FCS Week 9 Lecture Note

Notebook:	Fundamentals of Computer Science		
Created:	2021-04-13 10:30 AM	**Updated:**	2021-04-30 7:23 PM
Author:	SUKHJIT MANN		

Cornell Notes	**Topic:** Regular Languages: Part 1	Course: BSc Computer Science
		Class: CM1025 Fundamentals of Computer Science[Lecture]
		Date: April 30, 2021

Essential Question:

What is a regular expression and/or language?

Questions/Cues:

- What are regular operations?
- What are some properties of regular operations?
- What are Atomic regular expressions?
- Do the regular operations preserve the regularity?
- What is the precedence of regular operations?
- What is the link between a regular language, regular expression and finite automata?
- How do we convert a finite automaton to a regular expression?

- An Alphabet, Σ, is a non-empty set of symbols
- A string or word is a finite sequence of letters drawn from an alphabet
- Empty strings denoted by ε
- The set of all strings composed from letters in Σ is denoted by Σ^*
- The set of all non-empty strings composed from letters in Σ is denoted by Σ^+
- A language is a collection of strings over an alphabet.

Regular operations

Let L_1 and L_2 be languages.
The following operations are regular operations:

- Union: $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- Concatenation: $L_1 \circ L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- Star: ${L_1}^* = \{x_1 x_2 .. x_m | \text{ m} \geq 0, each\ x_i \in L_1\}$
  - Union is the set of strings that are either in L1 or L2
  - Concatenation is the set of strings of the form xy, where x is in L1 and y is in L2
  - Star operation is a unary operation, L1* means the set of all strings that are concatenations of zero or a few strings drawn from L1

Example

- If A = {red, green, pink} and B = {apple, banana, kiwi}

- $A \cup B$ = {red, green, pink, apple, banana, kiwi}
- $A \circ B$ = {redapple, redbanana, redkiwi, greenapple, greenbanana, greenkiwi, pinkapple, pinkbanana, pinkkiwi}
- A^* = {ε, red, green, pink, redred, redgreen, redpink, greenred, greengreen, greenpink, pinkred, pinkgreen, pinkpink,...}

Properties of regular expression continued

Concatenation and union

- $(A \cup B) \circ C = (A \circ C) \cup (B \circ C)$
- $A \circ (B \cup C) = (A \circ B) \cup (A \circ C)$

Kleene star

- $\emptyset^* = \{\varepsilon\}$
  The Kleene star puts together any number of strings from the language. If the language is empty it can put together 0 strings, resulting in empty string
- $\varepsilon^* = \varepsilon$
- $(A^*)^* = A^*$
- $A^* A^* = A^*$
- $(A \cup B)^* = (A^* B^*)^*$

Atomic regular expressions

- The empty language, $\emptyset$, is a regular expression, which is the empty regular language
- Any letter a in Σ is a regular expression and its language is $\{a\}$
- Empty string, ε, is a regular expression representing the regular language $\{\varepsilon\}$

Compound regular expressions

The regular operations preserve the regularity:

- Concatenation. If R_1 and R_2 are regular expressions, so is $R_1 \circ R_2$
- Union. If R_1 and R_2 are regular expressions, so is $R_1 \cup R_2$
- Kleene star. If R is a regular expression, so is R^*

The language of the regular expression

- What is the language of ab^*?
- $\{a, ab, abb, abbb, \dots\}$
- What is the language of $ab^* \cup b^*$?
- $\{a, ab, abb, abbb, \dots\} \cup \{\varepsilon, b, bb, bbb, \dots\} = \{\varepsilon, a, b, ab, bb, abb, bbb, \dots\}$
- What is the language of $ab^+ \cup b^+b$?
- $\{ab, abb, abbb, \dots\} \cup \{bb, bbb, bbbb, \dots\} = ab^* \cup b^*/\{a, \varepsilon, b\}$

Examples on binary alphabet, $\Sigma = \{a, b\}$

- What is the language of $\Sigma^* a$?
  - $\{a, aa, ba, aaa, aba, baa, bba, \dots\}$
  - What is this language?
  - All binary strings ending in a

- What is the language of $\Sigma^* a\ \Sigma^*$?
  - $\{a, aa, ab, ba, aaa, aab, aba, abb, baa, bab, bba, \dots\}$
  - What is this language?
  - What is it not?
  - It does not contain all b's and empty string
  - Language of all binary words with at least one a

Read regular expressions

- Precedence of operations

*, concatenation, union

- Example: which one of the following is in the language of $a \cup bc^*$?

