CPSC-402 Report
Compiler Construction

Anthony Walujono
Chapman University

May 18, 2022

Abstract

Short summary of purpose and content.

Contents

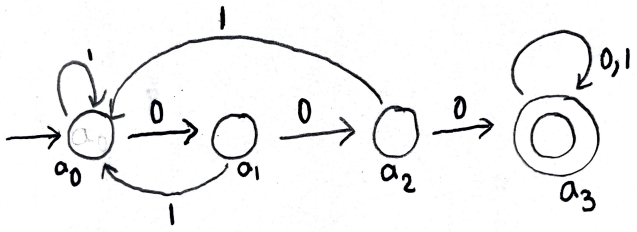
1 Introduction

This will cover everything in Compile Construction.

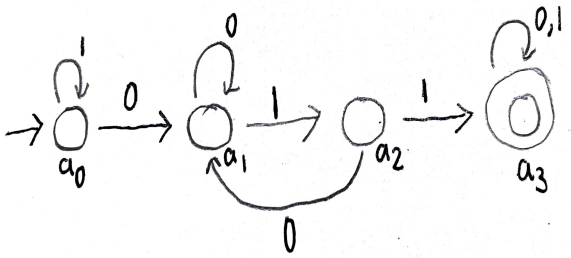
2 Homework

2.1 Week 1: Searching for Strings

2.24: Give DFA's accepting the the following languages over the alphabet {0,1}:
b) The set of all strings with three consecutive 0's (not necessarily at the end).

