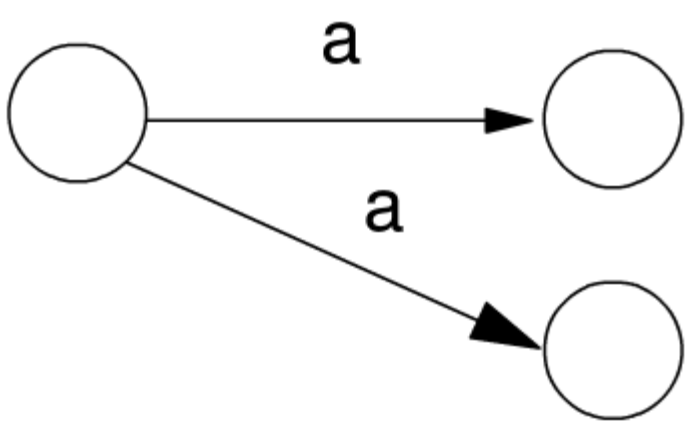
CSE431 – Translation of Computer Languages

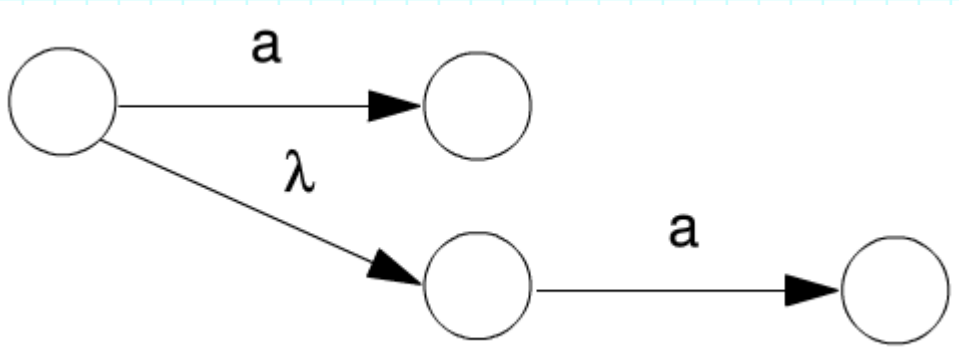
- Non-Deterministic Automata
- Transformations and Optimizations

