# CS 473 - Homework 4

Christopher Chapline

February 25, 2015

## 1 Problem 1

**Proof.** Let $E_1 = 0(10)*$ and let $E_2 = (01)*0$.

Basis: Let us first consider $w = 0$. This is accepted by both $E_1$ and $E_2$ because the remainder of their expressions are closed over, meaning that they are not required for acceptance. Thus, our basis is satisfied.

Induction: Let $w$ be a string that is accepted by $E_1$ and $E_2$. Assume that we append a 10 to the end of the string. This will result in a string that is still accepted by both expressions. We know that $w$ is accepted by $E_1$ based on the inductive hypothesis. Knowing this, we can say that $w10$ is accepted because it falls into the pattern of repetition specified by the Kleene closure of (10). We also know that $w$ is accepted by $E_2$ by the inductive hypothesis. Knowing this, we can say that the last 0 in $w10$ is the last zero in the expression $E_2$ and that the preceeding pattern of 01 repetitions is captured by the Kleene closure of 01.

Thus, the expressions $E_1$ and $E_2$ accept the string $w10$. ■

## 2 Problem 2

**Proof.** Let $L_1$ and $L_2$ represent the languages accepted by $E_1 = (\epsilon + 01)*$ and $E_2 = (01)*$ respectively.

Basis: Let $s$ be $\epsilon$. We know that $\epsilon(E*)$ is true for all expressions $E$. Thus, we can conclude that $\epsilon((\epsilon + 01)*)$ is true and that $\epsilon((01)*)$ is true. Our basis is satisfied.

Induction: Let $w$ be a string in $L_1$ and $L_2$. The concatenation of the string 01 onto $w$ would yield the string $w01$. This string is in both $L_1$ and $L_2$ because it represents the pattern 01 which has been concatenated onto a string which is already in both languages, by the inductive hypothesis. Because both $E_1$ and $E_2$ represent closures over the pattern 01, the string $w01$ is in $L_1$ and $L_2$.

■

## 3 Problem 3

**Proof.** Consider the subexpression $E_1 = (b0)*$. We must show the equivalence between $E_1$ and the following automaton:

$$\begin{tikzcd}
3 \arrow[r, bend left, "0"] & 2 \arrow[l, bend left, "b"]
\end{tikzcd}$$

Basis: Consider the empty string. The empty string will start in state 2 and take no transitions, meaning it will stay in the same state. The expression $E_1$ will accept the string based on the definition of the Kleene closure.

Induction: Let $w$ be a string that is accepted by $E_1$ and transitons back to state 2 in the automaton. The concatenation of $b0$ onto $w$ will result in two transitions being taken from state 2 (which is where $w$ takes us based on the inductive hypothesis) thus returning to state 2. In the regular expression, this concatenation is performed on a string already accepted by $E_1$ (inductive hypothesis). This concatenation is accepted based upon the definition of the Kleene Closure.

Thus, the strings accepted by $E_1$ start and end in the same state for the automaton.

■

**Proof.** Consider the subexpression $E_2 = (a1)*$. We must show equivalence between $E_2$ and the following automaton:

$$\begin{tikzcd}
5 \arrow[r, bend right, "a"'] & 6 \arrow[l, bend right, "1"']
\end{tikzcd}$$

Basis: Consider the empty string. The empty string will start in state 5 and take no transitions, meaning it will stay in the same state. The expression $E_2$ will accept the string based on the definition of the Kleene closure.

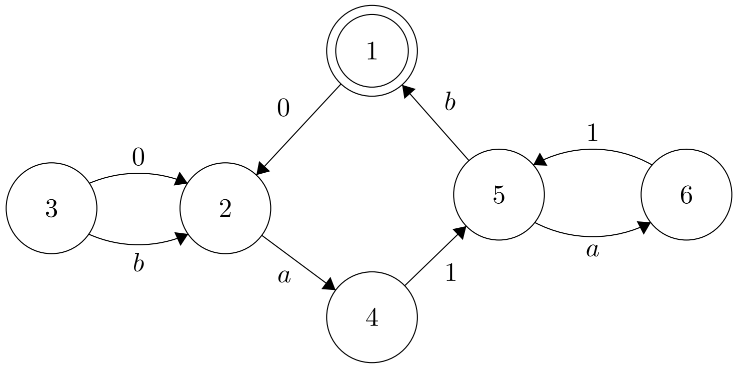
Induction: Let $w$ be a string that is accepted by $E_2$ and transitons back to state

5 in the automaton. The concatenation of $a1$ onto $w$ will result in two transitions being taken from state 5 (which is where $w$ takes us based on the inductive hypothesis) thus returning to state 5. In the regular expression, this concatenation is performed on a string already accepted by $E_2$ (inductive hypothesis). This concatenation is accepted based upon the definition of the Kleene Closure.