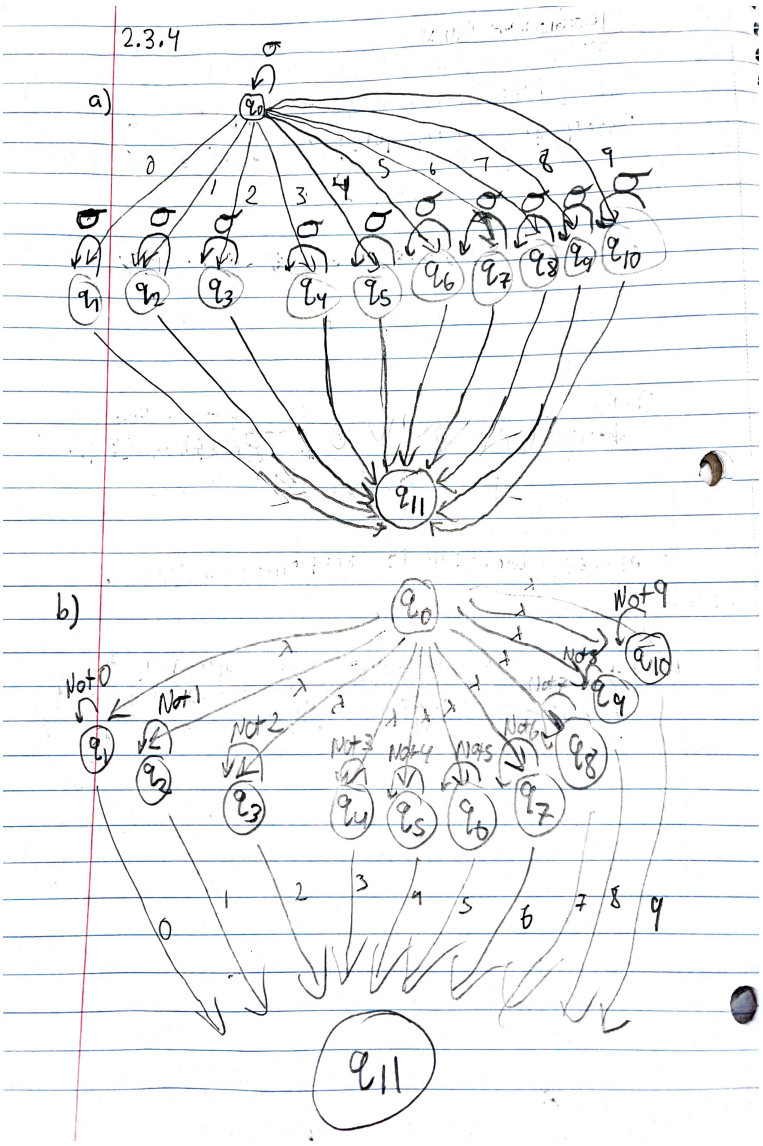


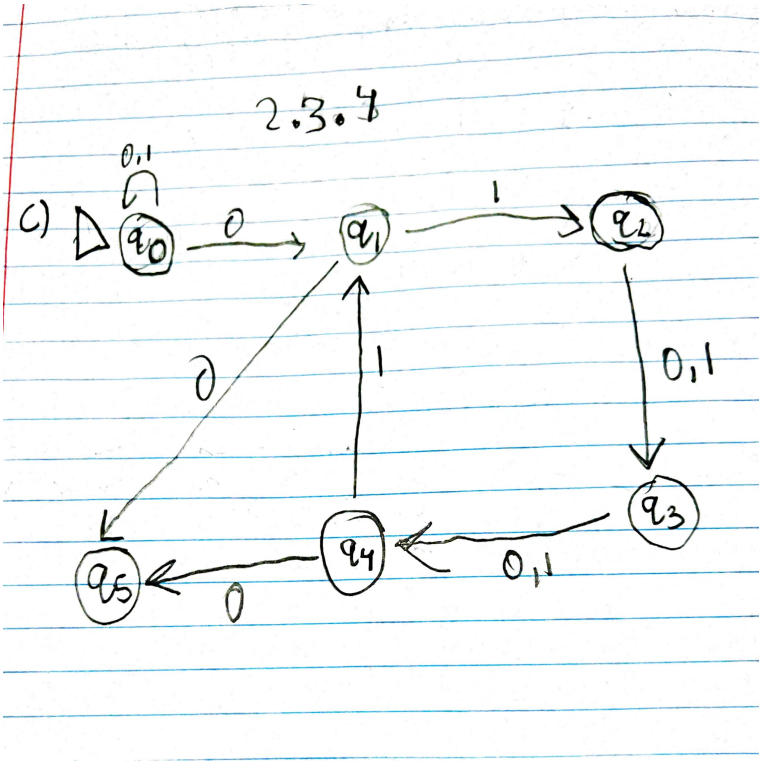
c) The set of strings with 011 as a substring.



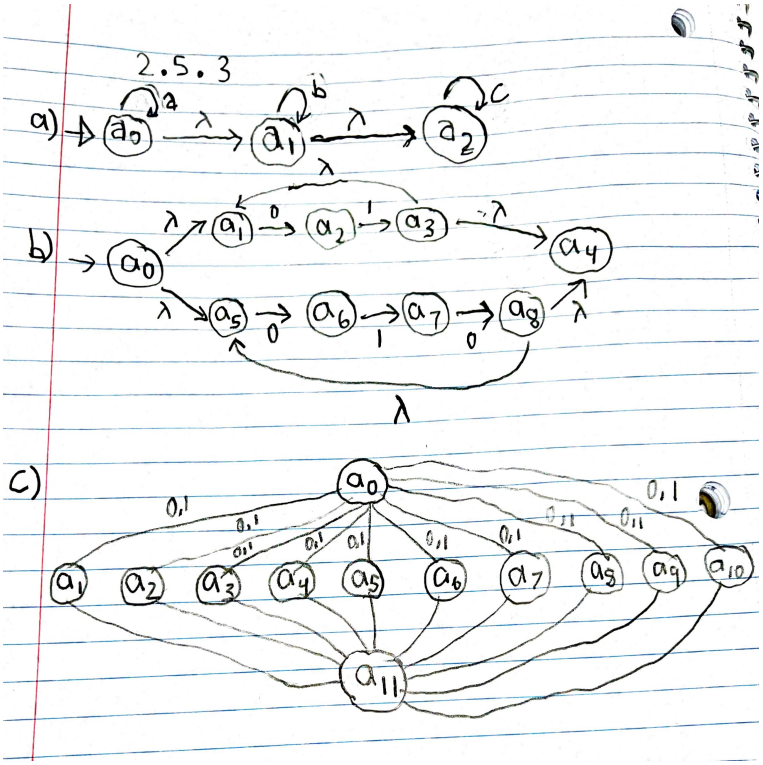
2.2 Week 2: Regular Expression and NFA

2.3.4: Give nondeterministic finite automata to accept the following languages. Try to take advantage of nondeterminism as much as possible.


