HW2 Grp2

① Let $L=\{w\in\{a,b,\dots z,0,1\}^* \mid w$ has an odd number of 1's$\}$
Let $L'=\{v\in\{a,o,n,e\}^* \mid$ the word "one" appears in v an odd number of times and the letter e only appears in these one's$\}$.
Give a homomorphism h from the language of L to that of L'. Next make a regular expression for L' by using the construction from class to obtain a regular expression for h(L).

We map h(1) = one. This ensures if w has 2K+1 many 1's for some K≥0, then h(w) will have at least 2K+1 many one's. We want to complete our map h so that h(w) has exactly 2K+1 many one's. We want to ensure that strings like oneon, none, aonaone, all can be hit by our map.
To do this we take
$h(a)=a,\ h(b)=a,\ \dots,\ h(m)=a,\ h(n)=n,\ h(o)=o,$
$h(p)=a,\ \dots\ h(z)=a,\ h(0)=a.$
In particular $h(e)=a$. We can do this last because a string $v\in L'$ contain e only in occurences of one. The circled cases ensure our homomorphism is onto.

Next we make a regular expression for L.
Let $R=(a\cup b\cup c\cup\dots\cup z\cup 0)$. Then a regular expression for L is

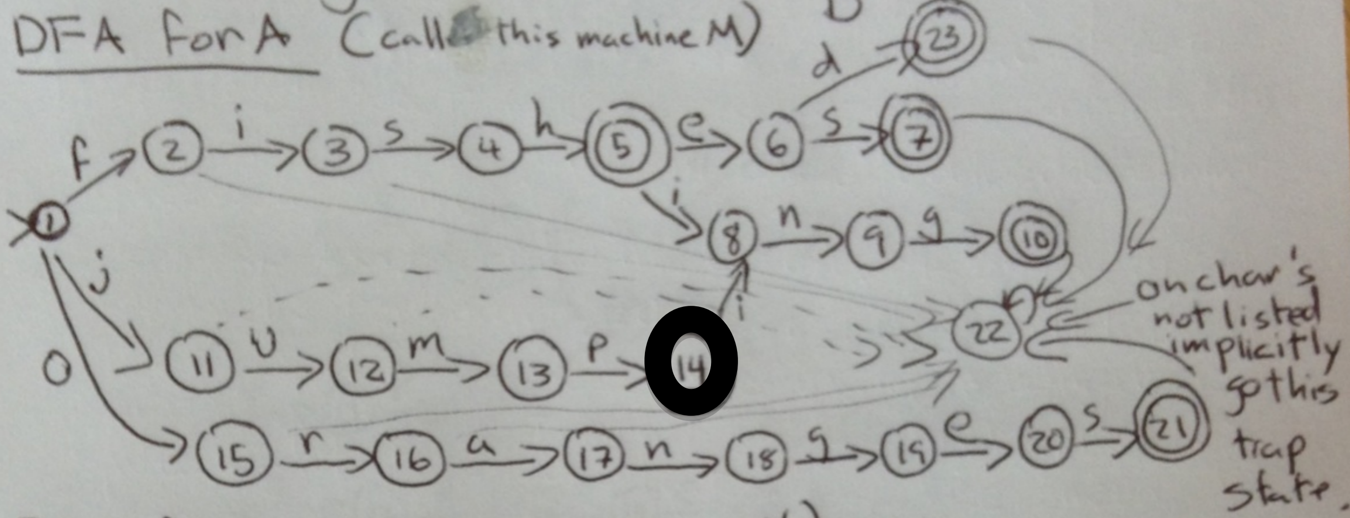
$$\underbrace{R^*1}_{\text{at least one } 1}\underbrace{(R^*1R^*1R^*)^*}_{\text{thereafter, an even number of } 1\text{'s}}.$$

Call this R_L.

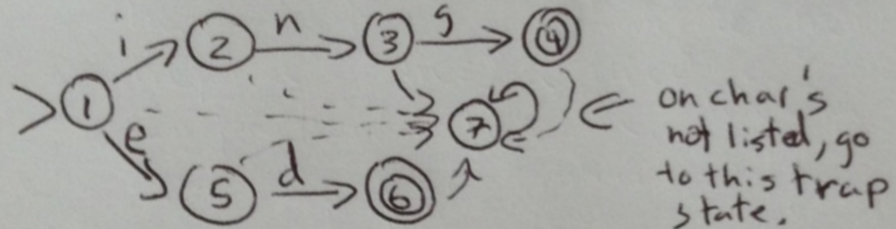
The construction from class of a regular expression for h(L) from one for L preceded by induction on the complexity of subexpressions of R_L.
1st Build regex's for subexpressions of R_L of complexity 1. These subexpressions are a, b, c, ... z, 0, 1. We apply h to get the corresponding h(L) subexpression so we have a, ..., a, o, n, a ..., a, one.
Next we compute regex's for subexpressions of complexity 2 & so on. Doing this, we get the regex $h(R)=(a\cup\dots\cup a\cup o\cup n\cup a\dots a)$
The final regex is $(h(R))^*one((h(R))^*one(h(R))^*one(h(R))^*)^*$.

2 Let $A = \{fished, fish, fishes, fishing, jumping, oranges\}$ and $B = \{ing, ed\}$. Give DFAs for each of these languages. Then using Ginsberg and Spanier's construction give an automata for $\frac{A}{B}$.

