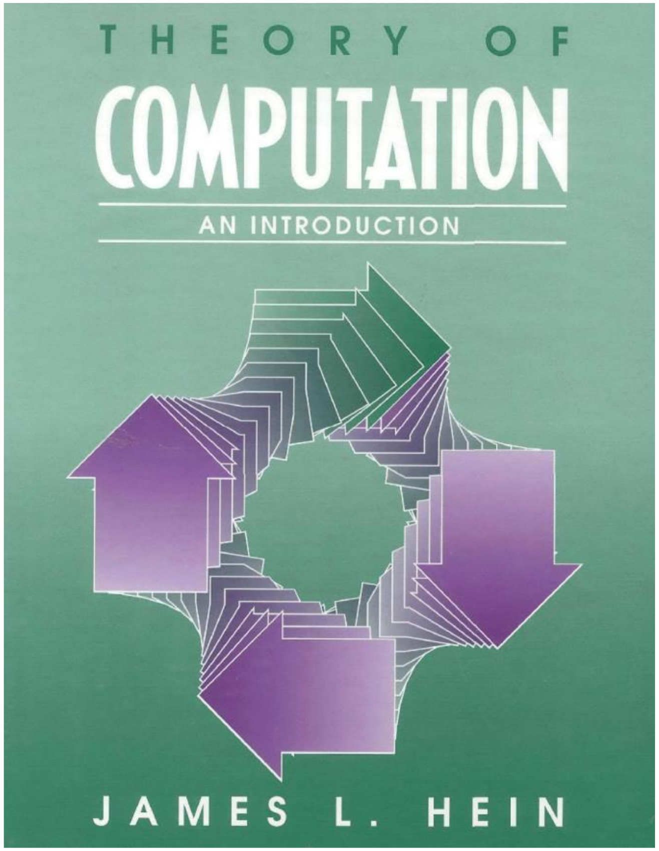
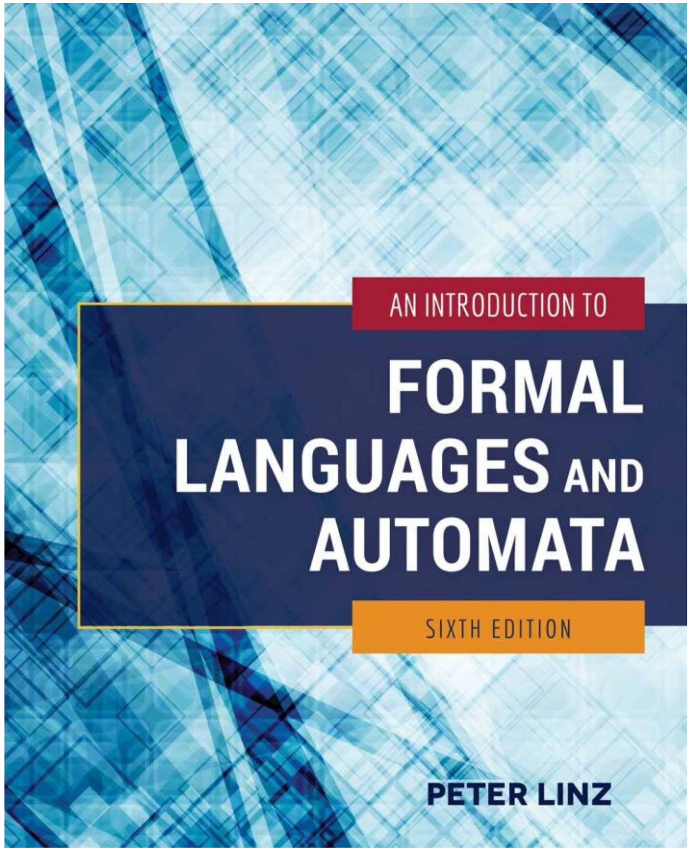
Automata and Formal Languages