  1. $bcbc$ 2. $accc$ 3. aaa 4. $bccc$

  The answer is 4.

  $a \cup bc^* = a \cup b(c^*) = a \cup (b(c^*))$

  So the language of this expression is $\{a, b, bc, bcc, bccc, \dots\}$

All binary words containing bb

- $\{bb, abb, bba, bbb, aabb, abba, abbb, babb, bbaa, bbab, bbba, bbbb, \dots\}$
- So we know there is at least one occurrence of bb
- It can have anything including an empty string before it
- It can have anything including an empty string after it
- $(a \cup b)^* bb\ (a \cup b)^*$ or
- $\Sigma^* bb\ \Sigma^*$

All binary words ending with ab or ba

- $\{ab, ba, aab, bab, aba, bba, \ldots\}$
- First look at all binary words ending with *ab*
- There is *ab* at the end; it could have any binary string occurring before
- $\Sigma^* ab$
- Similarly, strings ending with *ba*
- $\Sigma^* ba$
- Take the union $\Sigma^* ab \cup \Sigma^* ba$

Binary strings with at most one a

- What does at most mean? Either 0 or 1
- No occurrence of a: $\{\varepsilon, b, bb, bbb, \ldots\}$
- b^*
- One occurrence of a
- Anything before and after is b *(as many as we like)* $\{a, ab, ba, abb, bab, bba, \ldots\}$
- $b^* a\, b^*$
- Take the union: $b^* a\, b^* \cup b^*$

Example – continued

- Binary strings of length 3
  - $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$
  - $\Sigma\Sigma\Sigma$ (same as sigma to power of three)

- Binary strings of length at least 3
  - It means we have three letters or more
  - Previous example followed by empty string or any strings
  - $\Sigma\Sigma\Sigma\Sigma^*$
  - What is an alternative for $\Sigma\Sigma^*$? Σ^+
  - $\Sigma\Sigma\Sigma^+$

Binary strings of length at most 3

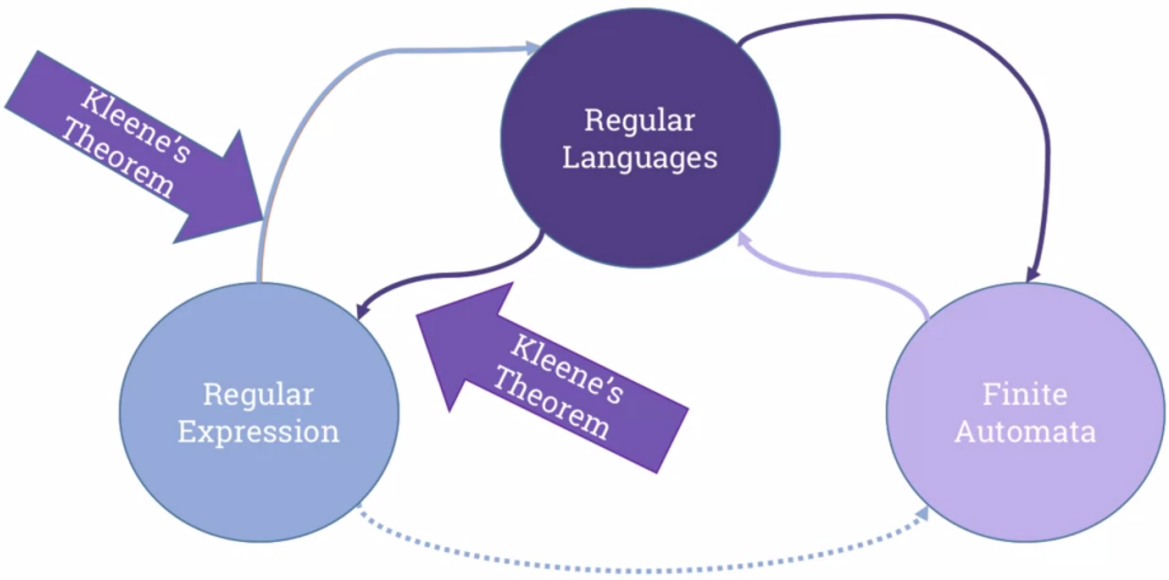
- Binary strings of length 0, $\{\varepsilon\}$
  - Regular expression: ε
- Binary strings of length 1, $\{a, b\}$
  - Regular expression: Σ
- Binary strings of length 2, $\{aa, ab, ba, bb\}$
  - Regular expression: $\Sigma\Sigma$
- Binary strings of length 3, $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$
  - Regular expression: $\Sigma\Sigma\Sigma$
    - $\varepsilon \cup \sum \cup \sum\sum \cup \sum\sum\sum$