2.3.4
a)
b)
Not 0
Not 1
Not 2
Not 3
Not 4
Not 5
Not 6
Not 7
Not 8
Not 9
λ
0
1
2
3
4
5
6
7
8
9

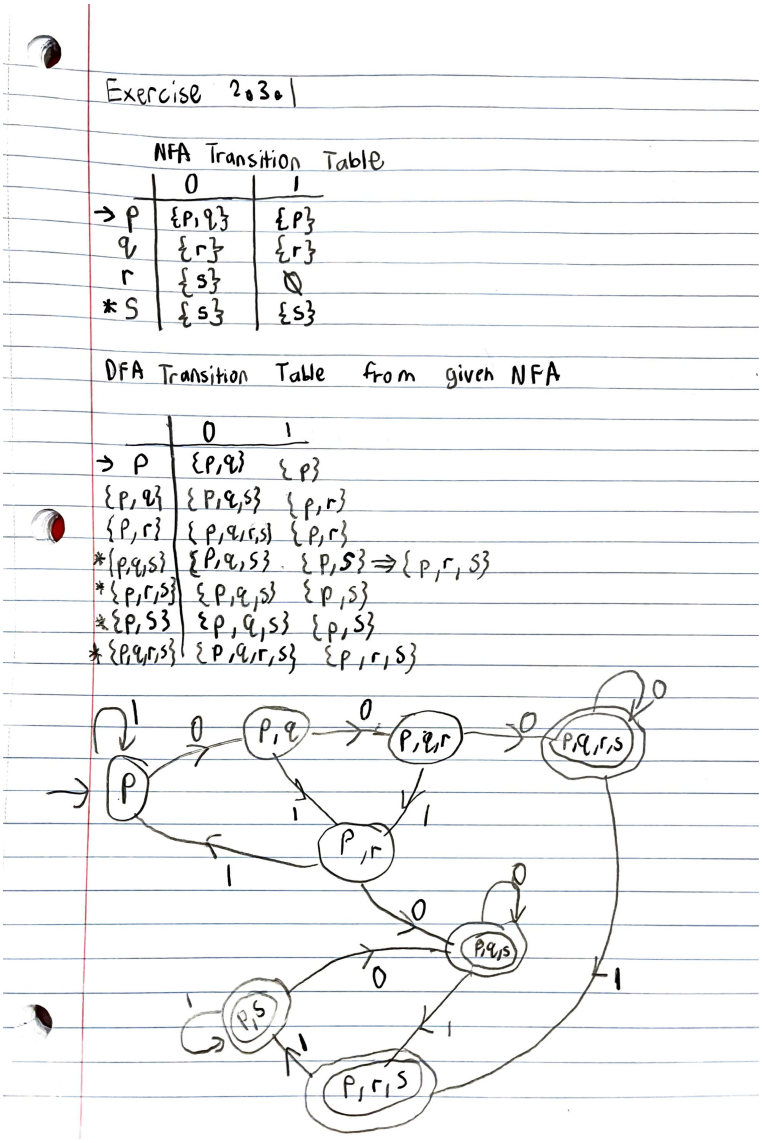


2.5.3: Design ϵ-NFA's for the following languages. Try to use ϵ transitions to simplify your design.



