Theory of Computability Midterm 1

Elizabeth Hunt (A02364151)

October 6, 2023

1 Problem 1

1.1 Stage 1

We skip Stage 1; there are no productions in the form $A \to BC$ or $A \to s$.
$P' = \{\}$

1.2 Stage 2

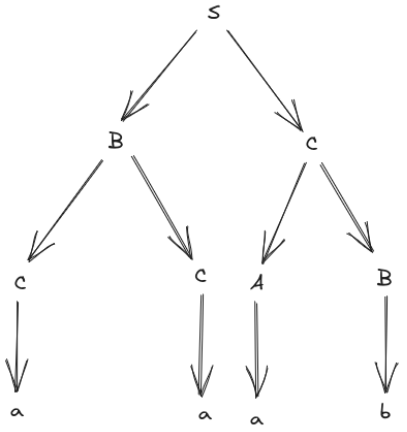
$P' = \{C_a \to a, C_b \to b, C_c \to c, C_d \to d\}$
And our new productions are $\{S \to C_aSC_bC_b, S \to C_aSC_a, S \to C_bSC_aC_a, S \to C_bSC_b, S \to C_cC_d\}$

1.3 Stage 3

We replace $S \to C_aSC_bC_b$ with $\{S \to C_aD_1, D_1 \to SD_2, D_2 \to C_bC_b\}$
We replace $S \to C_aSC_a$ with $\{S \to C_aD_3, D_3 \to SC_a\}$
We replace $S \to C_bSC_aC_a$ with $\{S \to C_bD_4, D_4 \to SD_5, D_5 \to C_aC_a\}$
We replace $S \to C_bSC_b$ with $\{S \to C_bD_6, D_6 \to SC_b\}$.
We add $S \to C_cC_d$ as it is in CNF already.
Thus,

$$
\begin{aligned}
P' = &\{C_a \to a, C_b \to b, C_c \to c, C_d \to d\} \\
&\cup \{S \to C_aD_1, D_1 \to SD_2, D_2 \to C_bC_b\} \\
&\cup \{S \to C_aD_3, D_3 \to SC_a\} \\
&\cup \{S \to C_bD_4, D_4 \to SD_5, D_5 \to C_aC_a\} \\
&\cup \{S \to C_bD_6, D_6 \to SC_b\} \\
&\cup \{S \to C_cC_d\}
\end{aligned}
$$

2 Problem 2



Yes, we can recognize the string by this derivation.

3 Problem 3

Because strings in $L(M_1)$ and $L(M_2)$ are recognized by Discrete Finite Automata, they must be regular languages.

By the Myhill-Nerode theorem, if L is a regular language it can be recognized by a unique DFA with a minimal number of states. In other words, we know that if two DFA recognize the same language, they must have the same minimal DFA.

Let minimize(D) be the minimization algorithm given in Lecture 04 returning a deterministic set of states.

Then, we know M_1 is equivalent to M_2 when minimize(M_1) is congruent to minimize(M_2). This is only true when all descriptors (Σ, q_0, δ, etc...) are also equivalent.

In the below pseduo code we just check the equivalence of the set of states, alphabet, and start state. Then we perform a search to see if $(\delta_1) = M_1$ is $\subseteq$ of $(\delta_2) = M_2$ and $\delta_2 \subseteq \delta_1$, and if both are true, then $\delta_1 = \delta_2$.

If all are equivalent, then the languages recognize the same strings!

```
def minimize(dfa):
    minimized = dfa.copy()
    # ... mutate minimized according to minimize()
    return minimized

def delta_subseteq(start_state, sigma, delta1, delta2):
    visited = set()
    for transition in delta2.keys():
        if transition not in delta1 or \
               delta1[transition] != delta2[transition]:
            return False
    return True

def equivalent(m1, m2):
    minimized_m1 = minimize(m1)
    minimized_m2 = minimize(m2)
    if minimized_m1.Q != minimized_m2.Q or \
           minimzed_m1.sigma != minimized_m2.sigma or \
```

```
        minimized_m1.q0 != minimized_m2.q0 or \
        minimized_m1.F != minimized_m2.F:
    return False

m2_delta_includes_m1_delta = delta_subseteq(minimized_m1.q0, \
                                            minimized_m1.sigma, \
                                            minimized_m1.delta, \
                                            minimized_m2.delta)

m1_delta_includes_m2_delta = delta_subseteq(minimized_m2.q0, \
                                            minimized_m2.sigma, \
                                            minimized_m2.delta, \
                                            minimized_m1.delta)

return m2_delta_includes_m1_delta and m1_delta_includes_m2_delta
```

