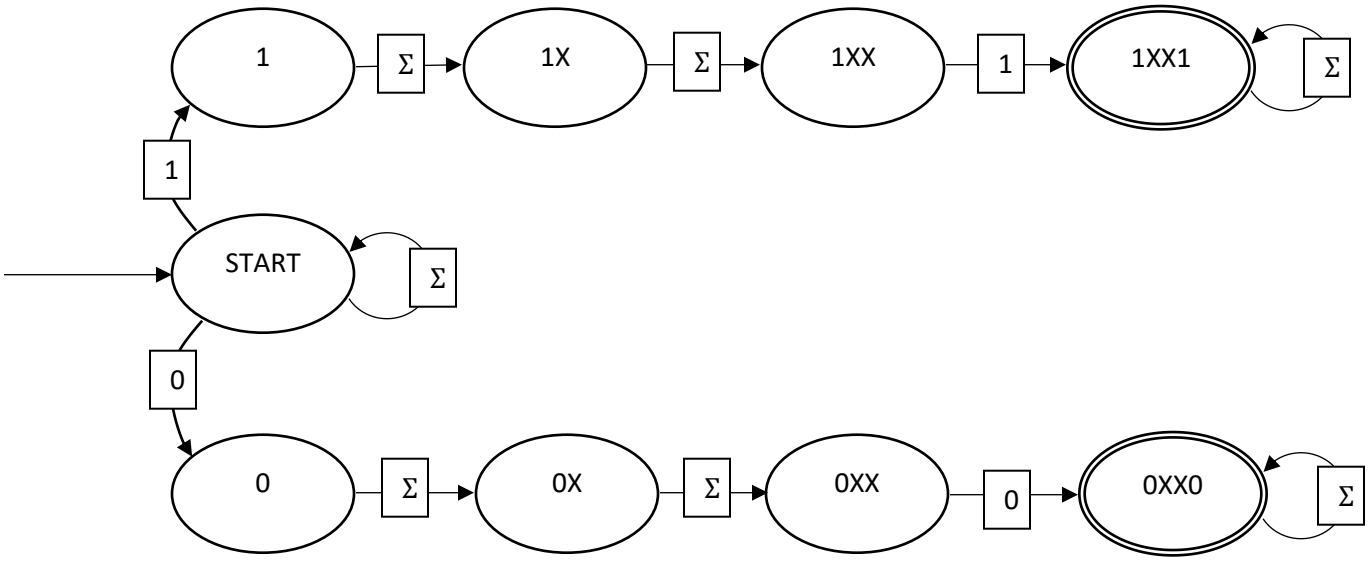
Brock Boehler
bvb2@illinois.edu
ECE 374 B Spring 2019
Homework 0

Problem 1

(a)



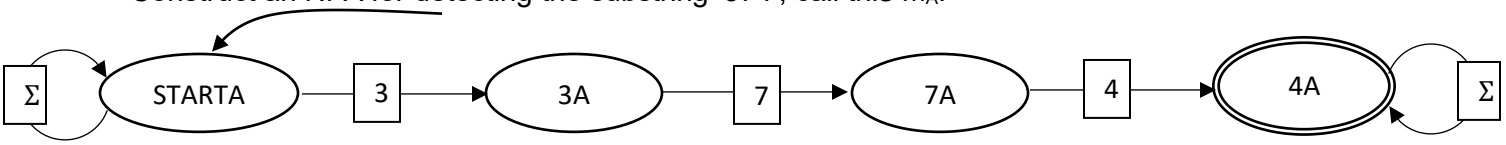
Explanation of States:

- START – We have not seen anything or we have not encountered anything relating to the substring 1xx1 or 0xx0
- 1 – we have seen the initial 1 in the string 1xx1
- 1X – We have seen 1 followed by a random character
- 1XX – We have seen 1 followed by two random characters
- 1XX1 – we have seen the substring where there are two 1's three spaces apart
- 0 - we have seen the initial 0 in the string 0xx0
- 0X - we have seen 0 followed by a random character
- 0XX – we have seen 0 followed by two random characters
- 0XX0 – we have seen the substring with two 0's three spaces apart

This NFA works by checking for two strings at the same time. It checks for one substring containing two 1's three spaces apart and one substring containing two 0's three spaces apart. See state explanation for further information.