Lecture 05

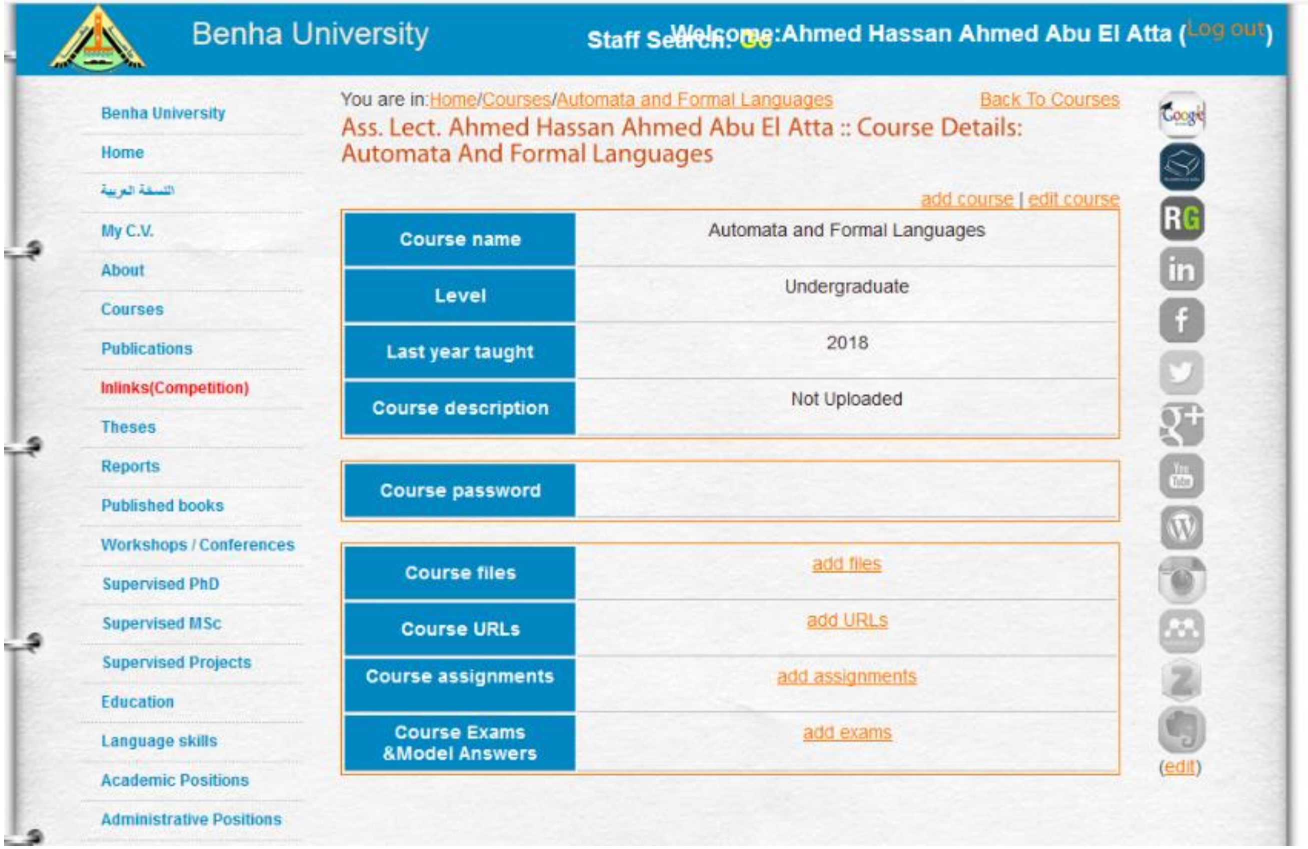
Books


THEORY OF
COMPUTATION
AN INTRODUCTION
JAMES L. HEIN


AN INTRODUCTION TO
FORMAL
LANGUAGES AND
AUTOMATA
SIXTH EDITION
PETER LINZ

PowerPoint

http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14767



You are in: Home/Courses/Automata and Formal Languages   Back To Courses

Ass. Lect. Ahmed Hassan Ahmed Abu El Atta :: Course Details: Automata And Formal Languages

add course | edit course

Course name	Automata and Formal Languages
Level	Undergraduate
Last year taught	2018
Course description	Not Uploaded

Course password	

Course files	add files
Course URLs	add URLs
Course assignments	add assignments
Course Exams &Model Answers	add exams

Regular Expressions & NFA

Agenda

- Regular Expressions
- The Operations Priority
- Languages Associated with Regular Expressions
- RE to NFA
- Examples
- Regular Expression to Finite Automaton
- Examples

Regular Expressions

The set of regular expressions over an alphabet “A” is defined inductively as follows, where + and • are binary operations and * is a unary operation:

Basis:

- Λ, ø, and a are regular expressions for all a ∈ A.

