Regularitet og Automater: Assignment 1

Rasmus Dalsgaard (201605295)

April 2017

1 Introduction

Given the definition of $prefix$, define a compliant and similar function, $Prefix$ for a language $S \subseteq \Sigma$ such that $Prefix(S)$ is the language consisting of prefix strings from S:

$$Prefix(S) = \cup_{x \in s} prefix(x)$$

1.1 Subproblem 1

Let the regular expression r be $a + bc$.
What are the two languages $L(r)$ and $Prefix(L(r))$?

Defining L to be a function over language, $L(r)$ is a function for computing any r such that $r' = L(r)$. Much alike, $Prefix(L(r))$ computes the language abiding prefixes of r.
Obeying the magnitudes of precedence, any concatenation must be done before alternations like the *or* operator $+$, resulting in L function on a *or* bc, yielding the result of above example as

$$L(r) = L(\{a + bc\}) = L(\{a\}) \cup L(\{bc\}) \Rightarrow \{a, bc\}$$

Knowing that

$$prefix(x) = \{y \in \Sigma^* | \exists z \in \Sigma^* \ni x = yz\}$$

. We may conclude that $\forall x$, $prefix(x)$ must return the entire set of prefixes for x. I.e.

$$prefix(\{ab\}) \Rightarrow \{\Lambda, a, ab\}$$

. Expanding upon this formula and combining the knowledge of $L(r)$, extracting the full set of r by $Prefix(L(r))$ will terminate in

$$Prefix(L(r)) = Prefix(L\{a + bc\}) \Rightarrow \{\Lambda, a, b, bc\}$$

1.2 Subproblem 2: Proof by induction

Construct a proof for closedness of the class of regular expressions of $Prefix$, showing that if S is a regular language, then $Prefix(S)$, too, is a regular language.

Base cases

Given an arbitrary regular expression r, I theorize that any r' exists such that $Prefix(L(r)) = L(r')$. This may be unveiled by first setting up base cases. First, handling the empty language $r = \emptyset$ is defined as $r' = \emptyset$:

$$Prefix(L(r)) = Prefix(\emptyset) = \emptyset = L(r')$$

For something a bit more tangible, a language with a single character $r = a \in \Sigma^*$ such that $r' = a$:

$$Prefix(r) = Prefix(a) \rightarrow \{\Lambda, a\} = L(r')$$

Lastly, a case with multiple substrings of r may exist such that it be comprised of any functional operation applied to r: $r = r_i$ and r_j, be in concatenation, alternation, Kleene* or similar operations.

Hypothesis

Assume that $Prefix(L(r_i)) = L(r_i')$ such that substrings of r can be described, in part or in whole, by $Prefix(L(r_1'))$ and $Prefix(L(r_2'))$.

Case 1: $r = r1r2$

Before we go any further, let's clear out a special case of concatenating any r_1 with $\emptyset$

$$r = r_1 r_2 = r_1\emptyset = \emptyset$$

Inductive step

Let $r = r_1 r_2$ such that $r' = r_1' + r_1 r_2'$. Operating on the assumption that

$$Prefix(L(r_i)) = L(r_i')$$

and that $r = r_1 r_2$, we can conclude that

$$Prefix(L(r)) = Prefix(L(r_1 r_2))$$

, but only as long as $\forall r_i \neq \emptyset$. This may, in turn, be expanded as $Prefix(L(r_1)L(r_2))$. We can now perform the $prefix(x)$ on a larger scale, expanding the formula even further, such that

$$\cup_{z \in L(r_1)L(r_2)} prefix(z)$$

is diffused into $\cup_{x\in}$ for any x in either $L(r_1)$ or for any y in $L(r_2)$ so that $z = xy$:

$$\cup_{x\in L(r_1)\wedge y\in L(r_2)} prefix(xy)$$

Using the given lemma, we may now summarize that

$$Prefix(L(r)) = \cup_{z\in L(r_1)L(r_2)} prefix(z) = \cup_{x\in L(r_1)\wedge y\in L(r_2)} prefix(x) \cup \{x\} prefix(y)$$