Regular language and regular expression

- Kleene's Theorem: a language is regular if and only if it can be described by a regular expression

- If and only if means we need a two-way theorem

1. If a language is described by a regular expression, then it is regular
2. If a language is regular, it can be described by a regular expression

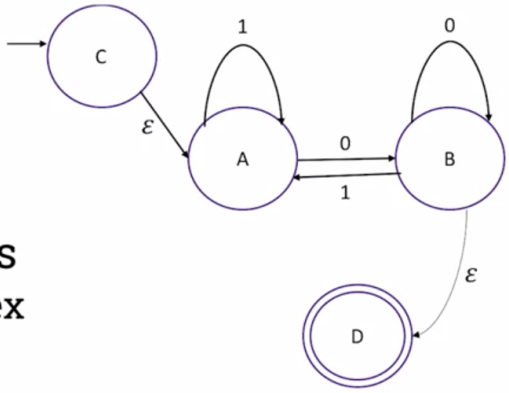


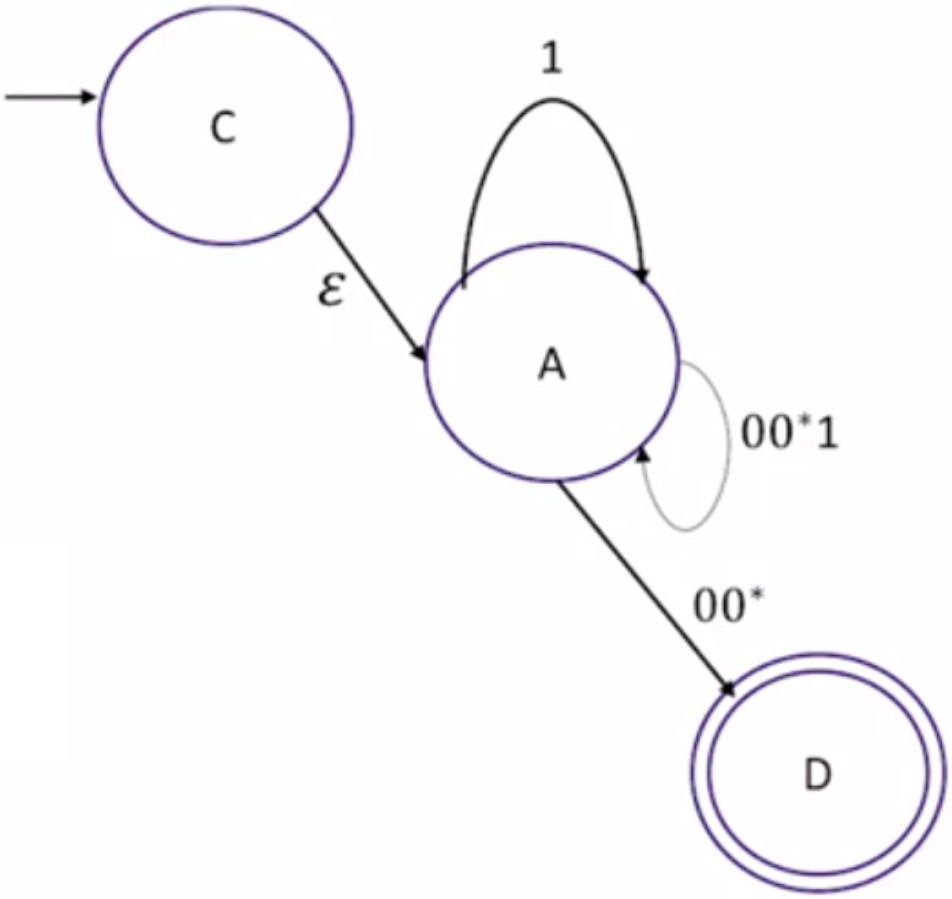
Two main theorems – part 1

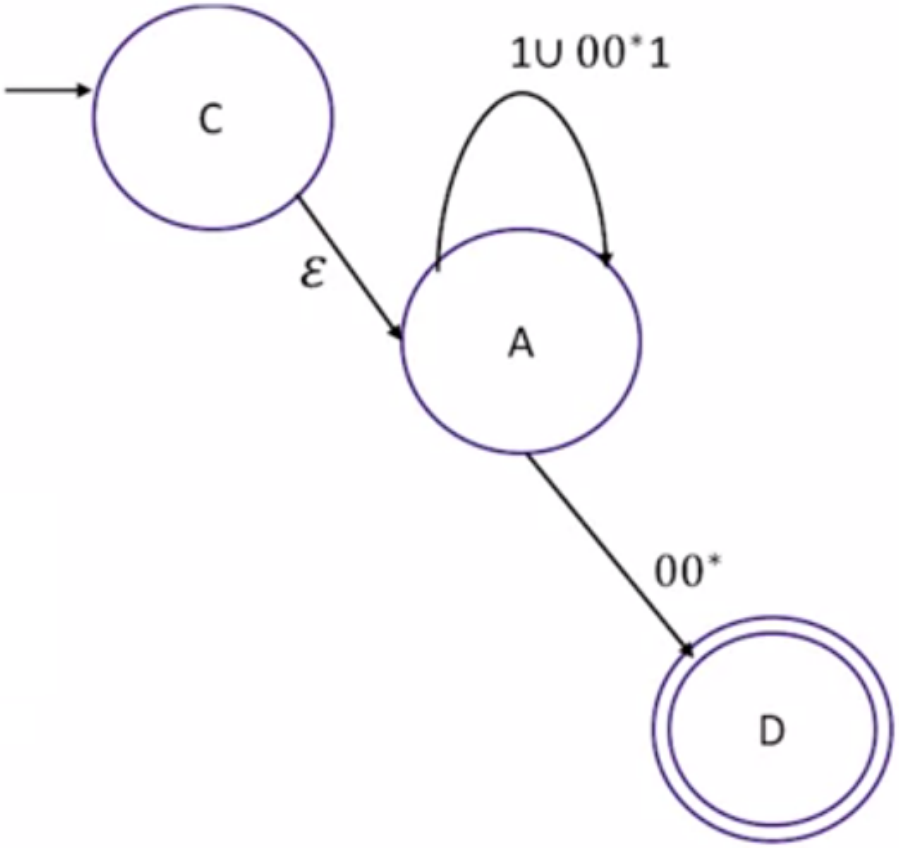
1. If L = L(A) for some finite automaton A, then there is a regular expression R, such that L(R) = L.
   - Proof Idea: can we convert a finite automaton to a regular expression?