Induction:

- If R and S are regular expressions, then the following expressions are also regular:

(R), R + S, R.S, and R*.

The Operations Priority

*	**highest (do it first),**
•	
+	**lowest (do it last).**

$a + b.a^*$ = $(a + (b.(a^*)))$

Languages Associated with Regular Expressions

$L(\emptyset) = \emptyset$,

$L(\Lambda) = (\Lambda)$,

$L(a) = (a)$ for each $a \in A$,

$L(R + S) = L(R) \cup L(S)$,

$L(R \bullet S) = L(R)L(S)$ (language product),

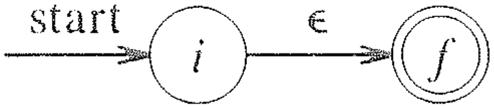
$L(R^*) = L(R)^*$ (language closure).

RE to NFA

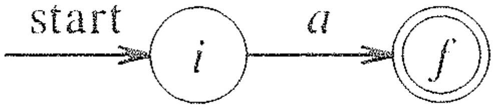
First parse *r* into its constituent sub expressions.

Construct NFA's for each of the basic symbols in *r*.

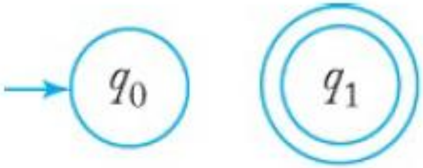
- for ε



- for *a* in Σ

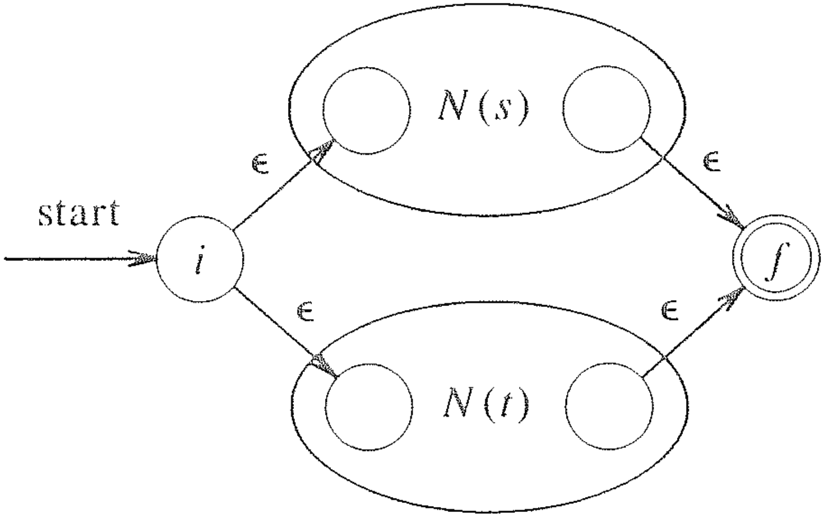


- for *ø*

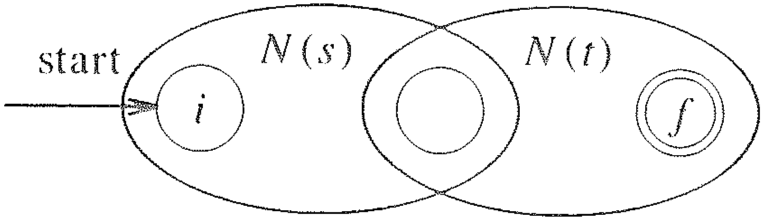


RE to NFA (cont.)