Thus, the strings accepted by $E_2$ start and end in the same state for the automaton.

■

**Proof.** We will now prove that the expression $(0(b0)*a1(a1)*b)*$ is equivalent to the following automaton:



Basis: Consider the empty string, $w = \epsilon$. This input will result in the automaton taking no transitions and ending in an accepting state. Similarly, the empty string is in the language of any expression which has been closed over. Therefore, the empty string is in the language of the regular expression and the automaton.

Induction: Let $w$ be a string accepted by $E$ and the automaton. Consider the concatenation of 0, followed by zero or more repetitions of $b0$, $a1$, zero or more repetitions of $a1$, followedly lastly by a $b$. The 0 concatenated onto $w$ will take the automaton to state 2.

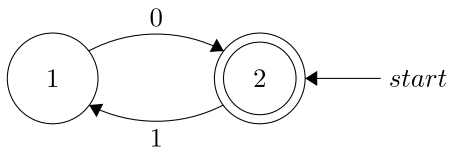
Because $w$ is accepted by the automataon, we know that we are starting the state 1. The zero or more repetitions of $b0$ will return the automaton to state 2 by the first lemma. The $a1$ will take the automaton from state 2 to state 5. The zero or more repetitions of $a1$ will return the automaton to state 5 by the second lemma. The $b$ following the $a1$ repetition will transition the automata back to state 1, which means the string is accepted.

Because $w$ is accepted by $E$ and $E$ is closed over, we can concatenate a string onto $w$. Provided that the string concatenated onto $w$ is accepted by the expression that is closed over, we know that that the regular expression accepts the concatenation. Assume $s = 0\phi a1\alpha b$ where $\phi$ represents zero or more repetitions of $b0$ and $\alpha$ represents zero or more repetitions of $a1$. The acceptance of $\phi$ and $\alpha$ follow from the first and second lemmas respectively. Assuming that $\phi$ and $\alpha$ accept their inputs and then rest of the string concatenated onto $w$ is formed correctly, the regular expression will accept the input.

■

## 4 Problem 4

**Proof.** We will first demonstrate that the subexpression $E_1 = (01)*$ and the following automaton are equivalent:

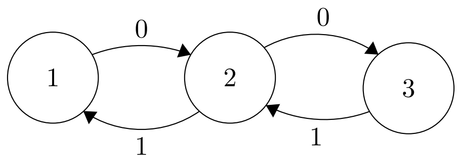


Basis: We will first consider the empty string. On the empty string, the automaton does not transition meaning that the automaton will end at state 2, which is an accepting state. $E$ will accept the empty string because of the definition of the Kleene closure.

Induction: Let $w$ be a string accepted by $E$ and the automaton. Consider the string $s = w10$. By the inductive hypothesis, we know that the automaton will start at state 2. The sequence 01 will transition the automaton to state 1 and then back to state 2. The automaton will accept this input. $E$ will also accept the input because the 10 is a successive concatenation of the pattern 10 which is closed over in $E$. We know that $w$ consists entirely of zero or more repetitions of 10.

■

**Proof.** We will now prove that the expression $E = 0(10) * 0(1(10) * 0)*$ equivalent to the following automaton:



Basis: First consider the string $w = 00$. Starting from the start state (1), $w$ will cause the automaton to transition to state 3. This state is accepting. For

$E$, the first 0 corresponds to the 0 at the start of the pattern and the second 0 corresponds to the 0 after the first $(10)*$ term. The string $w$ is in the language of $E$ because everything after the second 0 is closed over, and is thus optional. This means that $E$ accepts 00.

