Regular Expressions

Lecture 10
Sections 3.1 - 3.2

Robb T. Koether

Hampden-Sydney College

Wed, Sep 14, 2016

Outline

1. Regular Expressions
2. Equivalence of Regular Expressions and DFAs
3. Assignment

Outline

1. Regular Expressions
2. Equivalence of Regular Expressions and DFAs
3. Assignment

Regular Expressions

- Regular expressions are like algebraic expressions except that they describe regular languages.
- Examples:
  - $\mathbf{a} \cdot \mathbf{a}^*$
  - $(\mathbf{a} \cdot (\mathbf{a} + \mathbf{b}))^*$
  - $(\mathbf{b} + \mathbf{a} \cdot \mathbf{b}^* \cdot \mathbf{a})^*$

Regular Expressions

Definition (Basic regular expressions)

The basic regular expressions are

- The symbols of the alphabet Σ
- λ
- $\varnothing$

Regular Operators

Definition (Regular operators)

Definition (Regular expression)

A regular expression is one of the basic regular expressions or

- $r_1 + r_2$
- $r_1 \cdot r_2$ (or $r_1 r_2$)
- r_1^*
- (r_1)

where r_1 and r_2 are regular expressions.

The regular operators are union, concatenation, and star.

- Regular expressions are defined recursively.

Language of a Regular Expression

Definition (Language of a regular expression)

The language of a regular expression is defined as follows.

- $L(a) = \{a\}$ for every $a \in \Sigma^*$.
- $L(\lambda) = \{\lambda\}$
- $L(\varnothing) = \varnothing$
- $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
- $L(r_1 \cdot r_2) = L(r_1)L(r_2)$
- $L(r_1^*) = (L(r_1))^*$
- $L((r_1)) = L(r_1)$

where r_1 and r_2 are regular expressions.

Language of a Regular Expression

- Examples

Language of a Regular Expression

- Examples
  - $$\begin{aligned} L(\mathbf{a}+\mathbf{b}) &= L(\mathbf{a}) \cup L(\mathbf{b}) \\ &= \{\mathbf{a}\} \cup \{\mathbf{b}\} \\ &= \{\mathbf{ab}\}. \end{aligned}$$

Language of a Regular Expression

- Examples
  -
$$\begin{aligned} L(\mathbf{a}+\mathbf{b}) &= L(\mathbf{a}) \cup L(\mathbf{b}) \\ &= \{\mathbf{a}\} \cup \{\mathbf{b}\} \\ &= \{\mathbf{ab}\}. \end{aligned}$$
  -
$$\begin{aligned} L(\mathbf{a}\cdot\mathbf{b}^*) &= L(\mathbf{a})L(\mathbf{b}^*) \\ &= \{\mathbf{a}\}\{\mathbf{b}\}^* \\ &= \{\mathbf{a}\}\{\lambda, \mathbf{b}, \mathbf{bb}, \mathbf{bbb}, \ldots\} \\ &= \{\mathbf{a}, \mathbf{ab}, \mathbf{abb}, \mathbf{abbb}, \ldots\}. \end{aligned}$$

Examples

Example (Regular expressions)

Write regular expressions for the following regular languages.

- All strings ending with **a**.
- All strings containing **aba**.
- All strings containing **aba** or **bab**.
- All strings containing **aba** and **bab**.
- All strings not containing **aaa**.

Outline

1 Regular Expressions

2 Equivalence of Regular Expressions and DFAs

3 Assignment

Theorem

A language is regular if and only if it is the language of some regular expression.

Proof ($\Leftarrow$).

- The basic languages $L(a)$, $L(\lambda)$, and $L(\varnothing)$ are regular.
- Union, concatenation, and Kleene star of regular languages are regular.
- Therefore, the language of a regular expression is regular.

$\square$

Regular Languages and Regular Expressions

Proof ($\rightarrow$), beginning.

- Let L be a regular language.
- We need to construct a regular expression r such that $L(r) = L$.
- We will use a generalized transition graph (GTG).

$\square$

Definition

A generalized transition graph is like a regular transition graph except that the labels are regular expressions. To make a transition from one state to another, we must read a *string* that matches the regular expression labeling that transition.

Converting a DFA to a Regular Expression

Proof, continued.

- Begin with a transition diagram for L.
- Replace each label (symbol) with the equivalent regular expression.
- Add a new start state that has no transitions into it, but has one λ-move from it to the original start state.
- Add a new accept state that has no transitions out of it, but has λ-moves into it from all of the original accept states.
- Remove all states from which the accept state is inaccessible.