2.3 Week 3: Convert NFAs to DFAs by Hand

2.3.1: Convert to a DFA the following NFA



Exercise 2.3.1

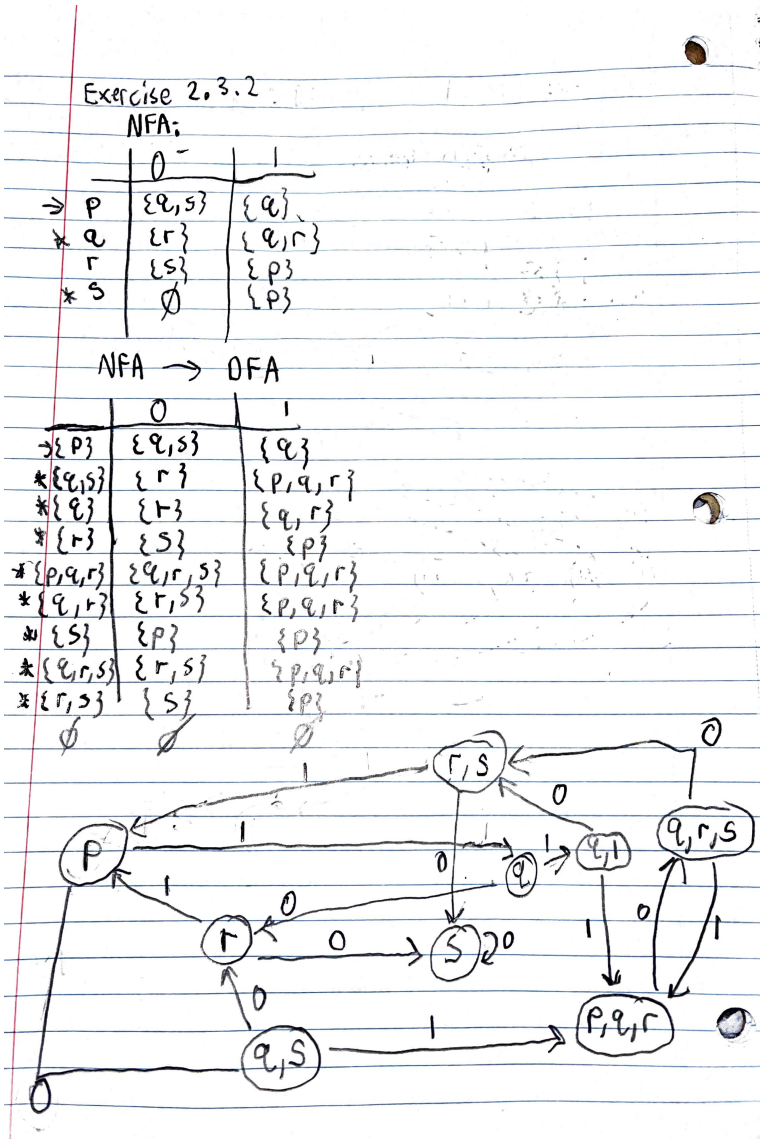
NFA Transition Table

	0	1
→ p	{p, q}	{p}
q	{r}	{r}
r	{s}	∅
* s	{s}	{s}

DFA Transition Table from given NFA

	0	1
→ p	{p, q}	{p}
{p, q}	{p, q, s}	{p, r}
{p, r}	{p, q, r, s}	{p, r}
*{p, q, s}	{p, q, s}	{p, s} ⇒ {p, r, s}
*{p, r, s}	{p, q, s}	{p, s}
*{p, s}	{p, q, s}	{p, s}
*{p, q, r, s}	{p, q, r, s}	{p, r, s}

2.3.2: Convert to a DFA the following NFA



2.4 Week 4: Introduction to Parsing

Question 1: Write out the parse tree (=concrete syntax tree) for the complete fibonacci program. Think about a question on this for the lecture.

```
using system;

public class fibonacci
{
   public static in Fib(int n)
   {
      if (n <= 1)
      {
         return n;
      }
      else
```

```
        {
           return Fib(n -1) + Fib(n - 2)
        }
     }
  }
```

Question 2: Write out the abstract syntax tree for the complete fibonacci program.

```
bnfc -d -m fibonacci.cf
make
```

3 Project

For this project, I plan to explain a compiler. The language that I would use is C++, and the compiler that I would use is g++. A code that I plan to use is recfile.cpp, which is shown below. My target language would be JVM.

```
#include <iostream>
double fahrenheitToCelsius(double fahrenheit){
    double celsius;

    celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
    return celsius;
}

int main(){
    double fahrenheit;

    std::cout << "Enter temperature in fahrenheit (in degrees) ";
    std::cin >> fahrenheit;
    std::cout << "Temperature in Celsius (in degrees) = "
              << fahrenheitToCelsius(fahrenheit) << std::endl;
}
```

