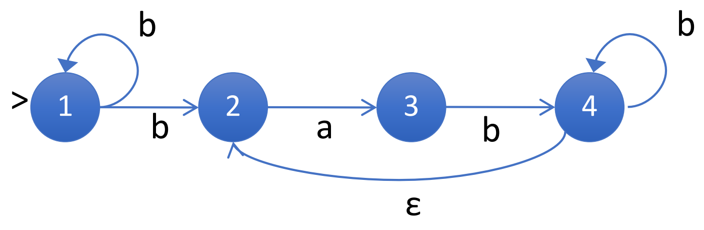
bb

*baba

b**a**b**a**b

Regex from FSA

- Draw a FSA and convert it to a regex:

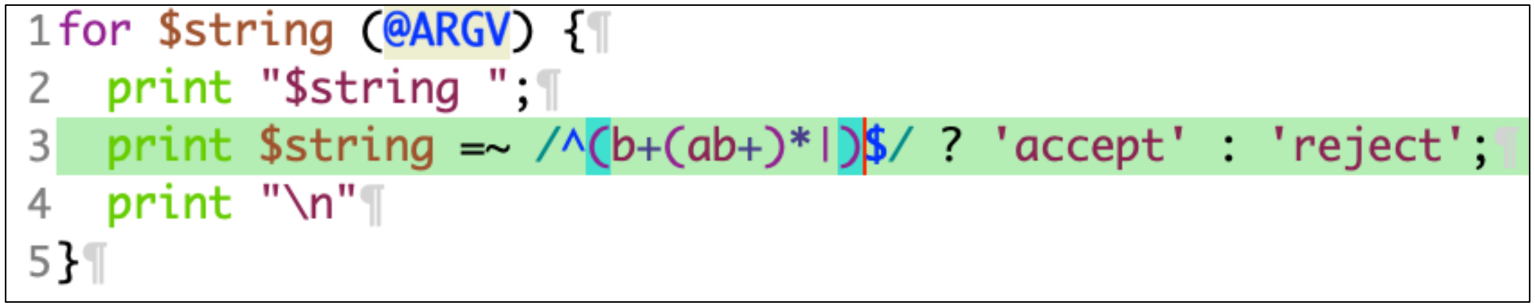


[Powerpoint Animation]

b* b (ab+)+

= b+(ab+)* | ε

Regex from FSA

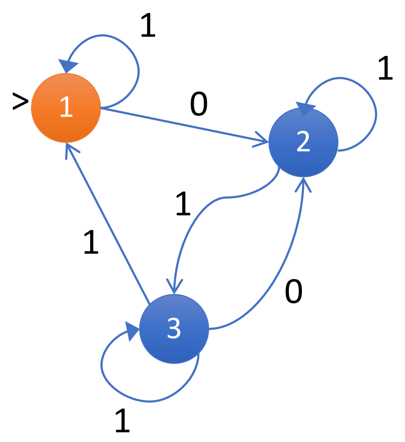


```perl
for $string (@ARGV) {
  print "$string ";
  print $string =~ /^(b+(ab+)*|)$/ ? 'accept' : 'reject';
  print "\n"
}
```

```
perl regex4.perl '' ab ba bab bb baba babab
 accept
ab reject
ba reject
bab accept
bb accept
baba reject
babab accept
```

Regex from FSA

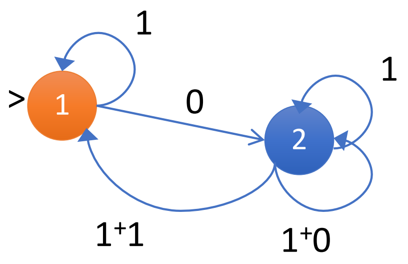
- Example:
  - Give a regex for the NDFSA:

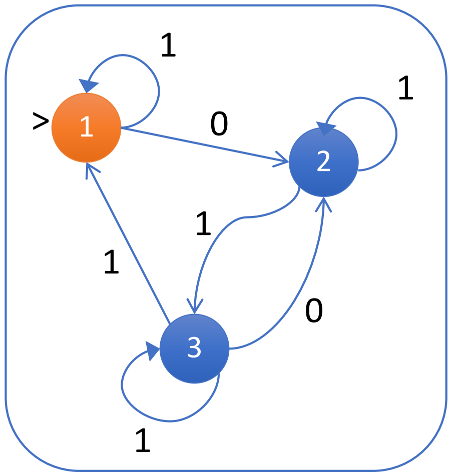


- State by-pass method:
  1. Delete one state at a time
  2. Calculate the possible paths passing through the deleted state
  3. Add the regex calculated at each stage as an arc
  - e.g.
    - eliminate state 3
    - then 2…

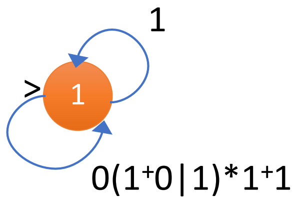
Regex from FSA

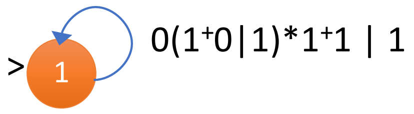
- eliminate state 3





- eliminate state 2





Answer: $(0(1^+0|1)^*1^+1 \mid 1)^*$

[Powerpoint animation]

Another way: Regex from FSA

The example from two slides ago …

- BUT:
  - let's do it in a different order, so:
  - step 1: eliminate state 2
  - step 2: eliminate state 3