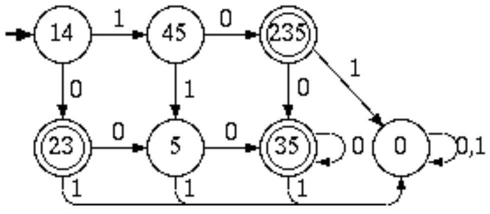
ARSDIGITA VNIVERSITY

Month 8: Theory of Computation

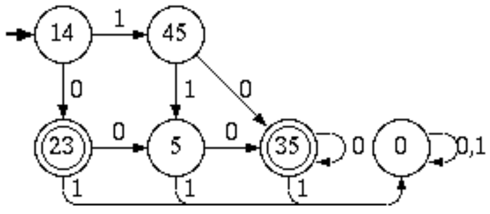
Problem Set 2 Solutions - Mike Allen and Dimitri Kountourogiannis

1. Minimizing DFAs

a. Convert to a DFA



b. Convert to a minimal DFA



c. Conver to a regular expression:
```
0 + (00 + 1 + 11)00*
```

d. Convert to a regular grammar using the NFA:
```
A -> 0 | 0B | 1C | 1D
B -> 0D
C -> 1D
D -> 0 | 0D
```

Using the DFA:
```
S -> 0A | 1B
A -> 0C | e
B -> 0D | 1C
C -> 0D
```

Using the regular expression:
```
S -> 0 | 110A | 000A | 10A
A -> 0 | 0A | e
```

2. Regular or Not?

a. **NOT**. (1.17b) {www | w is {a,b}*}

Assume that the language is regular. Let p be the pumping length, and choose s to be th string $0^p10^p10^p1$. Now we try to break it up into s=xyz. Since |xy| <= p and |y|>0, y can contain 0's. When we pump the string even just once we get $xy^2z = 0^{p+|y|}10^p10^p1$, and t not of the form www, since |y| > 0. This contradicts the pumping lemma, so the languag not regular.

b. **NOT**. (1.23c) $\{0^m1^n$ | m is not equal to n}

We know that

```
{0ⁿ1ⁿ | n >= 0}  = {0ᵐ1ⁿ | m,n >= 0}^{0*1*}ᶜ,
```

where we are using ^ to denote intersection and ^{c} to denote complement. The proof is b contradiction. If $\{0^m1^n$ | m is not equal to n} really were regular then $\{0^n1^n$ | n >= 0} wou also be regular because 0*1* is regular and because of the closure properties of regular Therefore it can't be regular

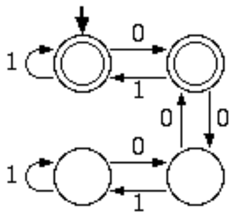
There is a direct way to prove it as well: If p is the pumping length and we take the strinc

c. **NOT**. (1.23a) $\{0^m1^n0^m \mid m,n >= 0\}$

Assume that the language is regular. Let p be the pumping length, and choose s to be th string 0^p10^p. Now we try to break it up into s=xyz. Since |xy| <= p, y can only have zeros Now $xy^jz = 0^{p+ (j-1) |y|}10^p$, and since |y|>0 the number of 0's on the left and right sides o xy^jz will not be the same for any j>1 so xy^jz will not be in the language, contradicting the pumping lemma. Therefore $\{0^m1^n0^m \mid m,n >= 0\}$ is not regular

d. **REGULAR**. The set of strings that have an even number of double zeros in them.

This can be decided by the DFA below.



e. **REGULAR**. The set of all strings of the form xwx^R where x and w are strings over the alph {0,1}.

This description is somewhat misleading. Since w can represent any string and x can be (1, this is the same as all strings which begin and end with the same character. This is eas seen to be a regular language.

f. **NOT**. The set of all strings over the alphabet {0} whose length is n! for some n>0.

Assume that the set is regular. Let p be the pumping length. Without loss of generality w can assume that p is at least 2. (We can always increase the pumping length. We only do because some of our calculations don't work for p = 1) Then, according to the pumping lemma, we can break the string $s=0^{p!}$ into s=xyz where y has positive length and |xy| <= Then $s=xy^2z = 0^{p! + |y|}$ must also be in the language, so p!+|y| must also be a factorial. E (p+1)!-p! = (p)p! > p >= |y| so it follows that p! + |y| < (p+1)!, and therefore p! + |y| is not factorial. This is a contradiction so the language cannot be regular.

g. **NOT**. The set of strings over the alphabet {0} of the form 0^n where n is not prime.