Doug Shook

Non-deterministic Finite Automata

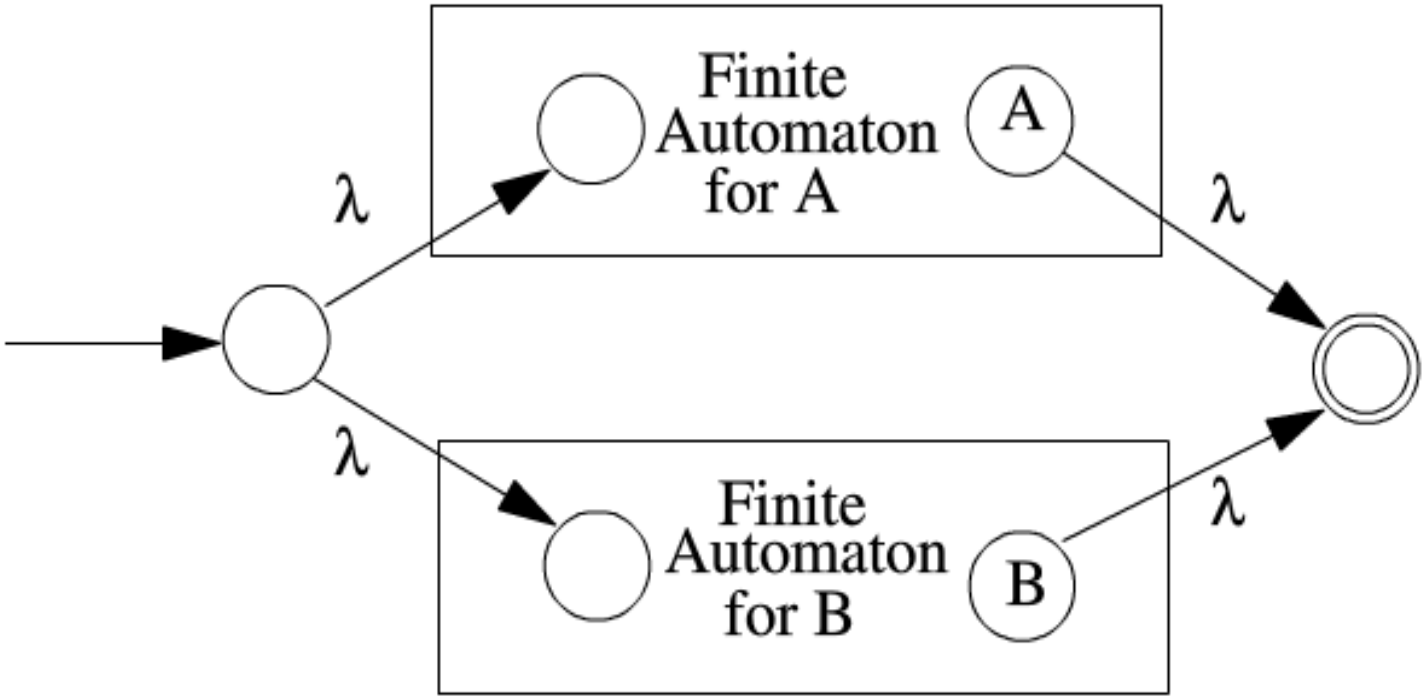
- DFA:
  - Only one “move” per input per state
- NFA:
  - Potentially many “moves” per input per state
  - Potentially move on lambda





Non-deterministic Finite Automata

- Regular expressions can be transformed into NFAs:
- Alternation looks like this:



- What about concatenation? Kleene*?

Exercises

- Create an NFA from the following regex:

  (ab)*

  a(a*b*)

  a(a|b)c*

  (a | (ba*))*a

NFA

- Wait a second…
  - Wasn't the claim that regex are the same as DFA?
  - And aren't these NFAs more complicated that DFAs?

