Lecture 3: Finite Automata

Administrivia

- Log into your class account ASAP (I still have account forms).
- Start forming teams:
  - Choose team name (letters, digits, underscores only, starting with capital letter)
  - Email me (`Hilfinger@cs.berkeley.edu`) name of team, and class logins of members (also mail changes).
- Good time to start learning Python (manuals online).
- Readings for next time: Subversion manual, Course Notes §2.1–2.7.
- Don't forget homework due Friday.

Classical Pattern-Matching Implementation

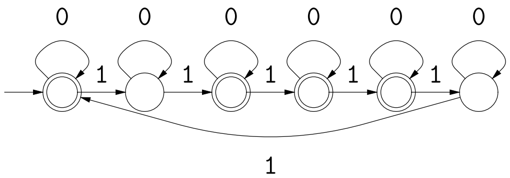
- For compilers, can generally make do with "classical" regular expressions.
- Implementable using *finite(-state) automata* or *FAs*. ("Finite state" = "finite memory").
- Classical construction:

regular expression ⇒ nondeterministic FA (NFA)
⇒ deterministic FA (DFA) ⇒ table-driven program.

Review: FA operation

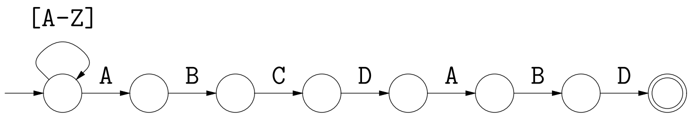
- A FA is a graph whose nodes are states (of memory) and whose edges are state transitions. There are a finite number of nodes.
- One state is the designated start state.
- Some subset of the nodes are final states.
- Each transition is labeled with a set of symbols (characters, etc.) or ϵ.
- A FA recognizes a string $c_1 c_2 \cdots c_n$ if there is a path (sequence of edges) from the start state to a final state such that the labels of the edges in sequence, aside from ϵ edges, respectively contain $c_1, c_2, \ldots, c_n$.
- If the edges leaving any node have disjoint sets of characters and if there are no ϵ nodes, FA is a DFA, else an NFA.

Example: What does this DFA recognize?



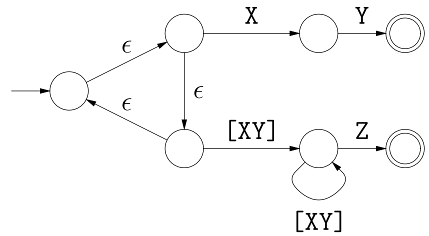
What is the simplest equivalent NFA you can think of?

Example: What does this NFA recognize?



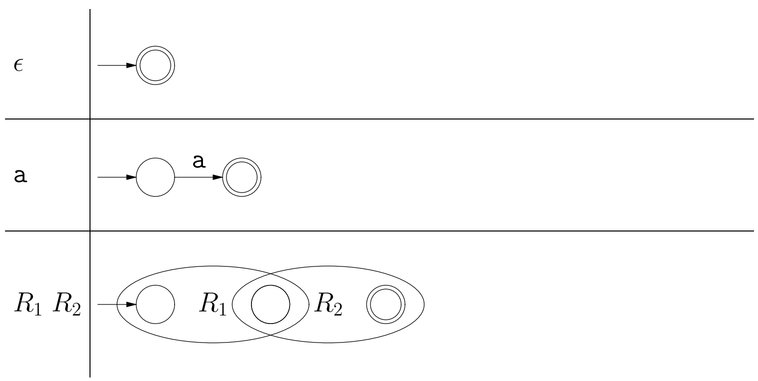
What is the simplest equivalent DFA you can think of?

Example: What does this NFA recognize?

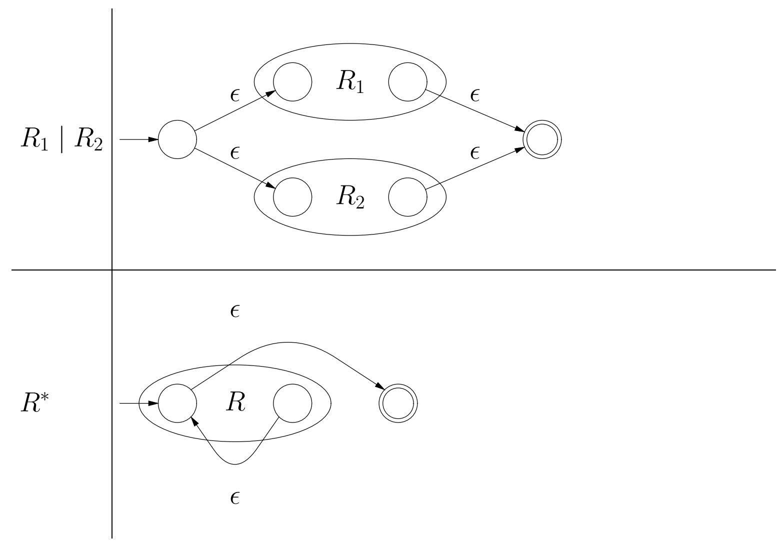


What is the simplest equivalent DFA you can think of?

