LING/C SC/PSYC 438/538

Lecture 18

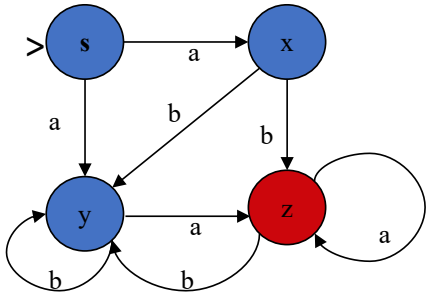
Sandiway Fong

Today's Topics

- FSA *contd*.
  - non-determinism
  - NDFSA to DFSA conversion
- Homework 10

Non-Deterministic Finite State Automata (NDFSA)

- **non-deterministic FSA (NDFSA)**
- *no restriction on ambiguity (surprisingly, no increase in power)*
- Example:



Non-Deterministic Finite State Automata (NDFSA)

```
function ND-RECOGNIZE(tape, machine) returns accept or reject

  agenda ← {(Initial state of machine, beginning of tape)}
  current-search-state ← NEXT(agenda)
  loop
    if ACCEPT-STATE?(current-search-state) returns true then
      return accept
    else
      agenda ← agenda ∪ GENERATE-NEW-STATES(current-search-state)
    if agenda is empty then
      return reject
    else
      current-search-state ← NEXT(agenda)
  end
function GENERATE-NEW-STATES(current-state) returns a set of search-states

  current-node ← the node the current search-state is in
  index ← the point on the tape the current search-state is looking at
  return a list of search states from transition table as follows:
    (transition-table[current-node,ϵ], index)
    ∪
    (transition-table[current-node, tape[index]], index + 1)
function ACCEPT-STATE?(search-state) returns true or false

  current-node ← the node search-state is in
  index ← the point on the tape search-state is looking at
  if index is at the end of the tape and current-node is an accept state of machine
  then
    return true
  else
    return false
```

Figure 2.19 An algorithm for NFSA recognition. The word *node* means a state of the FSA, and *state* or *search-state* means "the state of the search process", i.e., a combination of *node* and *tape position*.

Possible strategies for keeping track of multiple states:

1. Backtracking (***backup***)
2. Parallelism (*split the computation*)

algorithm gets complicated fast

Finite State Automata (FSA)

```
function D-RECOGNIZE(tape, machine) returns accept or reject

  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
   if End of input has been reached then
    if current-state is an accept state then
     return accept
    else
     return reject
   elsif transition-table[current-state,tape[index]] is empty then
    return reject
   else
    current-state ← transition-table[current-state,tape[index]]
    index ← index + 1
  end
```

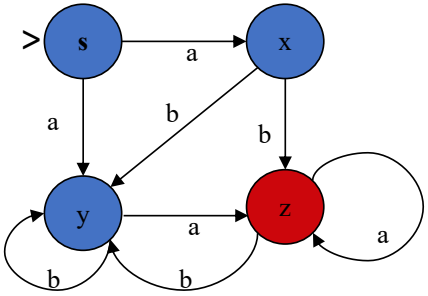
Figure 2.12 An algorithm for deterministic recognition of FSAs. This algorithm returns *accept* if the entire string it is pointing at is in the language defined by the FSA, and *reject* if the string is not in the language.

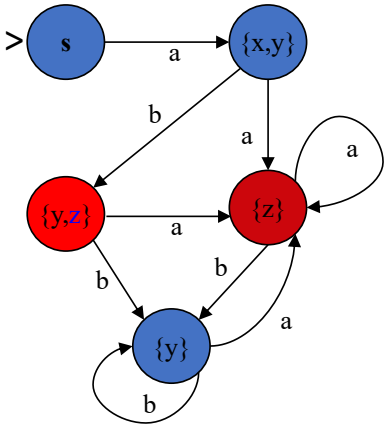
from Lecture 17

NDFSA → (D)FSA

[discussed at the end of section 2.2 in the textbook]

- construct a new machine
  - each state of the new machine represents the ***set of possible states*** of the original machine when stepping through the input
- **Note**:
  - new machine is equivalent to old one (*but has more states*)
  - new machine is deterministic
- **example**

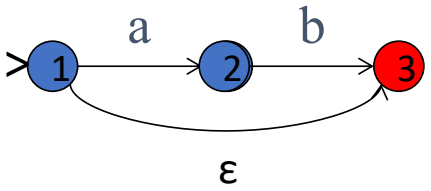


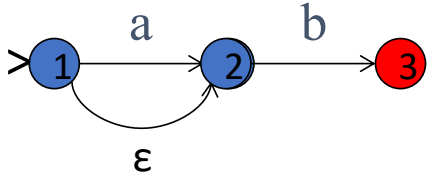


[Powerpoint Animation]

Workde EXERCISE

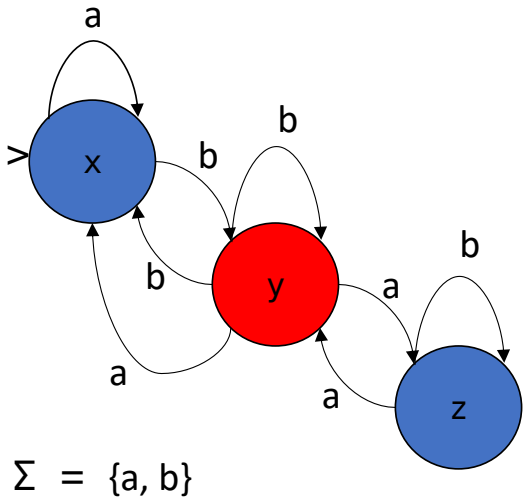
- Let's check our understanding:
  - apply the set-of-states construction technique to the two machines on the ε−transition slide from the previous lecture (*repeated below*)
- How to check your answer?
  - should confirm the machine produced is actually **deterministic** and accurately simulates its ε−transition counterpart





Homework 10

- Consider the following NDFSA:



- Q1: Why is it a NDFSA?
- Q2: What is the shortest nonempty string it does not accept?
- Q3: Which strings of length 4 does it accept? How many are there?
- Q4: Convert our NDFSA into a DFSA. How many states does the DFSA have? How many final states?
  - *use the construction shown in class*

Homework 10

- Extra Credit:
  - implement your DFSA in Perl or Python (only)
  - Which of the following strings does it accept?

1. aaaabbbaaa
2. aaaabbbaab
3. aaaabbbbba
4. aaaabbbaba

Homework 10

- Usual rules …
  - One PDF file
  - Subject: 438/538 Homework 10 *YOUR NAME*
  - Due date: Sunday midnight
- You can draw your machine for Q4 by hand (*make it legible*)
  - put in the set of states {…}
- For the EC question: attach your Perl/Python code.