(b)

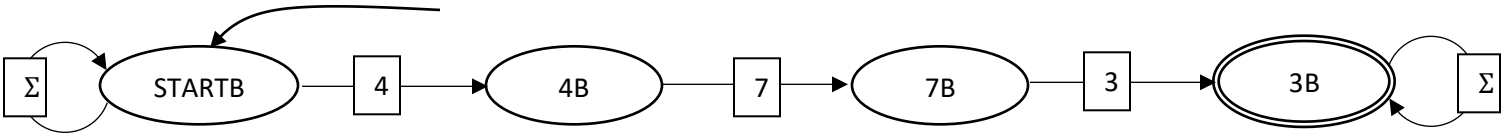
Construct an NFA for detecting the substring '374', call this M_A:



Description of States:

- STARTA – We have not read anything or we have not read the string '374' yet.
- 3A – We have read the initial 3 in '374'
- 7A – We have read the '37' part of '374'
- 4A – We have found the string '374'

Construct a second NDFA for detecting substring '473', call this M_B:



Description of States:

- STARTB – We have not read anything or we have not read the substring '473' yet.
- 4B – We have read the initial '4' in '473'
- 7B – We have read the '47' part of '473'
- 3B – We have read the string '473'

Having these two NFA's, M_A and M_B, we can then use product construction to develop an NFA that detects both '374' and '473', with the following parameters (note $M_A = (\Sigma, Q_A, S_A, A_A, \delta_A)$ and $M_B = (\Sigma, Q_B, S_B, A_B, \delta_B)$):

$$\Sigma = \{0, 1, \dots, 9\}$$

$$Q = Q_A \times Q_B$$

$$S = (STARTA, STARTB)$$

$$A = (4A, 3B)$$

$$\delta: (Q_A \times Q_B) \times \Sigma \rightarrow (Q_A \times Q_B)$$

$$P = (p_1, p_2) \in (Q_A \times Q_B); A \in \Sigma;$$

$$\delta(\mathrm{P}, \mathrm{A}) = \delta_A(p_1, A) \times \delta_B(p_2, A)$$

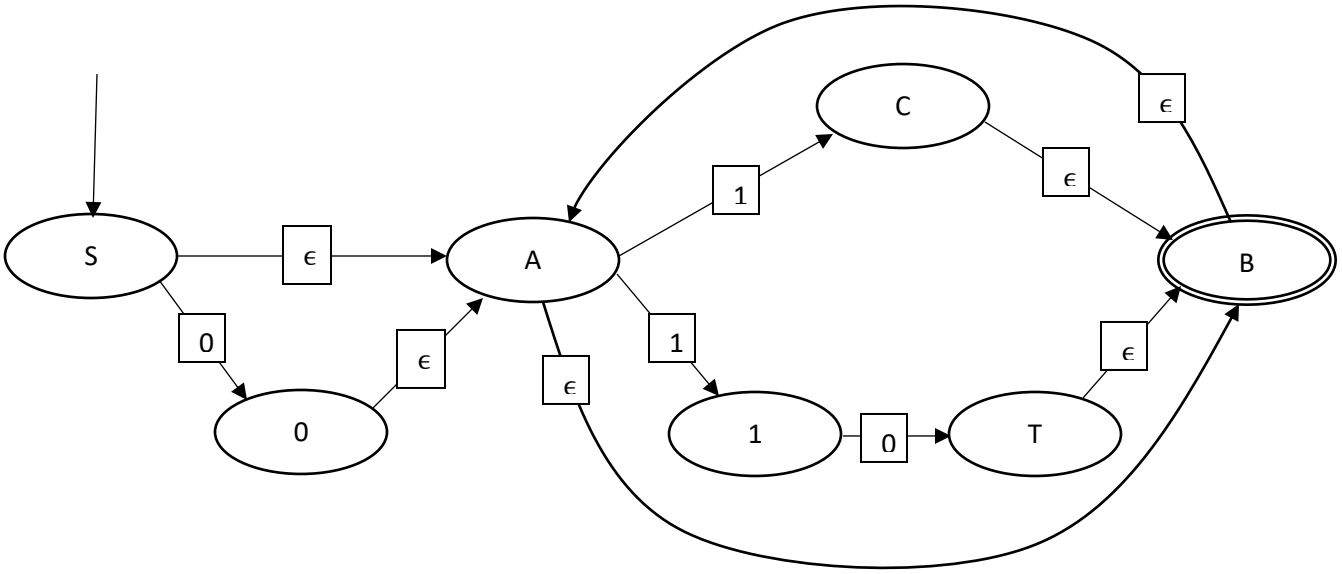
This NFA is the product construction of the two NFA's shown above. Whenever it reaches a state where both NFA's are satisfied and in the accepting state, we know that we have found a string that contains both substrings '374' and '472'.