Here is output using x86-64 gcc 12.1

```
fahrenheitToCelsius(double):
        push    rbp
        mov     rbp, rsp
        movsd   QWORD PTR [rbp-24], xmm0
        movsd   xmm0, QWORD PTR [rbp-24]
        movsd   xmm2, QWORD PTR .LC0[rip]
        movapd  xmm1, xmm0
        subsd   xmm1, xmm2
        movsd   xmm0, QWORD PTR .LC1[rip]
        mulsd   xmm0, xmm1
        movsd   xmm1, QWORD PTR .LC2[rip]
        divsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-8], xmm0
        movsd   xmm0, QWORD PTR [rbp-8]
        movq    rax, xmm0
        movq    xmm0, rax
        pop     rbp
        ret
```

```
.LC3:
        .string "Enter temperature in fahrenheit (in degrees) "
.LC4:
        .string "Temperature in Celsius (in degrees) = "
main:
        push   rbp
        mov    rbp, rsp
        push   rbx
        sub    rsp, 24
        mov    esi, OFFSET FLAT:.LC3
        mov    edi, OFFSET FLAT:_ZSt4cout
        call   std::basic_ostream<char, std::char_traits<char> >& std::operator<<
            <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&, char
            const*)
        lea    rax, [rbp-24]
        mov    rsi, rax
        mov    edi, OFFSET FLAT:_ZSt3cin
        call   std::basic_istream<char, std::char_traits<char> >::operator>>(double&)
        mov    esi, OFFSET FLAT:.LC4
        mov    edi, OFFSET FLAT:_ZSt4cout
        call   std::basic_ostream<char, std::char_traits<char> >& std::operator<<
            <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&, char
            const*)
        mov    rbx, rax
        mov    rax, QWORD PTR [rbp-24]
        movq   xmm0, rax
        call   fahrenheitToCelsius(double)
        movq   rax, xmm0
        movq   xmm0, rax
        mov    rdi, rbx
        call   std::basic_ostream<char, std::char_traits<char> >::operator<<(double)
        mov    esi, OFFSET FLAT:_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
        mov    rdi, rax
        call   std::basic_ostream<char, std::char_traits<char>
            >::operator<<(std::basic_ostream<char, std::char_traits<char> >&
            (*)(std::basic_ostream<char, std::char_traits<char> >&))
        mov    eax, 0
        mov    rbx, QWORD PTR [rbp-8]
        leave
        ret
__static_initialization_and_destruction_0(int, int):
        push   rbp
        mov    rbp, rsp
        sub    rsp, 16
        mov    DWORD PTR [rbp-4], edi
        mov    DWORD PTR [rbp-8], esi
        cmp    DWORD PTR [rbp-4], 1
        jne    .L7
        cmp    DWORD PTR [rbp-8], 65535
        jne    .L7
        mov    edi, OFFSET FLAT:_ZStL8__ioinit
        call   std::ios_base::Init::Init() [complete object constructor]
        mov    edx, OFFSET FLAT:__dso_handle
        mov    esi, OFFSET FLAT:_ZStL8__ioinit
        mov    edi, OFFSET FLAT:_ZNSt8ios_base4InitD1Ev
        call   __cxa_atexit
```

```
.L7:
        nop
        leave
        ret
_GLOBAL__sub_I_fahrenheitToCelsius(double):
        push    rbp
        mov     rbp, rsp
        mov     esi, 65535
        mov     edi, 1
        call    __static_initialization_and_destruction_0(int, int)
        pop     rbp
        ret
.LC0:
        .long   0
        .long   1077936128
.LC1:
        .long   0
        .long   1075052544
.LC2:
        .long   0
        .long   1075970048
```

References

[HMU] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Introduction to automata theory, languages, and computation, 3rd Edition. Pearson international edition, Addison-Wesley 2007