$\square$

Converting a DFA to a Regular Expression

Proof, continued.

- Note that the number of states in the GTG is at least 3.
- The proof will reduce the number of states down to 2, at which point we will have the regular expression.

□

Converting a DFA to a Regular Expression

Proof, continued.

- Choose a non-initial, non-final state q to be removed.
- For every state p with a transition into q and for every state r with a transition from q, create a transition from p to r, as follows.
- Let
  - r_1 be the label on the transition $p \to q$.
  - r_2 be the label on a loop $q \to q$, if there is one.
  - r_3 be the label on the transition from $q \to r$.
- Apply the label $r_1 r_2^* r_3$ to the new transition.

□

Converting a DFA to a Regular Expression

Proof, concluded.

- After doing this for every combination of transitions into q and out of q, remove state q and all of its transitions.
- Repeat this process until only the initial and final states remain.
- The label on that single remaining transition is the regular expression.

$\square$

Example

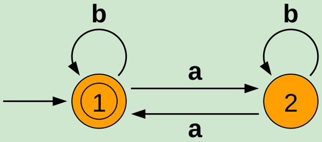
Example (Converting a DFA to a regular expression)

- Find a regular expression for the language

$$L = \{w \mid w \text{ has an even number of } \mathbf{a}\text{'s}\}.$$

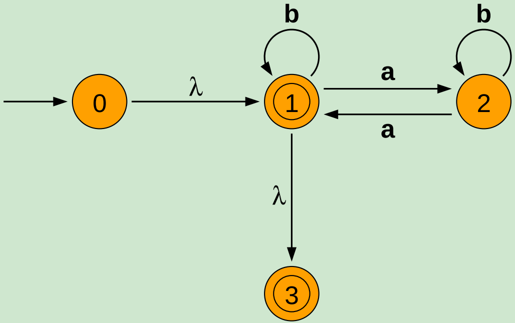
Example

Example (Converting a DFA to a regular expression)



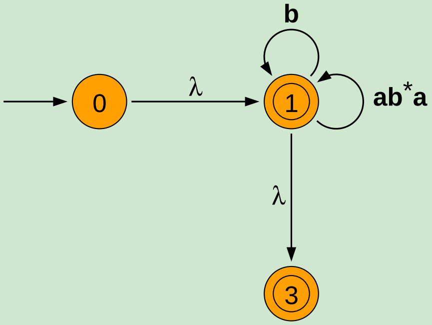
Example

Example (Converting a DFA to a regular expression)



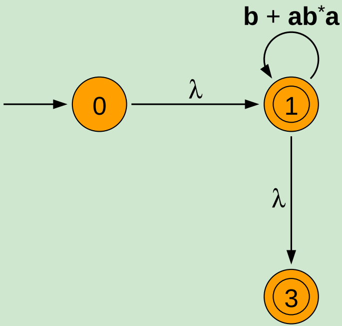
Example

Example (Converting a DFA to a regular expression)



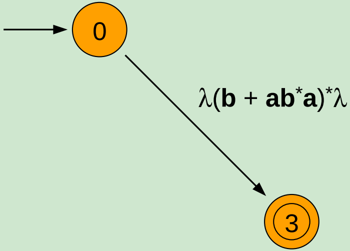
Example

Example (Converting a DFA to a regular expression)



Example

Example (Converting a DFA to a regular expression)



Example

Example (Converting a DFA to a regular expression)

- Therefore,

$$L((\mathbf{b} + \mathbf{ab}^*\mathbf{a})^*) = \{w \mid w \text{ has an even number of } \mathbf{a}\text{'s}\}.$$

- This regular expression "parses" the string

$$\mathbf{babbaababbbaab}$$

as

$$\mathbf{b|abba|aba|b|b|b|aa|b.}$$

Example

Example (Converting a DFA to a regular expression)

- Find regular expressions for the following languages
  - All strings containing an odd number of **a**'s.
  - All strings containing an even number of **a**'s and an even number of **b**'s.
  - All strings that do not contain **aaa**.

Outline

1. Regular Expressions
2. Equivalence of Regular Expressions and DFAs
3. Assignment

Assignment

Assignment

- p. 78: 3, 5, 11, 21, 22, 26, 27.
- p. 90: 1, 7, 10, 12b, 15ac.
- Write a regular expression that will match *****any text*****, where *any text* does not include the string ****.