Review: Classical Regular Expressions to NFAs (I)



Review: Classical Regular Expressions to NFAs (II)



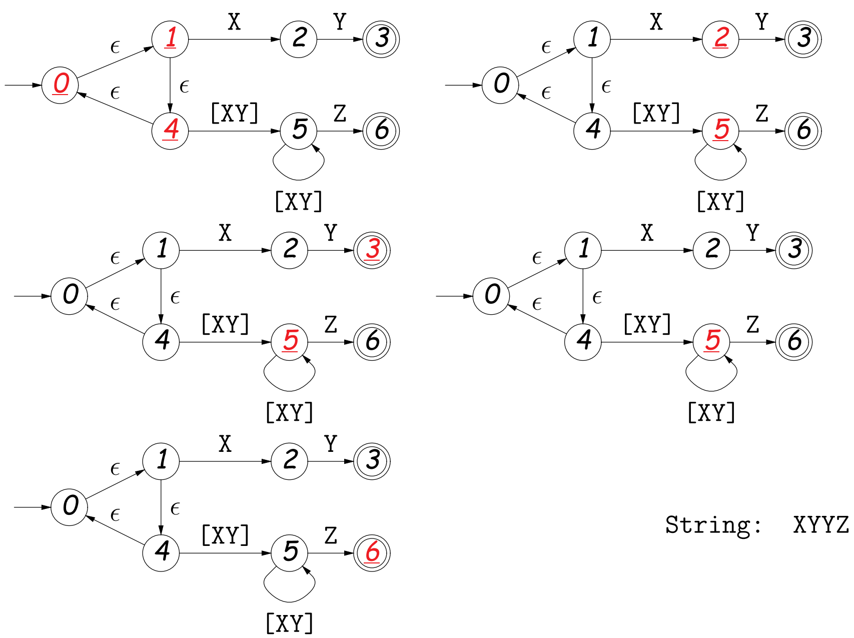
Extensions?

- How would you translate ϕ (the empty language, containing no strings) into an FA?
- How could you translate 'R?' into an NFA?
- How could you translate 'R+' into an NFA?
- How could you translate '$R_1|R_2|\cdots|R_n$' into an NFA?

Example of Conversion

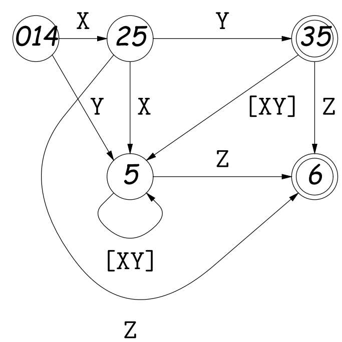
How would you translate `((ab)*|c)*` into an NFA?

Abstract Implementation of NFAs



Review: Converting to DFAs

- **OBSERVATION:** The set of states that are marked (colored red) changes with each character in a way that depends only on the set and the character.
- In other words, machine on previous slide acted like this DFA:



DFAs as Programs

- Can realize DFA in program with control structure:

```
state = INITIAL;
for (s = input; *s != '\0'; s += 1) {
   switch (state):
   case INITIAL:
      if (*s == 'a') state = A_STATE; break;
   case A_STATE:
      if (*s == 'b') state = B_STATE; else state = INITIAL; break;
   ...
   }
}
return state == FINAL1 || state == FINAL2;
```

- Or with data structure (table driven):

```
state = INITIAL;
for (s = input; *s != '\0'; s += 1)
   state = transition[state][s];
return isfinal[state];
```


What Flex Does

- Flex program specification is giant regular expression of the form $R_1|R_2|\cdots|R_n$, where none of the R_i match ϵ.
- Each final state labeled with some action.
- Converted, by previous methods, into a table-driven DFA.
- But, this particular DFA is used to recognize *prefixes* of the (remaining) input: initial portions that put machine in a final state.
- Which final state(s) we end up in determine action. To deal with multiple actions:
  - Match *longest* prefix ("maximum munch").
  - If there are multiple matches, apply *first* rule in order.

How Do They Do It?

- How can we use a DFA to recognize longest match?
- How can we use DFA to act on first of equal-length matches?
- How can we use a DFA to handle the R_1/R_2 pattern (matches just R_1 but only if followed by R_2, like R_1(?=R_2) in Python)?