4 Problem 4

We can construct a CFG:

$S \to aSbbb|abbb$

Which we convert to a stack machine:

read	pop	push
ϵ	S	aSbbb
ϵ	S	abbb
a	a	ϵ
b	b	ϵ

$M = (\{a, b, S\}, \{a, b\}, S, \delta)$

where

1. $\delta(\epsilon, S) = \{aSbbb, abbb\}$
2. $\delta(a, a) = \{\epsilon\}$
3. $\delta(b, b) = \{\epsilon\}$

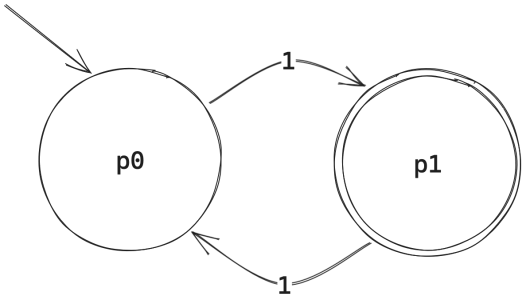
5 Problem 5

1. $S \to 0|0T|1T$
2. $T \to 1S|0S$

Is a right linear grammar, and is thus regular.

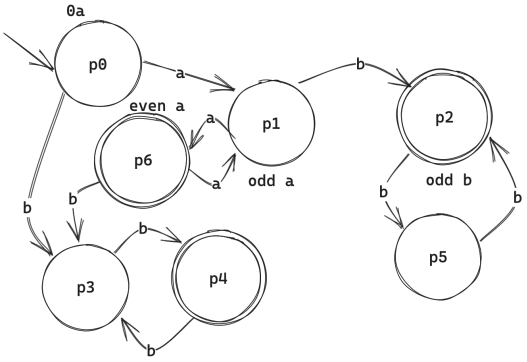
6 Problem 6

6.1 One



- $Q = \{p_0, p_1\}$
- $F = \{p_1\}$
- $\Sigma = \{1\}$
- $S = p_0$
- $\delta(p_0, 1) = p_1$
- $\delta(p_1, 1) = p_0$

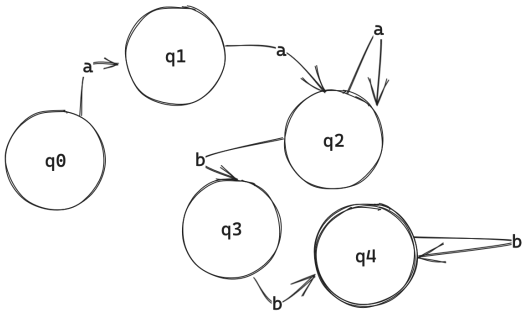
6.2 Two



- $Q = \{p_0, p_1, p_2, p_3, p_4, p_5\}$
- $F = \{p_2, p_4, p_6\}$
- $\Sigma = \{a, b\}$
- $S = p_0$
- $\delta(p_0, a) = p_1$
- $\delta(p_0, b) = p_3$
- $\delta(p_1, a) = p_6$
- $\delta(p_1, b) = p_2$
- $\delta(p_2, b) = p_5$
- $\delta(p_5, b) = p_2$

- $\delta(p_3, b) = p_4$
- $\delta(p_4, b) = p_3$
- $\delta(p_3, a) = \delta(p_4, a) = \delta(p_2, a) = \delta(p_5, a) = \emptyset$

7 Problem 7



Because the magnitude of each element in the range of δ is 1 then, intuitively, if we follow the subset construction algorithm with queue optimization we will only end up with new states identical to the ones present in the current NFA (i.e. we will visit no elements of the powerset whose magnitude is greater than one also).

Thus the DFA is:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $F = \{q_4\}$
- $\Sigma = \{a, b\}$
- $S = q_0$
- $\delta(q_0, a) = q_1$
- $\delta(p_1, a) = q_2$
- $\delta(q_2, a) = q_2$
- $\delta(q_2, b) = q_3$
- $\delta(q_3, b) = q_4$
- $\delta(p_4, b) = q_4$
- $\delta(q_0, b) = \delta(q_1, b) = \delta(q_3, a) = \delta(p_4, a) = \emptyset$