For the regular expression *s+t*,



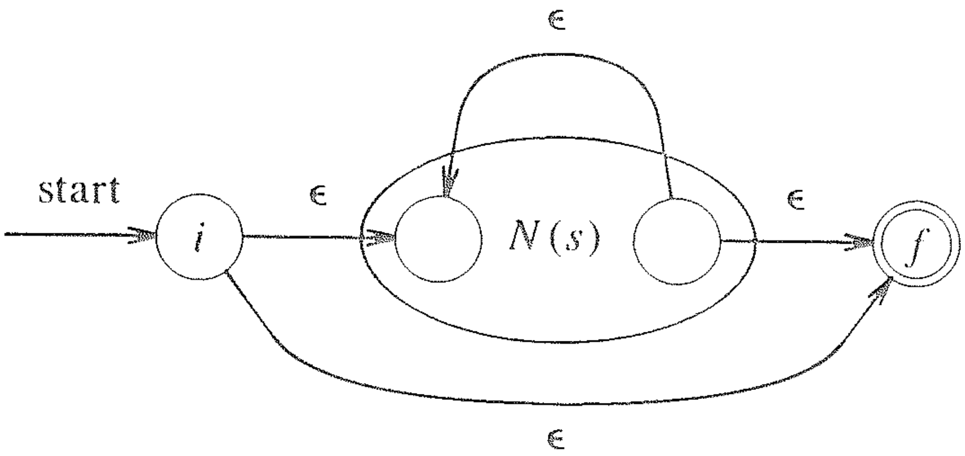
For the regular expression *s.t*,



RE to NFA (cont.)

For the regular expression *s**,

For the parenthesized regular expression (*s*), use *N(s)* itself as the NFA.

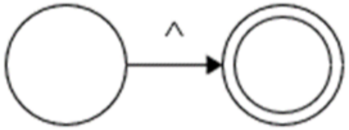


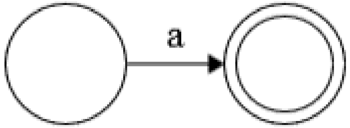
Every time we construct a new state, we give it a distinct name.