To prove this language is not regular, we instead examine the complement because the : of regular languages is closed under complement. Assume that the set is regular. Let p b the pumping lenght of the language. Then, according to the pumping lemma, we break the string $s=0^p$ into s=xyz where y has positive length. Then, $s=xy^iz = 0^{p + (i-1) |y|}$ must also be the set for any i. In particular let i = p+1. Then $xy^{p+1}z=0^{p+p|y|}$ must be in the set so p + p p(1 +|y|) must be prime. Thus we have a contradiction and the set cannot be regular.

3. Find the Flaw in the Proof

The flaw in the proof is the following: if p is the pumping length of 0*1* and $s = 0^p1^p$ is decomposed into xyz in the usual manner then s absolutely positively *can* be pumped since xy equal to $0^{p+(j-1)|y|}1^p$ which will still be in 0*1*.

languages may satisfy the pumping lemma's conditions without contradicting it. In this partic
case of

{$a^i b^j c^k$ | i,j,k >= 0,if i=1 then j=k},

if s is any string in the given language then the decomposition s = xyz with x = the empty strin
= to the first character of s, z = the rest of s, will always work with the pumping lemma. The
resulting xy^jz will always be in the language.

5. Decision Algorithms

   a. Contains all strings of the form 0*1*.

      Given a DFA, M, we can determine if it accepts all strings of the form 0*1* by taking its
      complement and intersecting it with the DFA which accepts 0*1*, and then minimizing tl
      resulting DFA. If the result is the machine that accepts nothing, then accept M, otherwise
      reject M.

   b. Is co-finite.

      Given a DFA, M, we can determine if it is cofinite by taking its complement and then chec
      to see whether the result has any cycles from which an accept state is reachable. If it doe
      not, then accept M, otherwise reject M.

   c. Has at least one string w that has 111 as a substring.

      Given a DFA, M, we can determine if it accepts at least one string of the form 111 by
      intersecting it with another DFA which accepts (0+1)*111(0+1)* and minimizing the resul
      DFA. If the result accepts anything, then accept M, otherwise reject M.

6. Regular Linear Grammars

   a. A regular grammar to generate the set of strings that are evenly divisible by 5

```
A₀ -> 0A₀ | 1A₁ | e
A₁ -> 0A₂ | 1A₃
A₂ -> 0A₄ | 1A₀
A₃ -> 0A₁ | 1A₂
A₄ -> 0A₃ | 1A₄
```

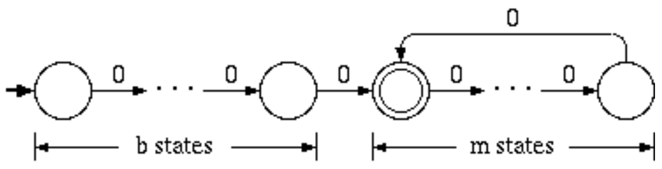
   b. A finite automaton can be converted into a right-linear grammar as follows:
      1. Reverse all the arrows in the finite automaton.
      2. Create a start state with epsilon transistions pointing towards each of the old accep
         states.
      3. Change the old start state to be the accept state.
      4. Convert the automaton as you would to get a left-linear grammar, except that term
         are placed on the right side. (ie A->Ba instead of A->aB)

7. Linear Grammars and Palindromes

   a. Since we know that left and right linear grammars describe only regular languages and t
      set of palindromes over the language {0,1} is not regular, neither grammar can describe
      palindrome.

8. Single Symbol Regular Languages

    a. Languages of the form 0^{mx+b} are accepted by DFAs of the form below, so they must be regular.



    b. Unions of machines from the set described by (a) are regular, though not of the form 0^m

    c. **Extra Credit:** This is actually fairly simple using the DFA characterization of regular sets. Since there can only be one transiton from each state the DFA is completely specified by giving

        i. The number of states, n, in the machine.
        ii. An n-bit number, f, which tells us whether each state is final or not.
        iii. A periodicity number, m, which has some value 0 <= m <= n

    The DFA is constructed by making state 1 the initial state, setting the j-th state to accept and only if the j-th bit of the number f is a 1, adding a transition on 0 from state j to state for 1<= j <= j-1 and finally if m is not zero, adding a transition from state n to state n-m+1 The language constructed from this DFA will finite if m=0, and otherwise will be eventual periodic, of period m.

    It pretty easy to see from the above description that, up to a finite set, every regular language over 0 is of the form

    $(0^{b\_1} + 0^{b\_2} + \ldots + 0^{b\_k})(0^m)^*$,

    for some choice of m, where {b_1, b_2, ..., b_k} is some subset of {0,1,2,...,m-1}

9. (extra credit) Simulating NFAs

10. (triple extra credit) Minimizing DFAs