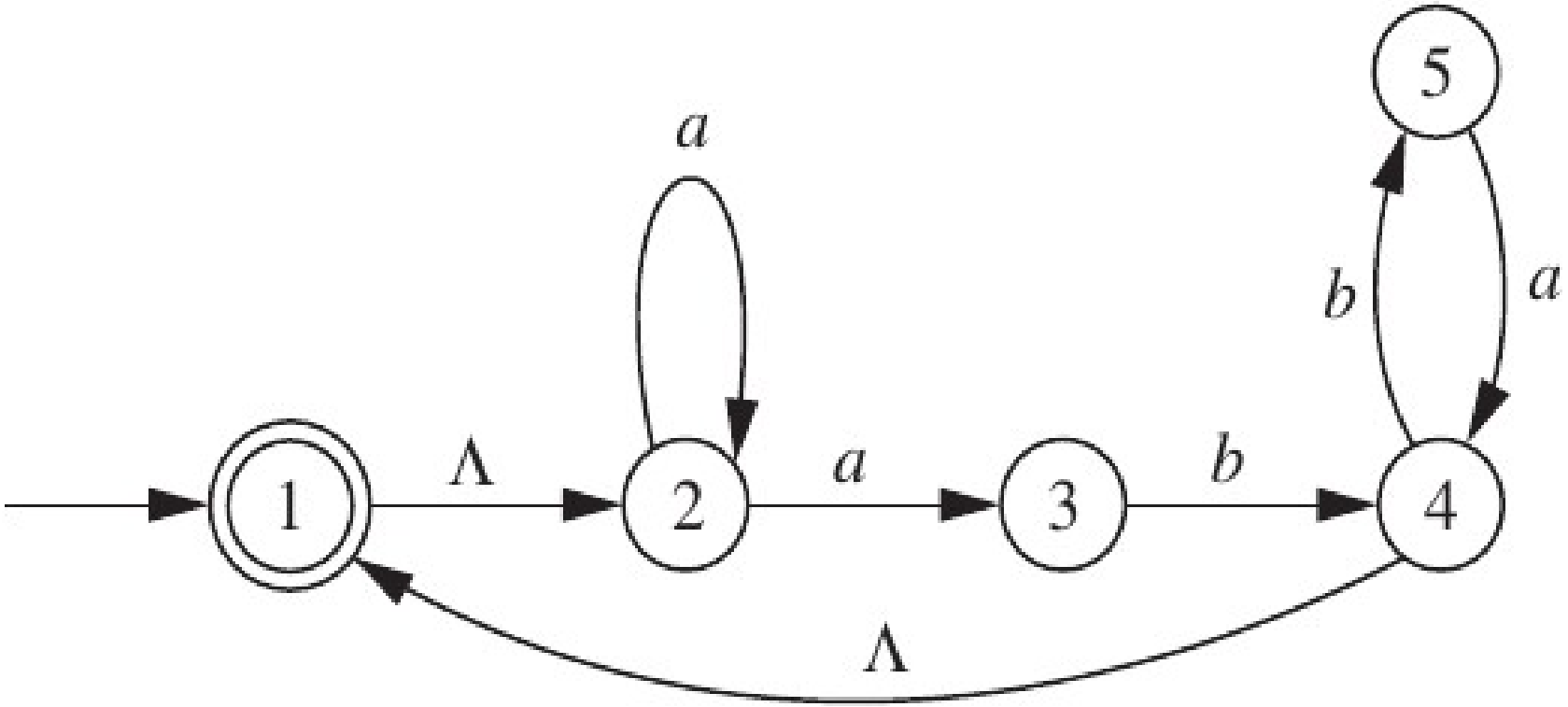
- Using *subset construction* we can convert an NFA to a DFA
  - NFAs and DFAs (and therefore regex) are equivalent!

Subset construction

- New states are created as a subset of NFA states
  - If I am at state A with an input of a, where could I end up?
  - Coast – gobble - coast
- Check transitions for all states, including new "subset" states
  - How do we know if a new state should be accepting?
  - How do we know when to stop?