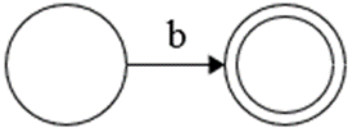
Example 08

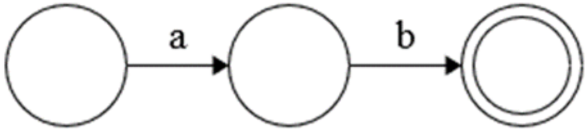
Find an NFA that accepts L (r), where

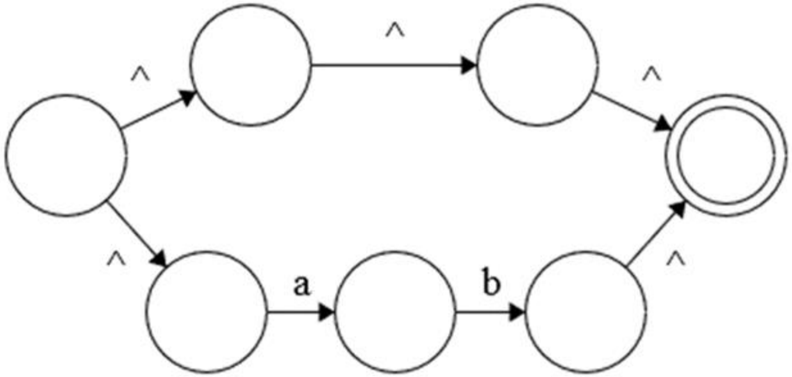
r = (Λ + ab)*

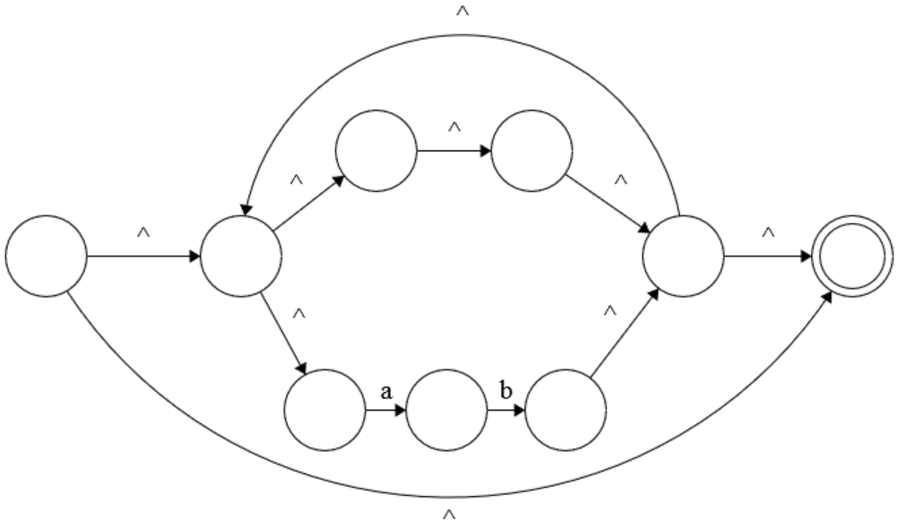












Example 09

Find an NFA that accepts each regular Expression

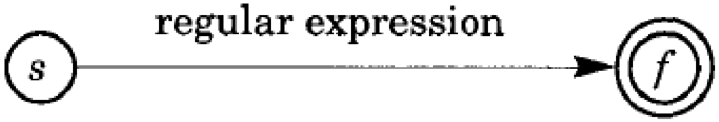
a*a + ab

(aab)*ab

ab*aa

Regular Expression to Finite Automaton

Given a regular expression, we start the algorithm with a machine that has a start state, a single final state, and an edge labeled with the given regular expression as follows:

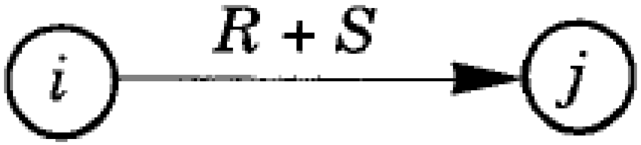


Applying the following rules until all edges are labeled with either a letter or Λ:

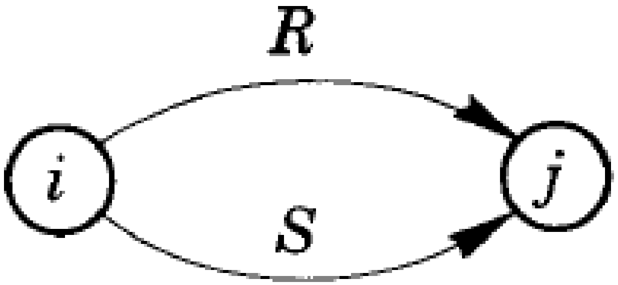
Regular Expression to Finite Automaton (cont.)

1. If an edge is labeled with ø, then erase the edge.

2. Transform any diagram like

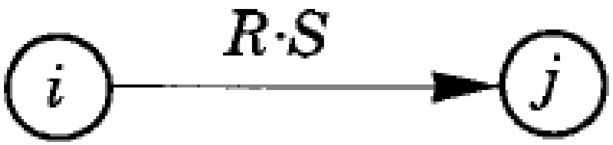


into the diagram

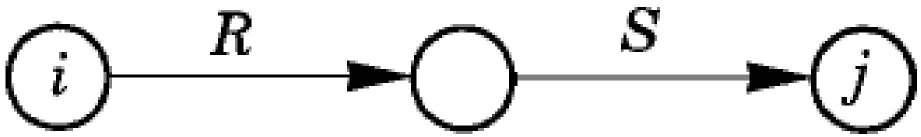


Regular Expression to Finite Automaton (cont.)

