CPSC-402 Report
Compiler Construction

Scott Fitzpatrick
Chapman University

March 27, 2022

Abstract

Contents

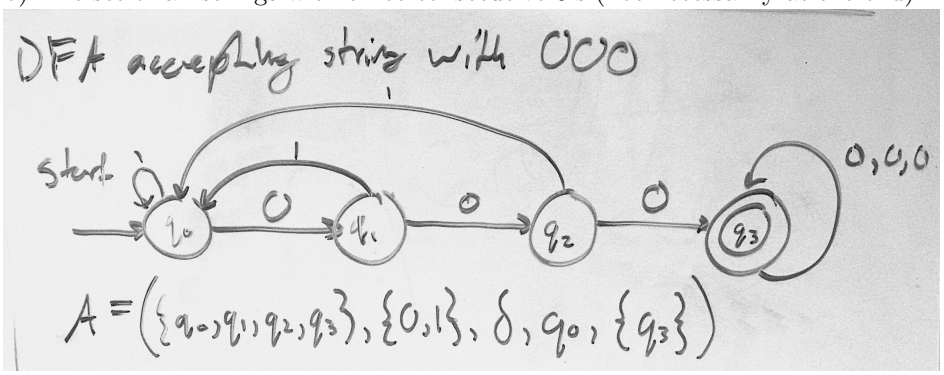
1 Introduction

2 Homework

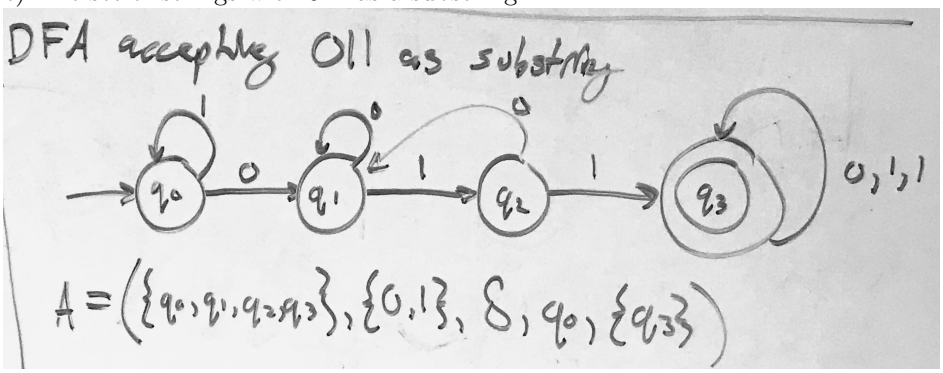
2.1 Week 1

Exercise 2.2.4: Give DFA's accepting the following languages over the alphabet $\{0, 1\}$:

b) The set of all strings with three consecutive 0's (not necessarily at the end)

