**Air Force Institute of Technology**

**Graduate School of Engineering and Management**

**Department of Electrical and Computer Engineering**

**CSCE 532 Automata and Formal Languages**

**Winter 2019**

# Day 4

# nondeterminism (cont.)

# regular expressions

§1.2 Nondeterminism

### Example (Sipser Exercise 1.11)

Prove that every NFA can be converted to an equivalent one that has a single accept state.

#### Solution

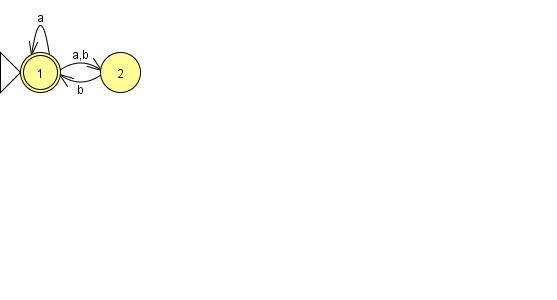
Proof: Let be an NFA. Now define where , , and

Then and only has a single accept state.

Note: To prove the claim that , we would have to show and , i.e. that and . Each of these follows by induction on the length of and is left as an exercise for the student.

### Example (Sipser Exercise 1.16a)

Use the construction given in Theorem 1.39 to convert the following NFA to an equivalent DFA.



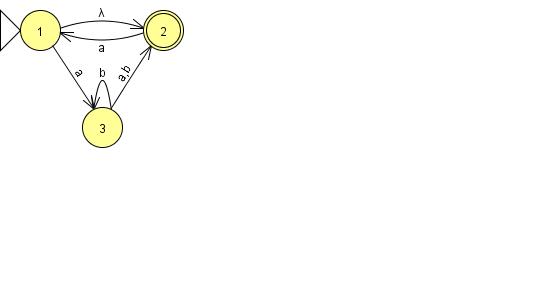
#### Solution

1. is as follows

|  |  |  |
| --- | --- | --- |
|  | a | b |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Practice (Sipser Exercise 1.16B)

Use the construction given in Theorem 1.39 to convert the following NFA to an equivalent DFA.



#### Solution

1. is as follows

|  |  |  |
| --- | --- | --- |
|  | a | b |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Example (Sipser Exercise 1.14b)

Show by giving an example that if is an NFA that recognizes language , swapping the accept and nonaccept states in doesn’t necessarily yield a new NFA that recognizes the complement of . Is the class of languages recognized by NFAs closed under complement? Explain your answer.

#### Solution

The first NFA below recognizes , while the second recognizes .

However, we already know that regular languages are closed under complement (because given a DFA , we can swap the accept and nonaccept states to recognize ). Thus, the counterexample above shows that swapping the accept and nonaccept states of an NFA does not necessarily yield an NFA recognizing , but it does not show that no such NFA exists.

### Definition

Let be a string over . Then the **reverse** of is

where . Also, let be a language. Then the **reverse** of is .

### Theorem

The class of regular languages is closed under reversal.

### Proof

Let be a regular language and a DFA with . Define the NFA where and

Then is regular.

Note 1: Again, to prove that , we would have to show and , i.e. that and . Again, each of these follows by induction on the length of and is left as an exercise for the student.

Note 2: We could also prove this theorem using regular grammars.

### Example (Sipser Exercise 1.32)

Let

contains all size 3 columns of 0s and 1s. A string of symbols in gives three rows of 0s and 1s. Consider each row to be a binary number and let

For example,

Show that is regular. (Hint: Working with is easier.)

#### Solution

Proof: The DFA with the following transition diagram recognizes .

Thus, is regular, and since regular languages are closed under reversal, so is .

Note: State corresponds to columns with a carry-in of zero (and for which no error has occurred previously), state corresponds to columns with a carry-in of one (and for which no error has occurred previously), and state corresponds to columns in which an error occurs (or occurred previously).

§1.3 Regular Expressions and Nonregular Languages

### Example (Sipser Exercise 1.20a)

Give two strings that are members and two strings that are not members of .

### Solution

and are members. and are not members.

### practice (Sipser Exercise 1.20b)

Give two strings that are members and two strings that are not members of .

### Solution

and are members. and are not members.

### Example (Sipser Exercise 1.12)

Let contains an even number of 's and an odd number of ’s and does not contain the substring . Give a DFA with five states that recognizes and a regular expression that generates . (Suggestion: Describe more simply.)

### Solution

can be described more simply as contains an odd number of ’s followed by an even number of 's.

Using JFLAP to convert the above DFA to a regular expression yields[[1]](#footnote-2) . Note that is equivalent to , which is clearly valid. However, is simpler.

### Example (Sipser Exercise 1.18a)

Give a regular expression generating .

### Solution

### Practice (Sipser Exercise 1.18c)

Give a regular expression generating .

### Solution

### Example (Sipser Exercise 1.19a)

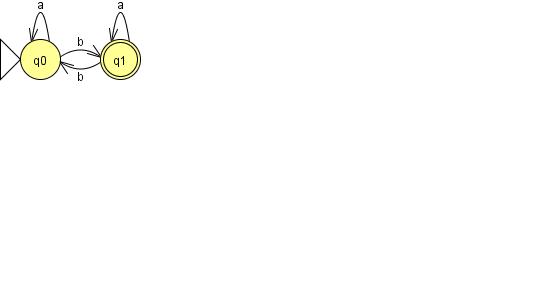
Use the procedure described in Lemma 1.55 to convert to an NFA.

### Solution

I used JFLAP:

Example (Sipser Exercise 1.21a)

Use the procedure described in Lemma 1.60 to convert the following finite automaton to a regular expression.



### Solution

Note that the given FA satisfies JFLAP’s definition of a generalized transistion graph (GTG), but not Sipser’s GNFA. JFLAP generates the regular expression for this FA. The parenthesized term describes strings that take the FA from to itself zero or more times, then to , then to again zero or more times, then back to . A string will be accepted by the FA if and only if it takes the FA through that path zero or more times, then through the first three steps of it.

In order to use the procedure described in Lemma 1.60, we must first define the GNFA corresponding to the above DFA:

1. ,
2. ,
3. is defined by the table below,
4. is the start state, and
5. is the accept state.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

CONVERT()

1. Let be the number of states of .
2. If return .
3. Select . Let be the GNFA ( where and for any and any , let , for , , , and .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. Compute CONVERT( and return this value.

; CONVERT()

1. Let be the number of states of .
2. If return .
3. Select . Let be the GNFA ( where and for any and any , let , for , , , and .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. Compute CONVERT( and return this value.

; CONVERT()

1. Let be the number of states of .
2. If return .

This regular expression describes the set of strings that take the original DFA from to itself zero or more times, then to , then zero or more times through the path that goes from back to , from to itself zero or more times, then back to .

1. Note that JFLAP uses instead of . [↑](#footnote-ref-2)