DFA for A (call this machine M)

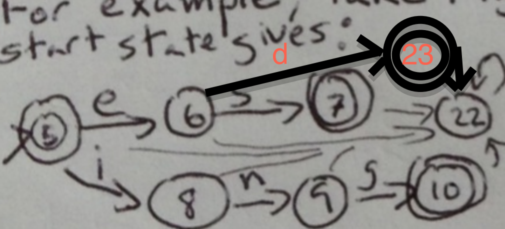


DFA for B (call this machine M')

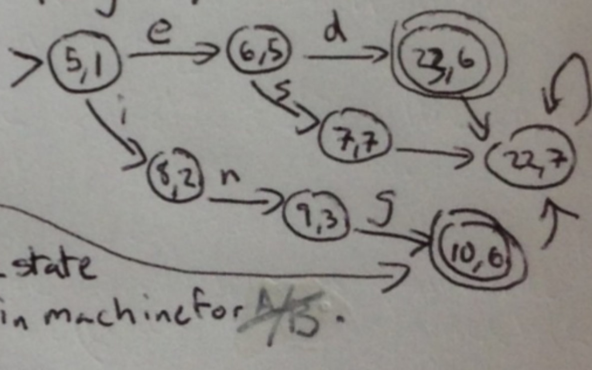


To compute $\frac{A}{B}$, Ginsberg Spanier say for each state i in M make a new Machine M_i where the start state is i and otherwise the machine is the same as M. Then using the Cartesian product construction build a machine for $L(M_i) \cap L(M')$ and via the reachability algorithm check if this is nonempty.

For example, take M_5. Restricting to states reachable from the start state gives:



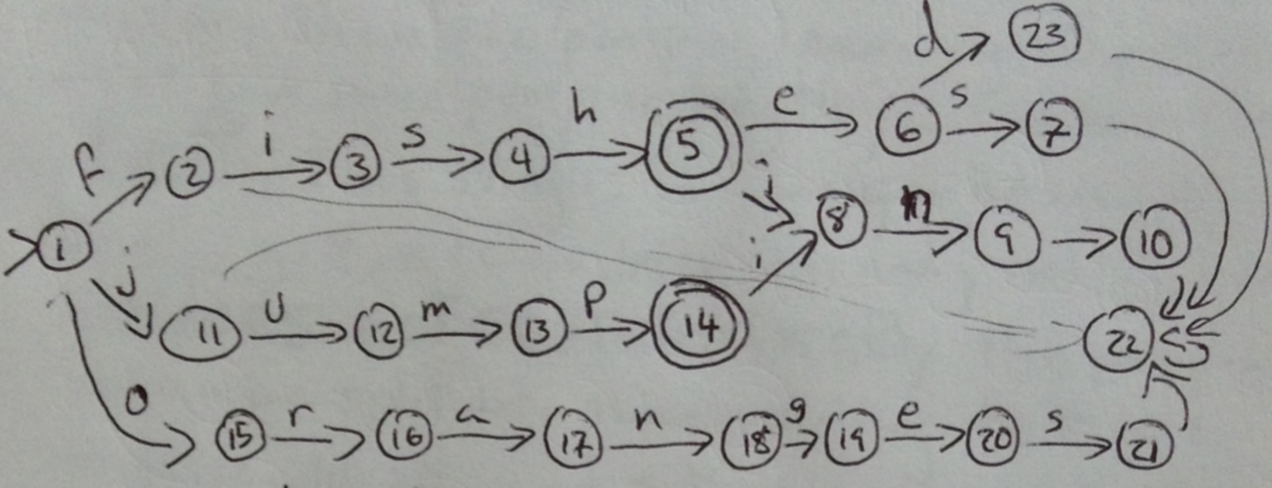
Doing a Cartesian Product with M' keeping only reachable states:



An accept state is reachable from start state in $M_5 \times M'$ so the state 5 should be accepting in machine for $\frac{A}{B}$. See next page...

(2 cont'd).

Ginsberg and Spanier then say machine for $\frac{A}{B}$ is M where final states are the states i such that $M_i \times M'$ has a reachable accept state. This gives:



transitions not listed go to trap state 22.

(3) Let L be the language consisting of strings $w = xyz$ over the alphabet $\{a, b, c\}$ where $|x| = |z|$ and y uses only symbols not in x or z. Show using the pumping lemma that L is not regular.

We will prove this by contradiction. Suppose L were regular. Then the pumping lemma can be applied and L must have some pumping length p. Consider the string $w = a^p c^p$. This string is in L as can be seen by choosing $x = a^p$, $y = \varepsilon$, $z = c^p$ as $|x| = p = |z|$ and y uses no characters so does not contain a or c. By the pumping lemma w can be split into three substring tuv s.t. $|tu| \leq p$, $|u| > 0$, and s.t. tu^iv is in L for all $i \geq 0$. Since $|tu| \leq p$, we know tu is a substring of x, so $t = a^k$, $u = a^j$ where $k + j \leq p$. Since $|u| > 0$, $j > 0$.

The pumping lemma says $tu^0v = tv = a^k c^p$ is in L. Suppose we try to split tv into xyz as per def of string in L. Since the 1st letter of this string is 'a', x contains the letter 'a'. Similarly, since the last letter contains a 'c' z contains a 'c'. Therefore, y cannot contain either a or c, and can be at most a string of b's.

But in $a^k c^p$, $|a^k| < |c^p|$, and so this string does meet the definition to be in L. Therefore, we have a contradiction. Therefore, our original assumption that L was regular must be false.