Induction: Let $w = 0\phi 0\alpha$ where $\phi$ and $\alpha$ are sub-strings that are accepted by $(01)*$ and $(1(10)*0)*$ respectively. There are several cases for addition into $w$:

**Case 1** *In this case, we insert a* 10 *into* $\phi$*. This is still accepted by the automaton based on the lemma above. In other words, as long as the substring* $\phi$ *is still accepted by* $(10)*$*, then the regular expression will accept it and the automaton will still transition into an accepting state at the end. This follows from the inductive hypothesis and the lemma above.*

**Case 2** *Assume that we insert a* 10 *into* $\phi$*. In this case, the automaton is already in an accepting state at the end of* $w$*. The insertion of a* 10 *will result in the automaton translating from state* 3 *to state* 2 *and the back into state* 3*, which is an accepting state. In the expression* $E$*, the addition will be handled by the surrounding elements in* $1(10)*0$*. This will yield acceptance by the expression* $E$*.*

**Case 3** *Assume that we insert a* 1 *followed zero or more repetitions of* 10 *and* 0*. We know that* $w$ *is in the accepting state* 3 *based upon the inductive hypothesis. Because of this, the insertion will have the effect of travleing from state* 3 *to state* 1*, then looping between states* 1 *and* 2 *for the number of repetitions of* 10*, ending on state* 2 *and will then transition to state* 3 *on the final* 0 *and yield acceptance. In the expression, this insertion is handled by definition by the* $(1(10)*0)$ *sub-expression.*

Because all three insertion cases will yield acceptance in both the automaton and the DFA, they accept the same language. ■

## 5 Problem 5

**Proof.** This proof follows directly from the the lemmas presented on slides 8 and 9 in class. Assuming that the first lemma is satisfied and the second lemma is satisfied with the remainder of the input, then the input string is accepted by the DFA and the regular expression.

■

## 6 Problem 6

Recall that $R^k_{i,j} = R^{k-1}_{i,j} + R^{k-1}_{i,k}(R^{k-1}_{k,k}) * + R^{k-1}_{k,j}$.

The k-paths calculuation for this DFA follows:

## 6.1 The 0-paths:

$R^0_{1,1} = \epsilon$, $R^0_{1,2} = \emptyset$, $R^0_{1,3} = 0$

$R^0_{2,1} = \emptyset$, $R^0_{2,2} = \epsilon$, $R^0_{2,3} = 1$

$R^0_{3,1} = 1$, $R^0_{3,2} = 0$, $R^0_{3,3} = \epsilon$

## 6.2 The 1-paths:

$R^1_{1,1} = \epsilon$, $R^1_{1,2} = \epsilon$, $R^1_{1,3} = 0 + 00$

$R^1_{2,1} = \epsilon$, $R^1_{2,2} = \epsilon$, $R^1_{2,3} = 1 + 01$

$R^1_{3,1} = 1$, $R^1_{3,2} = 0 + 1$ $R^1_{3,3} = \epsilon + 01$

## 6.3 The 2-paths:

$R^2_{1,1} = \epsilon$, $R^2_{1,2} = \epsilon$, $R^2_{1,3} = 0 + 00 + 1 + 01$

$R^2_{2,1} = \epsilon$, $R^2_{2,2} = \epsilon$, $R^2_{2,3} = 1 + 01$

$R^2_{3,1} = 1 + 0$, $R^2_{3,2} = 0 + 1$, $R^2_{3,3} = \epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)$

## 6.4 The 3-paths:

$R^3_{1,1} = \epsilon + (1 + 0)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (0 + 00 + 1 + 01)$, $R^3_{1,2} = \epsilon + (1 + 01)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (0 + 1)$ $R^3_{1,3} = (0 + 00 + 1 + 01) + (\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1))$

$R^3_{2,1} = \epsilon + (1 + 01)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (1 + 0)$, $R^3_{2,2} = \epsilon + (0 + 1)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (1 + 01)$, $R^3_{2,3} = (1 + 01) + (\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1))(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (1 + 01)$

$R^3_{3,1} = (1 + 0) + (1 + 0)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1))$, $R^3_{3,2} = (0 + 1) + (0 + 1)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (\epsilon + 01 +$

$(1+01)(\epsilon+01)*(0+1))$, $R^3_{3,3} = (\epsilon+01+(1+01)(\epsilon+01)*(0+1))+(\epsilon+01+(1+01)(\epsilon+01)*(0\text{+}1))(\epsilon\text{+}01\text{+}(1\text{+}01)(\epsilon\text{+}01)*(0\text{+}1))*(\epsilon\text{+}01\text{+}(1\text{+}01)(\epsilon\text{+}01)*(0\text{+}1))$

The final regular expression generated by k-paths is $((0 + 00 + 1 + 01) + (\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1))) + ((1 + 0) + (1 + 0)(\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)) * (\epsilon + 01 + (1 + 01)(\epsilon + 01) * (0 + 1)))$

This regular expression is much, much longer and more difficult to read than $0(01 + 10) * 0$. This is a standard pattern for regular expressions generated through the k-paths expansion method. They generally create very long and very complex regular expressions that are far from the optimal solution.