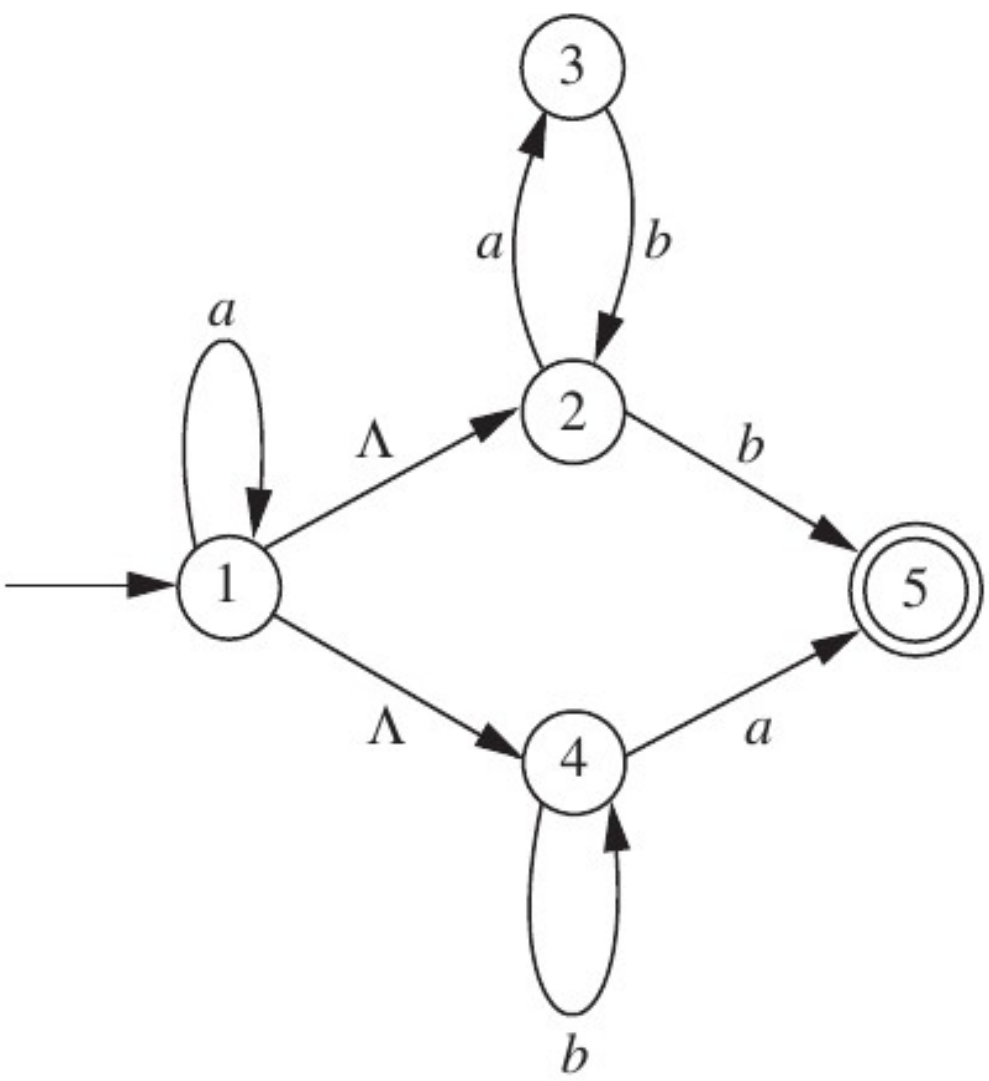
Subset construction

- Let's convert the following NFA to a DFA



Subset Construction

- Now try this one:



Minimization of DFAs

- When going from NFA -> DFA, the result can be quite large
  - If an NFA has n states, how many possible states might the corresponding DFA have?
- There is a method to minimize DFAs
  - Eliminate dead or unreachable nodes
  - Merge equivalent nodes

