Excercises

Chapter 2

Makabaka1880 [1]

1. Reducted for privacy

This is a sample document to demonstrate what cyan-report is capable of. The problems below come from the book *An Introduction to Formal Languages and Automata*.

1. Example Usage of Containers

We are given a language

$$L = \{ab, a, baa\}$$

And sentences

$$\begin{aligned} u_1 &= abaabaaabaa \\ u_2 &= aaaabaaaa \\ u_3 &= baaaaabaaaab \\ u_4 &= baaaaaabaa \end{aligned}$$

Problem

Which of the strings are in L^*?

Solution. In order for a sentence to be in the star-closure of a language L, then the sentence must be constructable from concatenations of substrings in L. Let's prove this lemma.

Lemma 1. $u \in L^* \iff \exists w_1, w_2, ..., w_n \in L, u = w_1 w_2 ... w_n$

Proof. From the definition of star-closure we have

$$u \in L^* \iff u \in \bigcup_{i=0}^{\infty} L^i$$

Therefore u must be in at least on of L^n. An induction n can be done, starting from $n = 0$ as the base case. Here, $u = \lambda$, wIHch exactly the empty concatenation of elements of L.

Now suppose the lemma holds for n. For $n + 1$ by definition

$$L^{n+1} = L^n L = \{vw : v \in L^n, w \in L\}$$

Therefore if $u \in L^{n+1}$, then u must be the concatenation of some v from L^n and w from L. By the inductive hypothesis, v is the concatenation of substrings from L, thus u is the concatenation of substrings from L and a substring from L, thes the concatenation of substrings of L. ■

Therefore in order for a sentence to be in L^*, one just need to guaruntee that the sentence can be constructed from $\{ab, a, baa\}$.

One can easily check that

$$\begin{aligned} u_1 &= abaaabaaabaa \\ u_2 &= aaaabaaaa \\ u_3 &= baaaaabaaaab \\ u_4 &= baaaaaabaa \end{aligned}$$

So the answer is **all of them are sentences under L^***.

Problem

Which of the strings are in L^4?

Solution. u_2 and u_3 are.

Proof. Previously constructed concatenations proved that

$$u_2, u_4 \in L^3 \subseteq L^4$$

By a DFS search it can be easily proven that $u_1, u_3 \notin L^4$.

set $L' \leftarrow L \cup L^0$
for s_1 in L' **do**
  for s_2 in L' **do**
    for s_3 in L' **do**

for s_4 in L' **do**
set $s \leftarrow s_1 s_2 s_3 s_4$
if $s = u_1$ **then return** s
else continue
s **not found**

Following is an implementation in `C99`.

```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

#define MAX_LEN 100

// L' = L U L^0
const char* L_prime[] = {"", "a", "aa"};
const int L_prime_size = 3;

bool search_u1(const char* target) {
    char concat[MAX_LEN];

    for (int i = 0; i < L_prime_size; i++) {
        for (int j = 0; j < L_prime_size; j++) {
            for (int k = 0; k < L_prime_size; k++) {
                for (int l = 0; l < L_prime_size; l++) {
                    // s = s1 s2 s3 s4
                    snprintf(concat, MAX_LEN, "%s%s%s%s",
                            L_prime[i], L_prime[j],
                            L_prime[k], L_prime[l]);
                    if (strcmp(concat, target) == 0) return true;
                }
            }
        }
    }
    return false; // s not found
}
```

$\blacksquare$