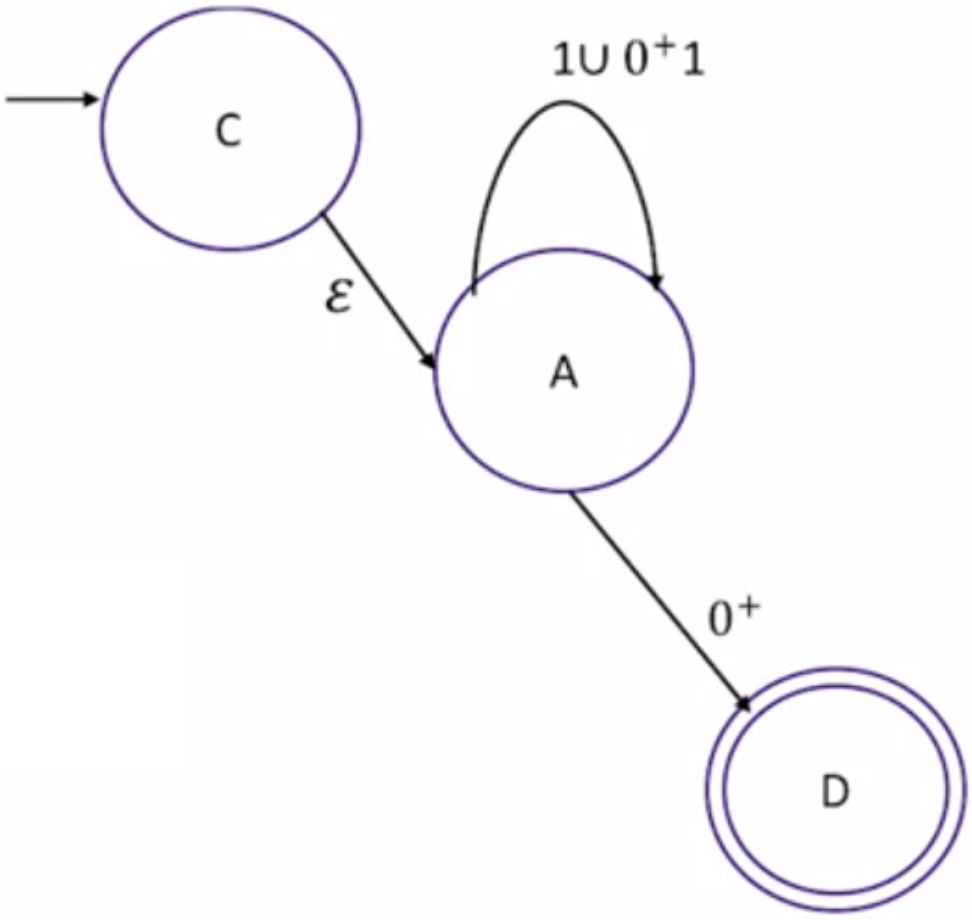
Converting a FA to RE

1. Create a new initial state
   - The transition is ε
2. Create a new final state
   - Connected to final states with transition ε
3. Remove states and transitions
   - Transitions become more complex expressions
4. Remove state B



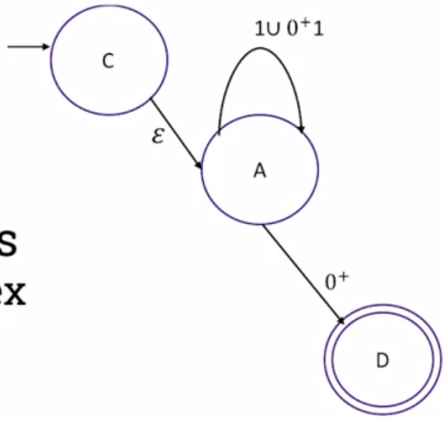

C
1
ε
A
00*1
00*
D


C
1∪ 00*1
ε
A
00*
D



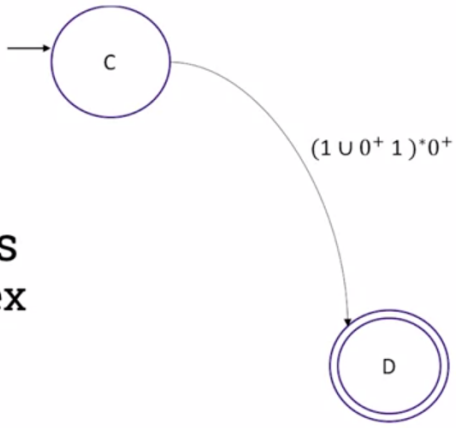
Converting a FA to RE

1. Create a new initial state
   - The transition is ε
2. Create a new final state
   - Connected to final states with transition ε
3. Remove states and transitions
   - Transitions become more complex expressions
4. Remove state B
5. Remove state A



Converting a FA to RE

1. Create a new initial state
   - The transition is ε
2. Create a new final state
   - Connected to final states with transition ε
3. Remove states and transitions
   - Transitions become more complex expressions
4. Remove state B
5. Remove state A