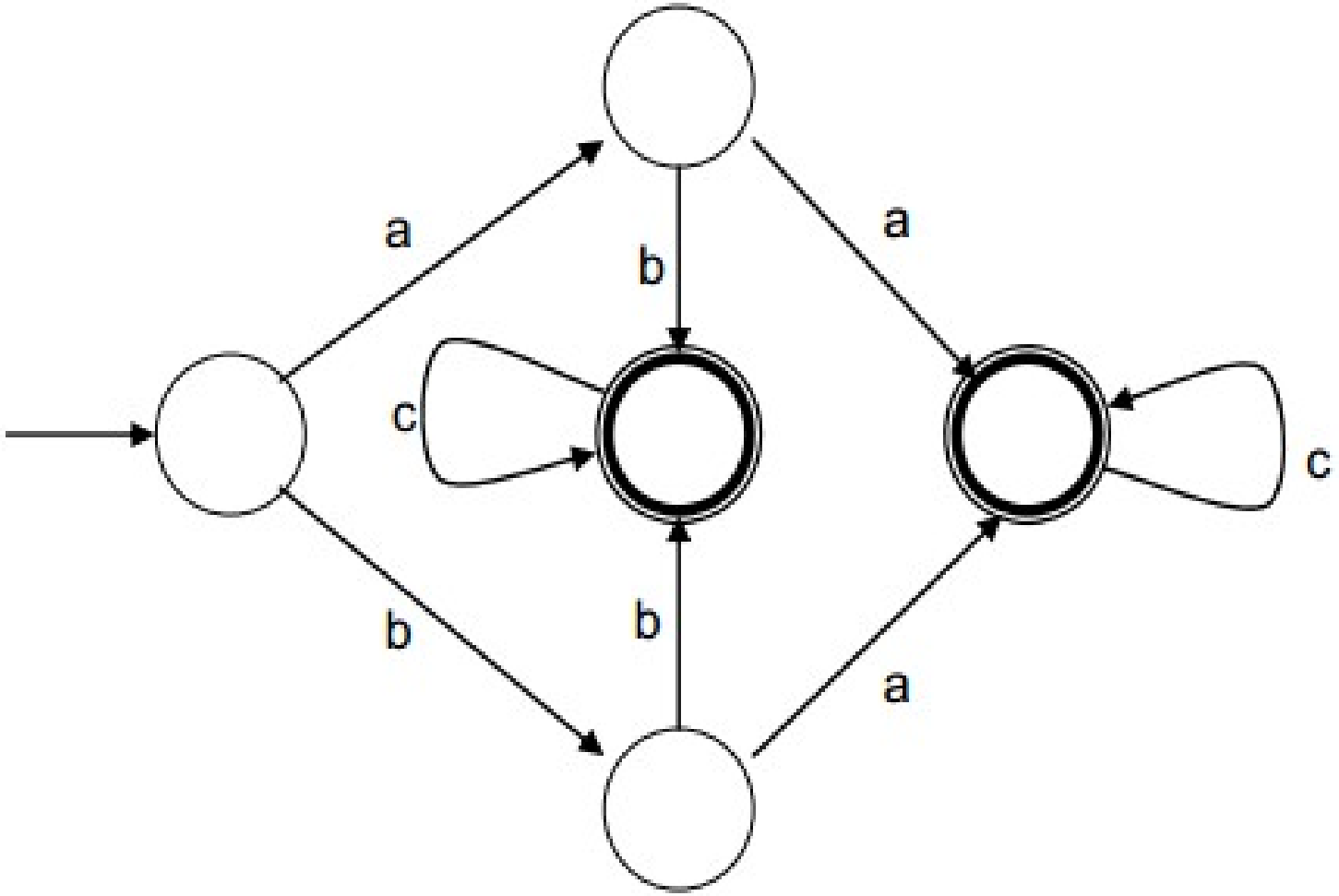
Minimization of DFAs

- The algorithm is a greedy one:
  - First merge all accepting states into one state
  - Merge all non-accepting state into one state

- For each “merged” state:
  - If there is an input that would cause the merged state to lead to two distinct states, split the merged state
    - Repeat until no more splits can be made