Problem 2

(a)

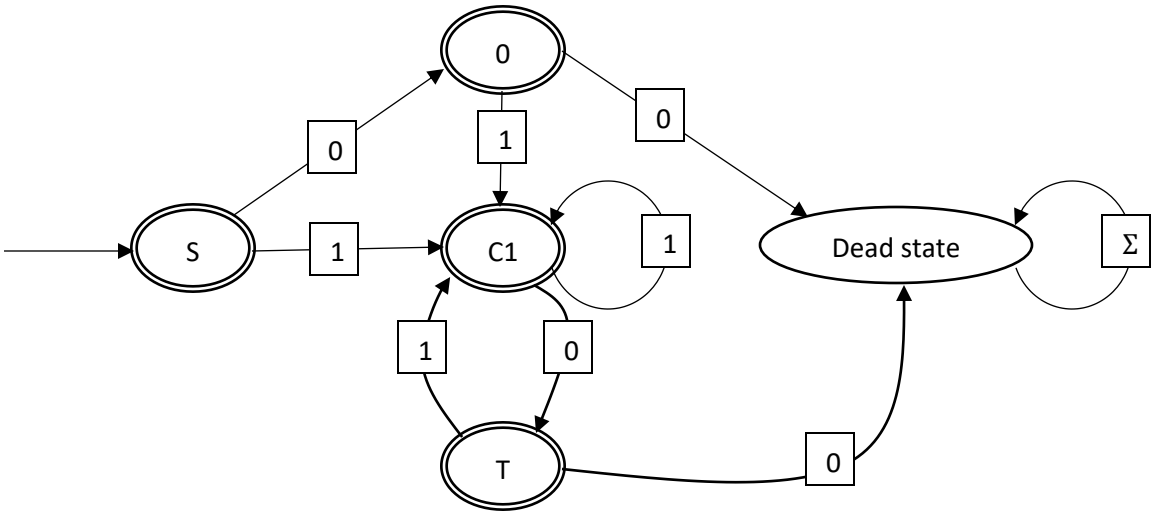
$(\boldsymbol{\epsilon} + \mathbf{0})(\mathbf{1} + \mathbf{10})^*$ is the language in which every word has no two consecutive zeros

- Construct an NFA corresponding to the regular expression using Thompson's algorithm

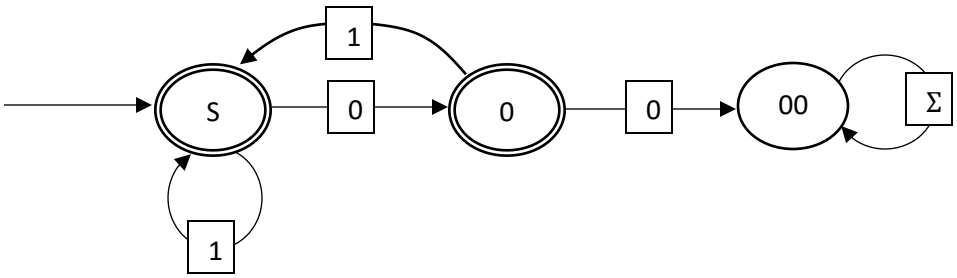


- Use the incremental sub set construction to convert the NFA to a DFA

q	ϵ-reach(q)	$q \in A$?	$\delta(q,0)$	$\delta(q,1)$
S	SAB	X	0	C1
0	0AB	X	Dead state	C1
C1	CAB1	X	T	C1
T	TAB	X	Dead state	C1
Dead state	Dead state		Dead state	Dead state



