CS 480 Syllabus and Portfolio Winter 2019

Bryan Muller

February 28, 2019

Portfolio

Course Tracker

You are required to track your progress in the course using this table.

Note: Currently, you see full credit for week one's work. (✓ means yes. Blank means no.) Update the table for week 2 and all subsequent weeks each class day and week during the semester.

Week	CRU	PFP	CDL	SAQ	PAQ	CDL	PPL
1	✓	✓	✓	✓	✓	✓	100%
2	✓	✓	✓	✓	✓	✓	100%
3	✓	✓	✓	✓	✓	✓	85%
4	✓	✓	✓	✓	✓	✓	100%
5	✓	✓	✓	✓	✓	✓	100%
6	✓	✓	✓	✓	✓	✓	100%
7	✓	✓	✓	✓	✓	✓	100%
8	✓	✓	✓	✓	✓	✓	100%
9							
10							
11							
12							
13							

This is an honest and true record of my work for this course.

Signature: _____

Grade Claims

On the week indicated, bring this updated document to my office and make your claim.

Claim Week	Grade Claim	Instructor Grade	Adjusted Grade
5	A-		
9	A		
13 - 14			

Evidences

Fill in your evidences here each week to build your portfolio. The number of pieces of evidence are determined by you. However, the more you have the better off you will be.

Week 5

Chapter 7 Exercises [100%]

DONE Exercise 7.5: NFA to DFA

- 1.

```
NFA
I    : 0|1   -> I
I    : 0|''  -> S0
S0   : 1     -> S01
S01  : 0     -> S010
S010 : 1     -> F
F    : 0|1   -> F


strings = ["{0:b}".format(i).zfill(4) for i in range(0, 20)]
accepts = ["accepts" if accepts_nfa(nfa7_5, s) else "rejects" for s in strings]
list(zip(strings, accepts))
```

- 2. A. Define language described in 5.2.1 in formal terms: L = {w : $\in$ Σ^* : $\forall$ x $\in$ {p $\in$ parts(w): len(p) = 3 } : count(x, 1) = 2 }

  Any word w in {0,1}* where for all partitions of w with length 3 have exactly two 1's.

  Negated: L = {w: $\in$ Σ^*: $\exists$ x $\in$ {p $\in$ parts(w): len(p) = 3} : count(x, 1) != 2 }

  Any word w wherin exists a partition x of length three which does not have exactly two 1's

  So yes, this is a correct negation. B.

```
complement_of_blocks_of_3 = md2mc('''
NFA
I : 0 -> I
I : 1 -> I
I : 0 -> S0
I : 1 -> S1
S0 : 0 -> S00
S0 : 1 -> S01
S1 : 0 -> S10
```

```
S01 : 0 -> F
S10 : 0 -> F
S00 : 0 -> F
S00 : 1 -> F
F : 0 -> F
F : 1 -> F
''')
dotObj_nfa(complement_of_blocks_of_3)

dotObj_dfa(min_dfa(comp_dfa(min_dfa(nfa2dfa(complement_of_blocks_of_3)))))
```

C.

```
nums = []
it = 1
while len(nums) < 20:
  for i in itertools.product([0,1],repeat=it):
    nums.append(i)
  it += 1
values = []
for each in nums:
  word = ""
  for i in each:
    word += str(i)
  values.append(word)
for each in values:
    print("String: ", each, " Accepted NFA: ", accepts_nfa(complement_of_blocks_of
```

- 3

  Worked on it with Daniel, Matt, Seth. Couldn't figure it out.