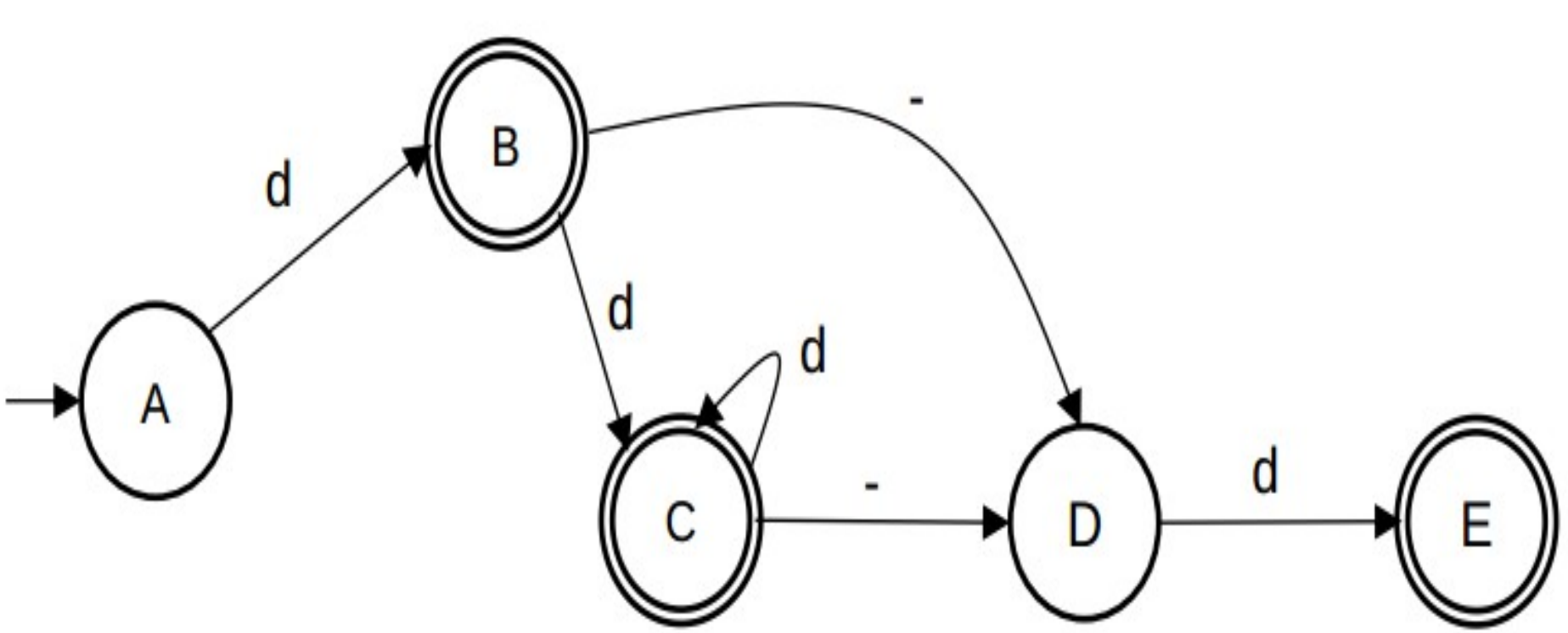
Minimization Example

- Let's minimize the following DFA:



Minimization Example

- Now try this one:



DFA to regular expression

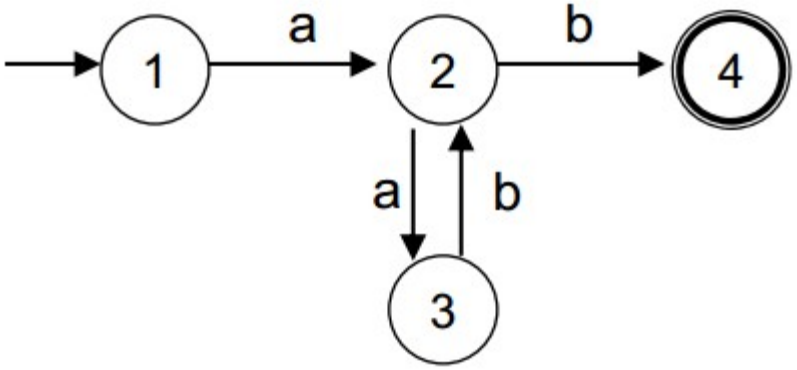
- It is also possble to go from a DFA (not an NFA!) to a regular expression

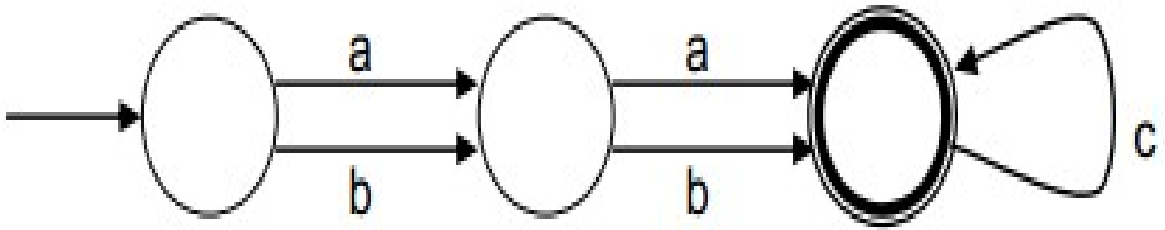
- There are three DFA pieces:
  - A|B
  - AB
  - XA*Y

What are the DFAs for these?