- Create another DFA with fewer states to recognize the language



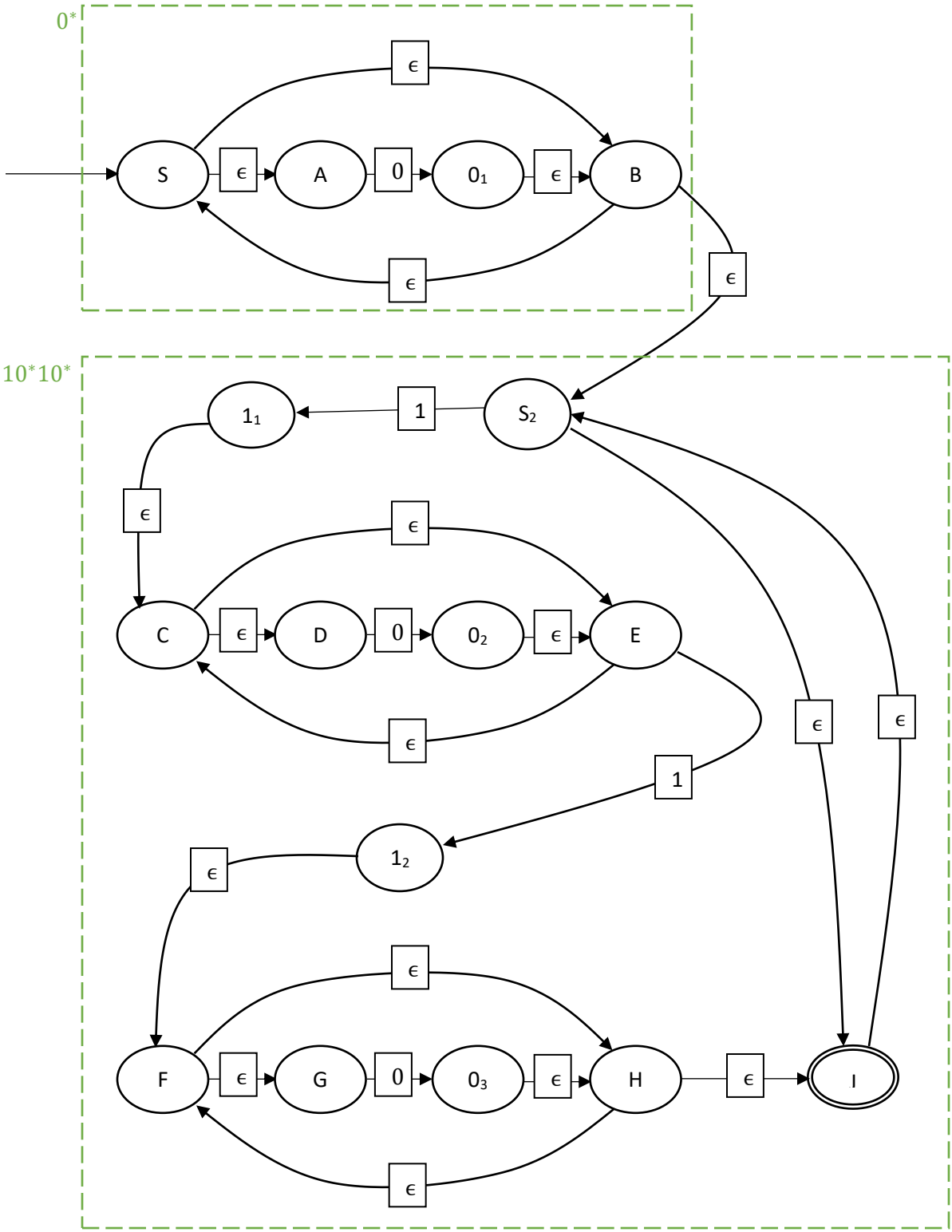
Explanation of States:

- S – We have not read anything or we have just read a 1
- 0 – We have read a single 0
- 00 – We have read two 00's sequentially and have entered a dead state

(b)