Taking advantage of the given lemma c, saying that

$$\forall A, B \subseteq \Sigma^* : \cup_{x\in A} \cup_{y\in B} \{x\} \cdot \{y\} = \cup_{x\in A}\{x\} \cdot \cup_{y\in B}\{y\} = A \cdot B$$

we can plug in the results directly, as long as $L(r_1), L(r_2) \subseteq \Sigma^*$, and extract that

$$Prefix(L(r_1)) \cup L(r_1) Prefix(L(r_2)) = L(r_1') \cup L(r_1)L(r_2') = L(r_1' \cup r_1 r_2') = L(r')$$

Case 2: $r = r_1 + r_2$

Inductive step

Let $r = r_1 + r_2$ such that $r' = r_1' + r_2'$. Operating on the assumption that

$$Prefix(L(r_i)) = L(r_i')$$

and that $r = r_1 + r_2$ we can conclude that

$$Prefix(L(r)) = Prefix(L(r_1 + r_2))$$

Unioning the regular expressions extracted from L, we can rewrite the previous formula as $Prefix(L(r_1) \cup L(r_2))$. Taking advantage of the given lemma b, we can affirm that

$$\cup_{x\in L(r_1)\cup L(r_2)} prefix(x)$$

can be expanded to unioning the strings of $r_1 prefix(x)$ with the analogue of r_2:

$$\{\cup_{x\in L(r_1)} prefix(x)\} \cup \{\cup_{x\ inL(r_w)} prefix(x)\}$$

which, in turn, is simplified to

$$Prefix(L(r_1)) \cup Prefix(L(r_2))$$

This leaves us with $Prefix(L(r_1)) = L(r_1')$ and $Prefix(L(r_2'))$. Combined with the previous simplified formula, we're handed the solution

$$Prefix(L(r_1 + r_2) = \cup_{x\in L(r_1)\cup L(r_2)} prefix(x) = (L(r_1') \cup L(r_2')) = L(r_1' + r_2') = L(r')$$

Case 3: $r = r_1^*$

Inductive step

Let $r = r_1^*$ such that $r' = \Lambda + r_1^* r_1'$. Assuming that

$$Prefix(L(r)) = L(r')$$

we start out by showing that

$$Prefix(L(r)) = Prefix(L(r_1^*)) = Prefix(L(r)^*)$$

This works due to the Kleene star operation; if $r = (ab)^*$ then

$$L(r) = L((ab)^*) = L(ab)* = \{ab\}^*$$

Summing up the total unions of the prefixes of any L extricated from r, we may rewrite the formula as

$$Prefix(\bigcup_{i=0}^{\infty} L(r_1)^i)$$

and this, because of the former attribute, is equal to

$$\bigcup_{i=0}^{\infty} Prefix(L(r_1))$$

There's a hitch, though, because the original formula bounded $i > 0$, so we're going to get a little creative and extract the $i = 0$ element from L and, which, thanks to the definition, is fairly simple:

$$Prefix(L(r_1)^0) = \{\Lambda\}$$

Unioning the empty string onto the remaining function is merely a matter of

$$Prefix(L(r_1)^0) \cup \bigcup_{i=1}^{\infty} Prefix(L(r_1)^i)$$

Applying the given lemma of

$$\forall i > 0 \wedge S \subseteq \Sigma^* : Prefix(S^i) = \cup_{k=0\ldots i-1} S^k Prefix(S)$$

we can now simplify the previous formula by stating that

$$Prefix(L(r_1)^0) \cup \bigcup_{i=1}^{\infty} Prefix(L(r_1) = \{\Lambda\} \cup \bigcup_{i=1}^{\infty} L(r_1)^i Prefix(L(r_1))$$

bending it towards the layout of the lemma

$$\{\Lambda\} \cup \bigcup_{k=0}^{i-1} L(r_1)^k Prefix(L(r_1))$$

making it easier to distribute the Kleene star rewrite yet again:

$$\{\Lambda\} \cup L(r_1)^* Prefix(L(r_1))$$

Shortening this even further, we can venture into the lands of

$$\{\Lambda\} \cup L(r_1)^* L(r_1') = \{\Lambda\} \cup L(r_1^* r_1')$$

And, to summarize this obscure adventure, the shortened version is

$$Prefix(L(r)) = Prefix(L(r_1^*)) = Prefix(L(r)^*) = L(\Lambda + r_1^* r_1') = L(r')$$

Proving that the set of prefixes over the language of any regular expression r, is, in fact, also a regular language, regardless or concatenation, alternation or Kleene closure, during which the class of regular languages are closed.