Exercises

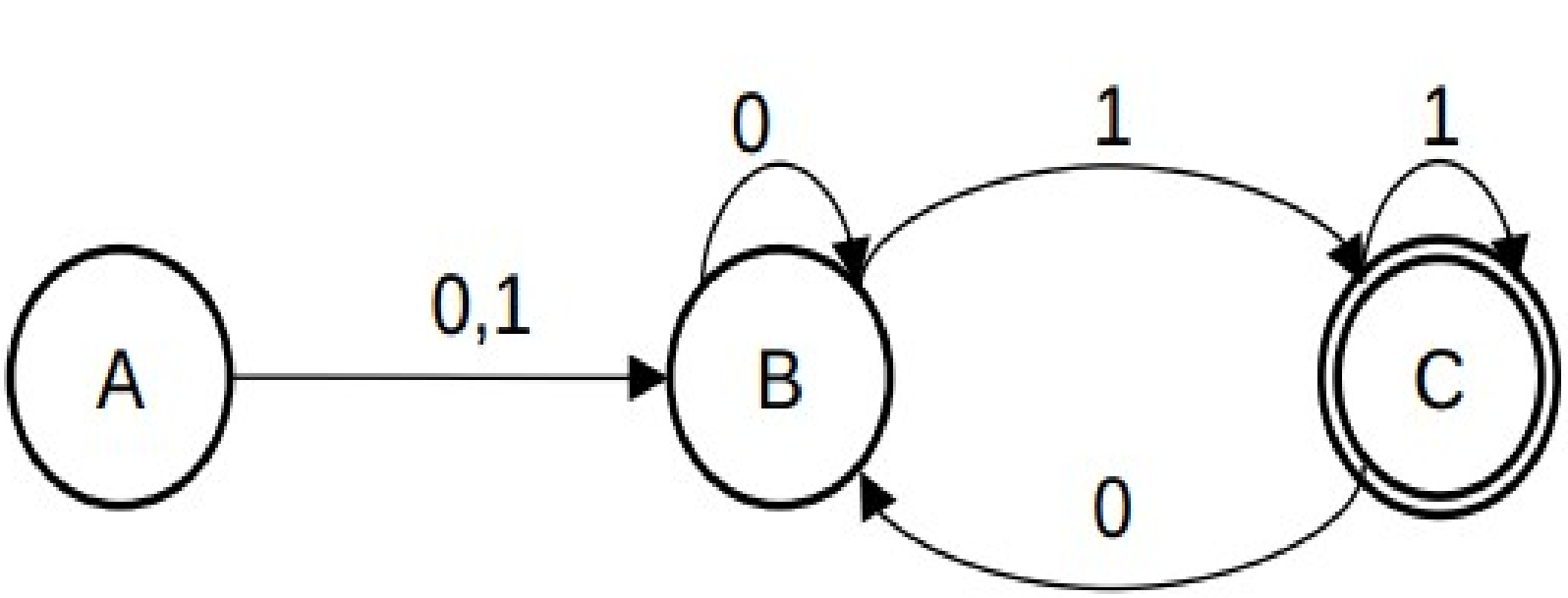
- Determine the regular expressions for the following:





Exercises

- Determine the regular expressions for the following:



Right Linear Grammars -> FSA

- Procedure
  - Non terminals become states
  - Terminals become transitions
  - Is the result deterministic?

- Try converting the following:

S -> aS
  | bA
A -> bA
  | λ

More RLG Examples

- S -> aA
  A -> bB
  | c
  B -> cA

- S -> aB
  | bS
  | λ
  B -> aS
  | bB

Regular Languages

- A regular language is defined as a language that can be expressed using a regular expression
  - Or DFA, NFA, RLG
- So why do we care?
- Is there such thing as a non-regular language?
  - What would be implied by such a language?

Regular Languages

- We care because regular languages are used for scanning
  - Topic for future lecture

- There are non-regular languages as well
  - Consider: matching parentheses
    - Alphabet = {(,)}
    - Language = (^k)^k such that k >= 1
  - Is this regular?
    - Better question: how do we know?
    - If it isn't regular then what is it?

More Practice

- Create an NFA for the following regular expression:

  aa* | aa*ba

- Eliminate the non-determinism
- Minimize the DFA, if necessary