DONE Exercise 7.6.1: Brzozoski's DFA minimization

- 1. Beginning DFA

```
bloated_dfa = md2dc('''
DFA
IS1 : a -> FS2
IS1 : b -> FS3
```

```
FS2 : a -> S4
FS2 : b -> S5
FS3 : a -> S5
FS3 : b -> S4
S4 : a | b -> FS6
S5 : a | b -> FS6
FS6 : a | b -> FS6
''')
```

Reverse turning it into an NFA

```
rev_bloated_nfa = md2mc('''
NFA
IS6 : a | b -> IS6
IS6 : a | b -> S4
IS6 : a | b -> S5
S4  : a -> IS2
S4  : b -> IS3
S5  : a -> IS3
S5  : b -> IS2
IS2 : a -> FS1
IS3 : b -> FS1
''')
```

Turn NFA into DFA

```
rev_bloated_dfa = md2mc('''
DFA
IS0 : a | b -> FS1
FS1 : a | b -> S2
S2  : a | b -> FS3
FS3 : a | b -> FS3
''')
```

Reverse reversed DFA

```
min_nfa = md2mc('''
NFA
IS1 : a | b -> IS1
```

```
IS1 : a | b -> S2
S2 : a | b -> IS3
IS3 : a | b -> FS4
''')
```

Convert back to dfa

```
min_dfa = md2mc('''
DFA
IS0 : a | b -> FS1
FS1 : a | b -> S2
S2 : a | b -> FS3
FS3 : a | b -> FS3
''')
```

- 3.

```
blimp = md2mc('''
DFA
I1 : a -> F2
I1 : b -> F3
F2 : a -> S8
F2 : b -> S5
F3 : a -> S7
F3 : b -> S4
S4 : a | b -> F6
S5 : a | b -> F6
F6 : a | b -> F6
S7 : a | b -> F6
S8 : a -> F6
S8 : b -> F9
F9 : a -> F9
F9 : b -> F6
''')
min1 = min_dfa(blimp)
min2 = min_dfa_brz(blimp)
iso_dfa(min1, min2)
```

True

Chapter 8 Exercises [100%]

DONE Exercise 8.2: NFA Operations

- 1. 001100100 001000101
- 2.

```
re_8_5_nfa = md2mc('''
NFA
I1 : '' -> St1
I1 : '' -> St2
St1 : '' -> I1
St2 : a -> St3
St3 : '' -> I1
IF2 : '' -> St4
IF2 : '' -> St5
St4 : c -> St6
St6 : '' -> St7
St7 : d -> St8
St8 : '' -> IF2
St5 : b -> St9
St9 : '' -> IF2
I1 : '' -> IF2
''')
dotObj_nfa(re_8_5_nfa)
```

- 3.

```
re_8_5_nfa = re2nfa("(''+a)*(b+cd)*")
dotObj_nfa(re_8_5_nfa)

re_8_5_nfa_hand = md2mc('''
NFA
I1 : '' -> St1
I1 : '' -> St2
St1 : '' -> I1
St2 : a -> St3
St3 : '' -> I1
IF2 : '' -> St4
IF2 : '' -> St5
```

```
St4 : c -> St6
St6 : '' -> St7
St7 : d -> St8
St8 : '' -> IF2
St5 : b -> St9
St9 : '' -> IF2
I1 : '' -> IF2
''')
dotObj_nfa(re_8_5_nfa_hand)

iso_dfa(nfa2dfa(re_8_5_nfa), nfa2dfa(re_8_5_nfa_hand))
```

True!

DONE Exercise 8.8: Sylvester's Formula

- 1. No. Any linear combination of 3 and 6 will always have to be a multiple of three. This means there are infinitely many natural numbers which cannot be expressed by 3 and 6.
- 2.

  The requirement that the greatest common divisor (GCD) equal 1 is necessary in order for the Frobenius number to exist. If the GCD were not 1, every integer that is not a multiple of the GCD would be inexpressible as a linear, let alone conical, combination of the set, and therefore there would not be a largest such number. For example, if you had two types of coins valued at 4 cents and 6 cents, the GCD would equal 2, and there would be no way to combine any number of such coins to produce a sum which was an odd number. On the other hand, whenever the GCD equals 1, the set of integers that cannot be expressed as a conical combination of { a1, a2, ..., an } is bounded according to Schur's theorem, and therefore the Frobenius number exists.

  – Wiki

Exercise 8.8.5: Postage Stamp

- 1. a. p,q = 5, 11 F(p,q) = (5*11) - 5 - 11 = 55 - 5 - 11 = 39 b. p,q = 5,1 F(p,q) = 39 p,q,r = 5,7,11 F(p,q,r) = 13
- 2.