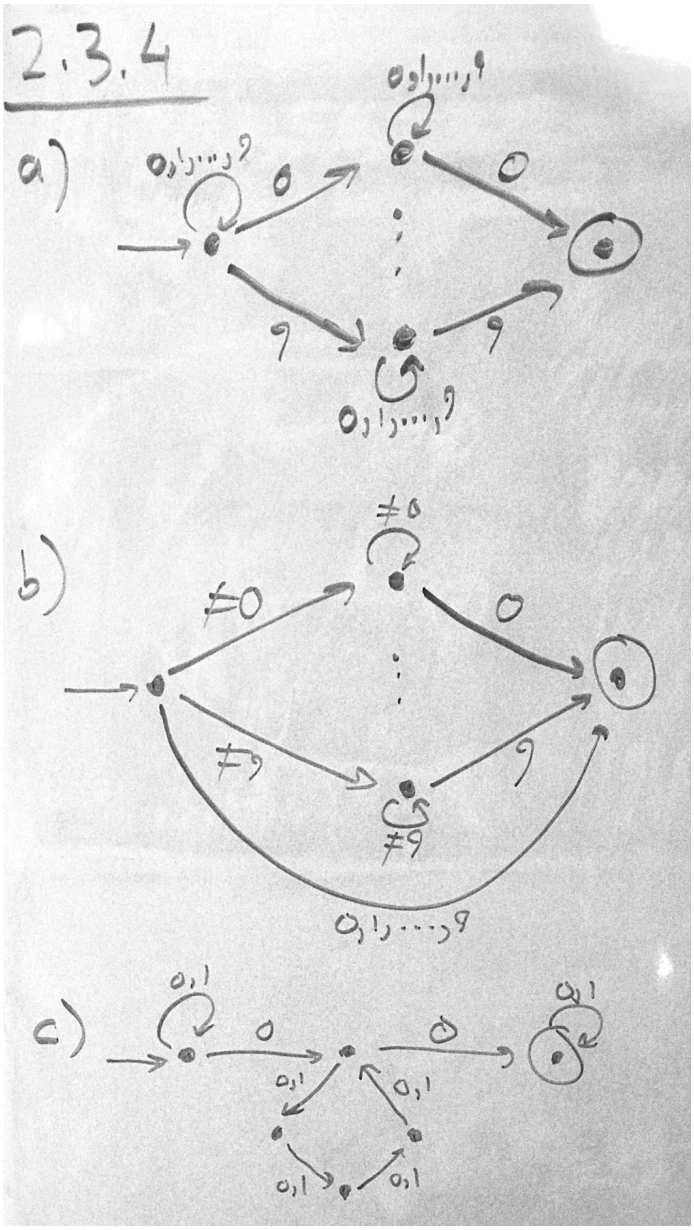


c) The set of strings with 011 as a substring

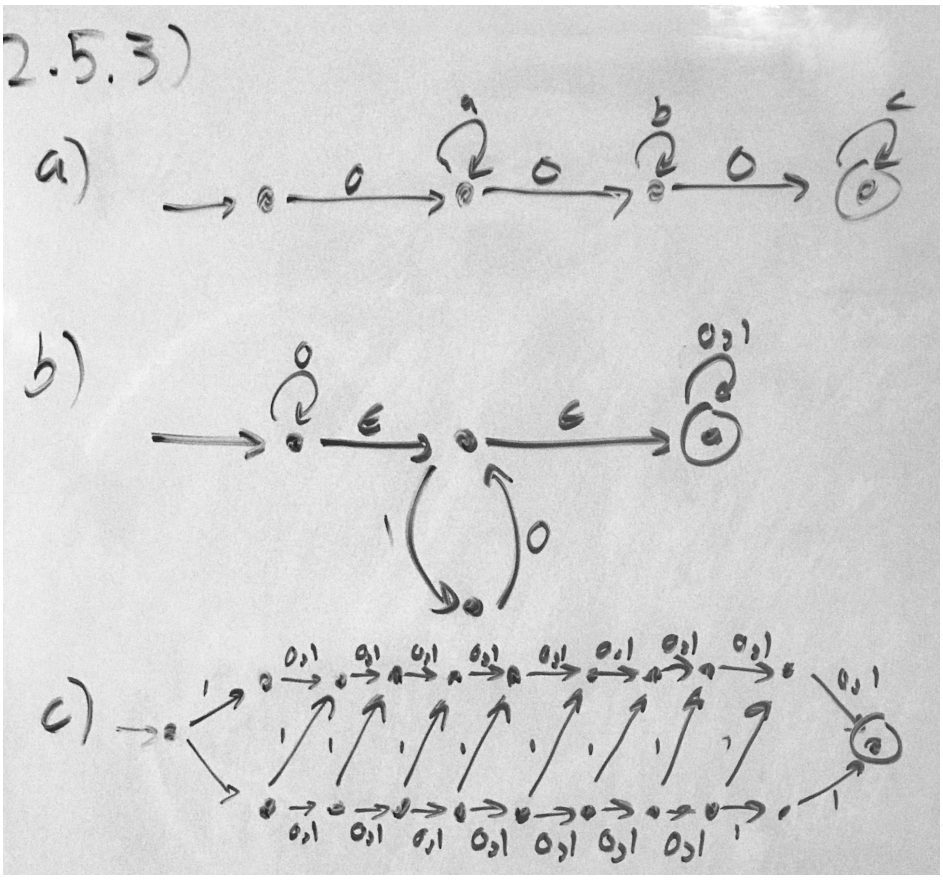


2.2 Week 2

Exercise 2.3.4: a) The set of strings over alphabet $\{0, 1, ..., 9\}$ such that the final digit has appeared before. b) The set of strings over alphabet $\{0, 1, ..., 9\}$ such that the final digit has not appeared before. c) The set of strings of 0's and 1's such that there are two 0's separated by a number of positions that is a number of 4. Note that 0 is an allowable multiple of 4.



Exercise 2.5.3: a) The set of strings consisting of zero or more a's followed by zero or more b's, followed by zero or c's. b) The set of strings that consist of either 01 repeated one or more times or 010 repeated one or more times. c) The set of strings of 0's and 1's such that at least one of the last ten positions is a 1.



2.3 Week 3

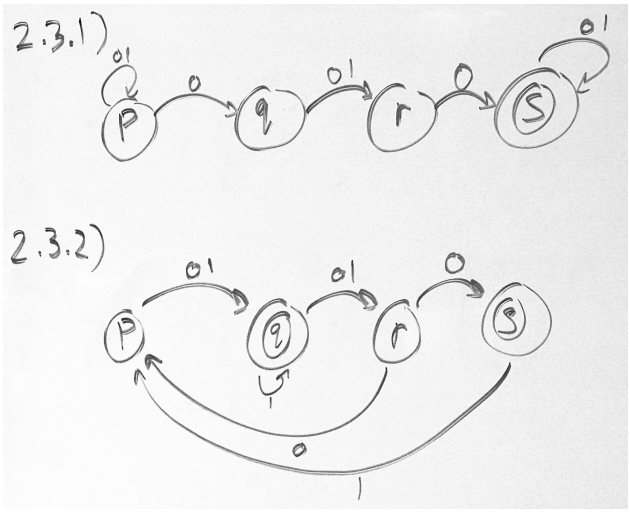
Exercise 2.3.1 : Convert to a DFA the following NFA:

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
$*s$	$\{s\}$	$\{s\}$

Exercise 2.3.2 : Convert to a DFA the following NFA:

	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	$\emptyset$	$\{p\}$

The following DFAs above can be drawn as:



In order to convert a dfa to an nfa, there can be only one final state. The initial state can remain the same, but the automata must result in one final state every time. Therefore, "final" should not return a set but instead a state.

```
-- convert an NFA to a DFA
nfa2dfa :: NFA s -> DFA [s]
nfa2dfa nfa = DFA {
-- exercise: correct the next three definitions
dfa_initial = [nfa_initial nfa],
dfa_final = let final qs = disjunction (map(nfa_final nfa) qs) in final,
dfa_delta = let
  f [] c = []
  f (q:qs) c = concat [nfa_delta nfa q c, f qs c] in f }
```

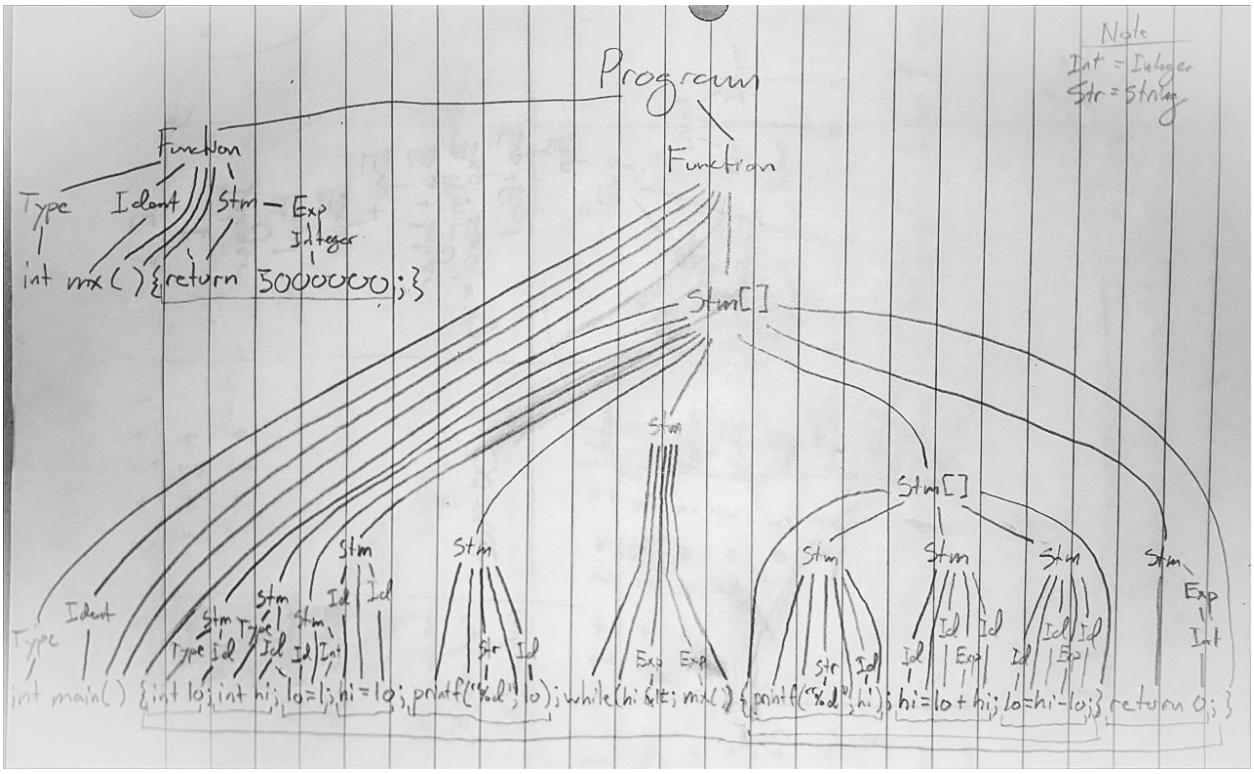
2.4 Week 4

```
// a fibonacci function showing most features of the C-- language

int mx () {
  return 5000000 ;
}

int main () {
  int lo ;
  int hi ;
  lo = 1 ;
  hi = lo ;
  printf("%d",lo) ;
  while (hi &lt; mx()) {
    printf("%d",hi) ;
    hi = lo + hi ;
    lo = hi - lo ;
  }
  return 0 ;
}
```

This is the parse tree (concrete syntax tree) for the C−− code listed above:



Here are the BNFC tokens for the same C−− code and the respective abstract syntax tree:

(To be added)

3 Project

This portion of the report will explain how the gcc compiler for the imperative programming langauge, C, works. Some example programs will be compiled and explained in detail. The target langauge will be assembly.

4 Conclusions

References