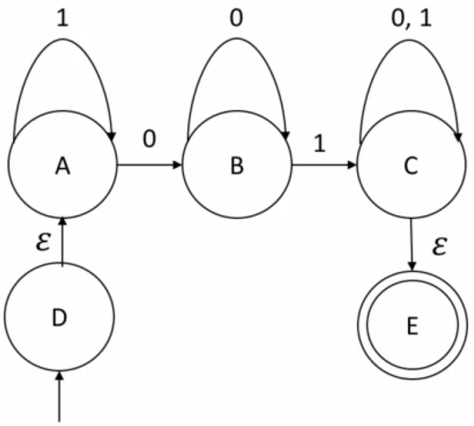


Two main theorems – part 2

2. If L = L(R) for some regular expression R, then there is a finite automaton A such that L(A) = L.

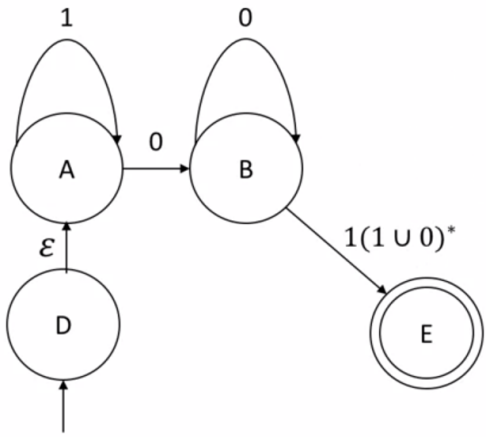
Example 1

1. Create a new initial state, D
2. Create a final state, E
3. Remove state C



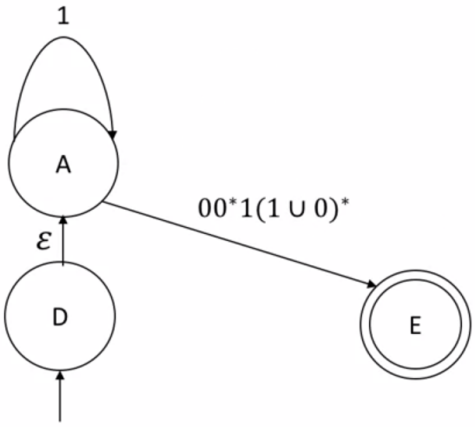
Example 1

1. Create a new initial state, D
2. Create a final state, E
3. Remove state C



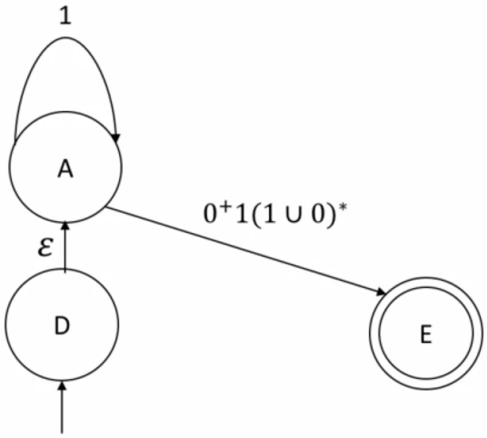
Example 1

1. Create a new initial state, D
2. Create a final state, E
3. Remove state C
4. Remove state B



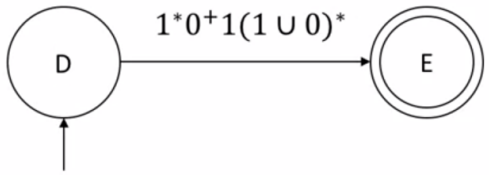
Example 1

1. Create a new initial state, D
2. Create a final state, E
3. Remove state C
4. Remove state B



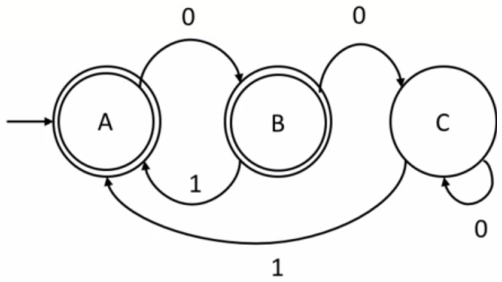
Example 1

1. Create a new initial state, D
2. Create a final state, E
3. Remove state C
4. Remove state B
5. Remove state A



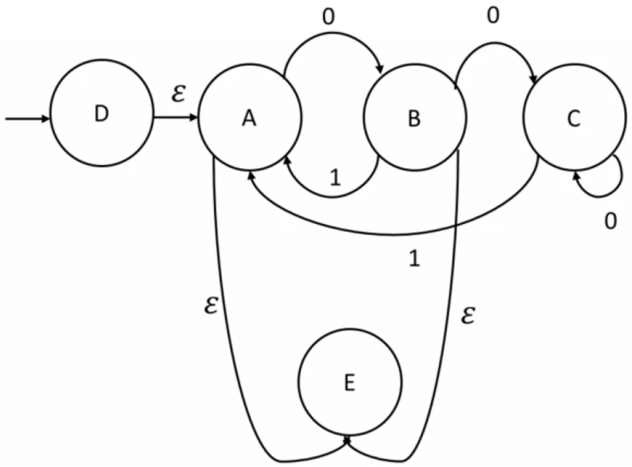