3. Transform any diagram like

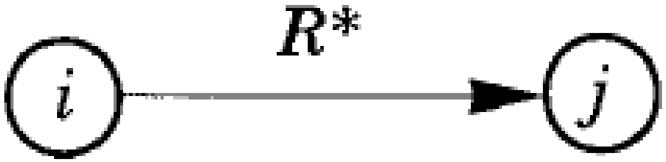


into the diagram

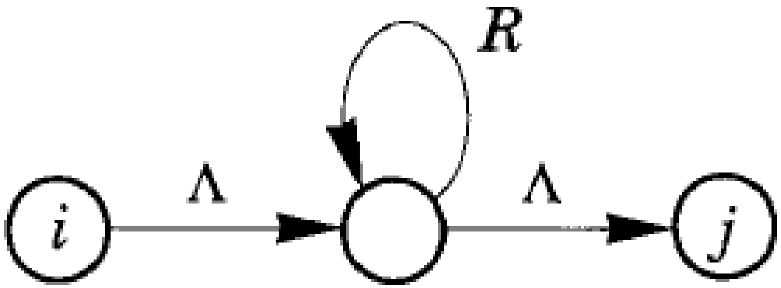


Regular Expression to Finite Automaton (cont.)

4. Transform any diagram like



into the diagram

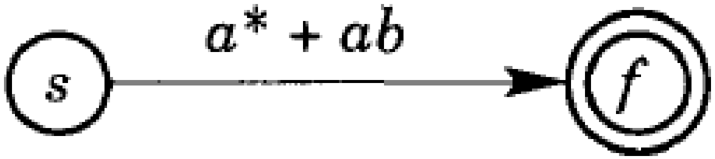


End of Algorithm

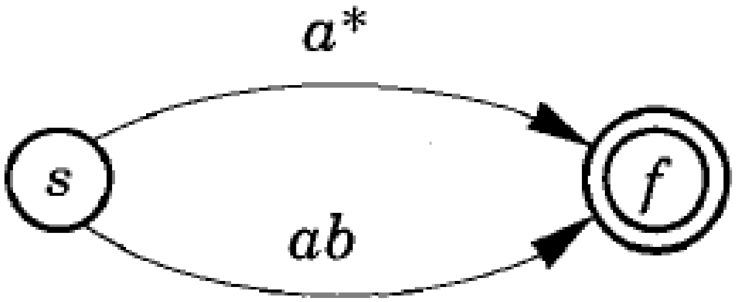
Example 10

Construct an NFA for a* + ab

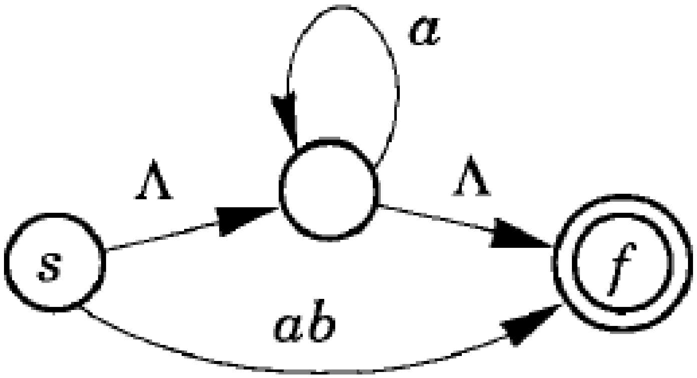
1



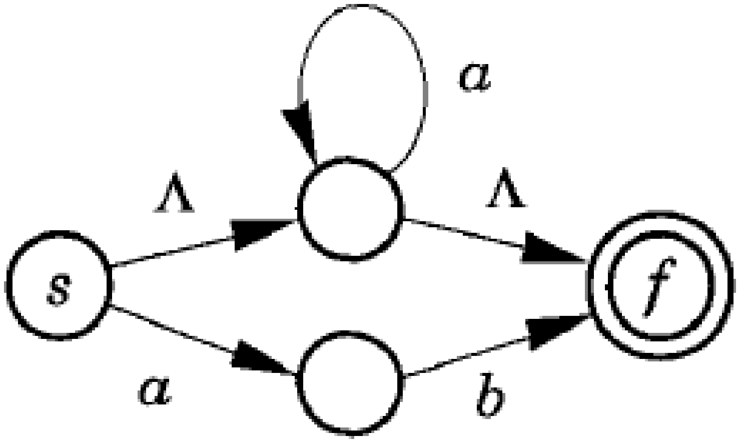
2



3



4



Example 11

Find an NFA that accepts each regular Expression

$$\Lambda + b$$

$$(1 + 01)^*(\Lambda + 0)$$

$$a^*a + ab$$

$$(aab)^*ab$$

$$ab^*aa$$