Example 2 – Multiple final states

- Create a new initial state, D



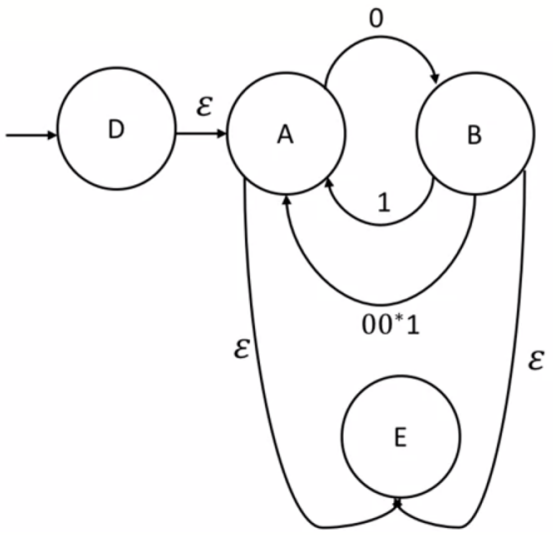
Example 2 – Multiple final states

- Create a new initial state, D
- Create final state E



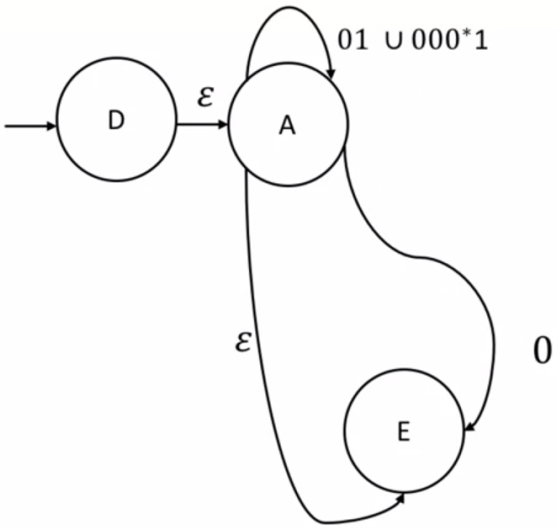
Example 2 – Multiple final states

- Create a new initial state, D
- Create final state E
- Remove C
- Remove B



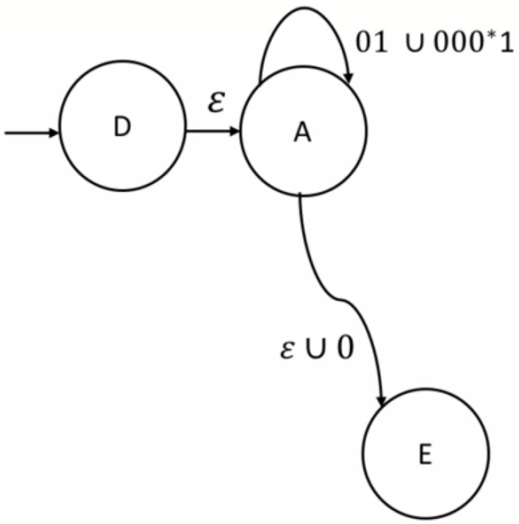
Example 2 – Multiple final states

- Create a new initial state, D
- Create final state E
- Remove C
- Remove B



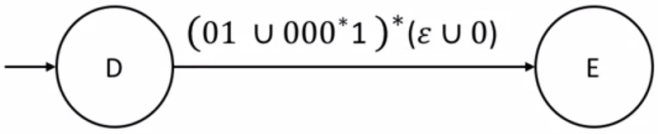
Example 2 – Multiple final states

- Create a new initial state, D
- Create final state E
- Remove C
- Remove B
- Remove A



Example 2 – Multiple final states

- Create a new initial state, D
- Create final state E
- Remove C
- Remove B
- Remove A



Summary

In this week, we learned about regular languages, regular operations, regular expressions and how to convert a finite automata into a regular expression.