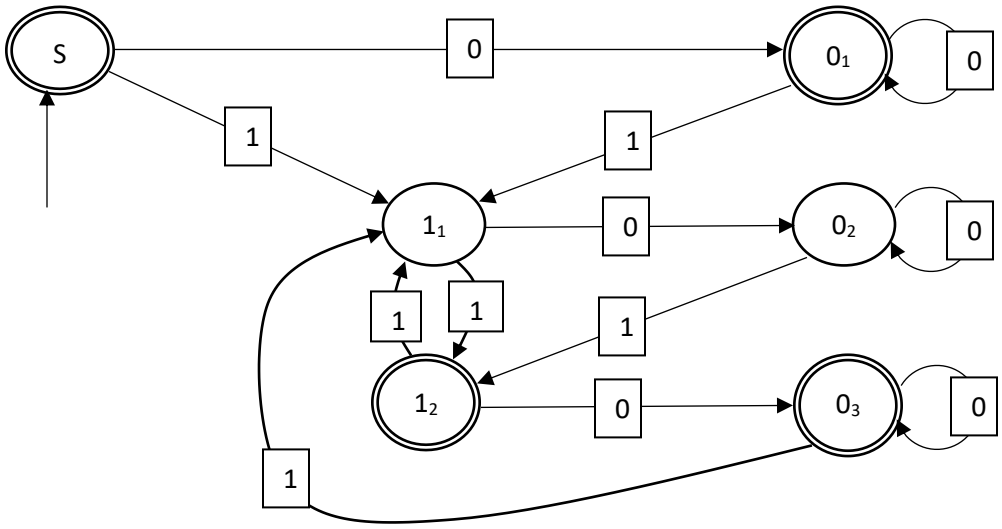
$0^*(10^*10^*)^*$ is the language of all words containing an even number of 1's

- Construct an NFA corresponding to the regular expression using Thompson's algorithm

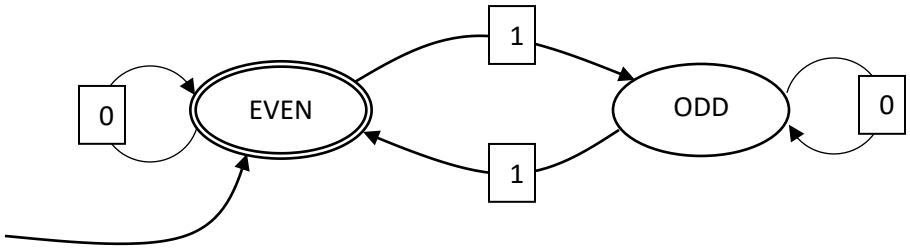


- Use the incremental subset construction to convert the NFA to a DFA

q	ϵ-reach(q)	$q \in A$?	$\delta(q,0)$	$\delta(q,1)$
S	$SABS_2I$	X	0_1	1_1
0_1	0_1SABS_2I	X	0_1	1_1
1_1	1_1CDE		0_2	1_2
0_2	0_2CDE		0_2	1_2
1_2	$FGHIS_2$	X	0_3	1_1
0_3	0_3FGHI	X	0_3	1_1



- Create another DFA with fewer states to recognize the language



Explanation of States:

- EVEN – We have read an even number of one's
- ODD – We have read an odd number of one's

Problem 3

Part A skipped

(b) Let $M = (\Sigma, Q, s, A, \delta)$ be the DFA that accepts language L(M). We can define a new NFA $M_R = (\Sigma, Q_R, s_R, A_R, \delta_R)$ with ϵ transitions that accepts the language $L(N) = \{x \in \Sigma^* \mid xx^R \in L(M)\}$

$$Q_R = (Q \times Q) \cup s_R$$

s_r is an explicit state in Q_R

$$A_R = \{(h, h) | h \in Q\}$$

$$\delta_R(s_R, \epsilon) = \{s, a \mid a \in A \text{ and } h \in Q\}$$

$$\delta_R\big((x, z), a\big) = \{(\delta(x, a),\ \delta^{-1}(z, a)\}$$

Where δ^{-1} is the inverse delta function, defined as $\delta^{-1}(x, a) = \bigcup_{q \in Q}\{\delta(q, a) = x\}$.

M_R works by simulating two copies of M, with one simulation starting from the default starting state, and one simulation starting at the accepting state and working backwards. When both machines end up in the same state h, we then know that xx^R has been accepted by L(M).

- State (x, z) means:
  - The left copy (the DFA working from start to the middle) is in state x
  - The right copy (the DFA working from the accepting state of M to the middle) is in state z
- M_R accepts the string IFF x moves the DFA M to a halfway state h, and the string x^R moves the DFA from state h to an accepting state

Proof:

- 0. Suppose we have a given DFA $M = (\Sigma, Q, s, A, \delta)$ and have the NFA mentioned above $N = (\Sigma, Q_R, s_R, A_R, \delta_R)$. We will prove that the NFA accepts all strings such that $L(N) = \{x \in \Sigma^* \mid x\mathrm{x}^\mathrm{R} \in L(M)\}$
- 1. To check if $xx^R \in L(M)$, let us break up xx^R into its two substrings: x and x^R. We can define the "walk" of a string in a DFA or NFA as the list of states it visits until it reaches the end of the string. Let W denote the walk of xx^R, and let h be an arbitrary state the walk visits in the middle of W. We further define h as the state that the DFA reaches upon completion of processing x, and the state where x^R begins and reaches an accepting state in.
- 2. Lemma: If x^R starts at h and ends at an arbitrary accepting state s, then if we start at state s and travel backwards using the inverse delta function using characters in x for our transition values, we will end up with a path that then ends at state h. (i.e. using our traditional delta function on x^R will move us from h to s, but using the inverse delta function on x will move us from s to h) Proof by induction:
  - Define the inverse delta-star function as follows:
    - Use inverse delta function as defined before proof

- $\delta^*_{\text{inv}}(q,w) = \begin{cases} \text{q; w } = \epsilon \\ \bigcup_{r\in\delta^{-1}(q,a)}\left(\delta^*_{\text{inv}}(r,x)\right); w = ax \end{cases}$
    - 2. 1Inductive hypothesis: Suppose for some string x and w, and some character a, with w = (ax), that for all strings with length less than or equal to |x|, if $\delta^*(t,x^R) = v$ then $\delta^*_{\text{inv}}(\text{v},\text{x})$ will result in a sequence of paths, with at least one leading back to, and ending in, the original state t. (t and v are both states)
    - 2.2 Base case: $x = \epsilon$. Because we have no string, our delta-star and delta-star-inverse functions start and end on the same state (t = v), thus $\delta^*_{\text{inv}}(\text{v},\text{x}) = \text{t}$, and because $\epsilon^R = \epsilon$, $\delta^*(t,x^R) = \text{v}$
    - 2.3 Inductive Step: because w = ax, we will first find the state that $\delta^*(\text{t},w^R)$ ends in:

$$\delta^*(\text{t},w^R)$$

$$= \delta^*(\text{t},(\text{ax})^R), \text{ def. of w}$$

$$= \delta^*(t,x^Ra), \text{ def of R function}$$

$$= \delta^*(\delta(t,x^R),\text{a}), \text{ def of delta-star}$$

$$= \delta^*(v,a), \text{ inductive hypothesis}$$

Define: $\delta^*(v,a) = z$

Now show that if we start at state $\delta^*(v,a)$, we will end up with a set of paths, with at least one of which ends in t:

$$\delta^*_{inv}(\delta^*(v,a),w)$$

$$\delta^*_{inv}(\delta^*(v,a),\text{ax}), \text{ def. of w}$$

$$= \cup_{r\in\delta^{-1}(\text{z},a)}\left(\delta^*_{\text{inv}}(r,x)\right), \text{ def of } \delta^*_{\text{inv}}$$

$$\Longrightarrow \text{t} \in \delta^*_{inv}(\delta^*(v,a),w), \text{ inductive hypothesis}$$

We have now shown that if $\delta^*(t,x^R) = v$, then $\text{t} \in \delta^*_{\text{inv}}(\text{v},\text{x})$

- 3. (prove strings in language are accepted by NFA) The NFA we construct to recognize the desired language must then be constructed of two components: one component to check if x goes from the start state in our DFA to a mid point h, and another to check if x^R goes from a start state to h. To do this, we make a product construction of 2 NFA's, with the end state being the cross where both sub-NFA's have reached point h
- 4. The NFA that checks that x goes from start to h is simple, name if NFA A, just re-use the original DFA and keep track of it's progress in one half
- 4. The NFA that checks if y goes from accepting state to h is more complex, name it NFA B. Take the original DFA, and set the starting state to an explicit new state. Have epsilon-transitions from this start state to all accepting states of the DFA, and have the delta function be the previously-defined inverse delta function.
- 5. NFA B implements the inverse delta-star-inverse function, by definition, of the original DFA. This means that if $\delta^*(h,\text{y}) = \text{h}$ then $h \in \delta^*_{inv}(a \in A,\text{y})$.

- 6. Therefore, we can plug x into both NFA A and NFA B and when both states end in the same mid-point h, we know that we have found a string that is accepted in our language.
- 7. (Prove strings not in language are not accepted by NFA) Suppose we have a string not in our language such that $yy^R \notin L(M)$. Because the original DFA would not end on a starting state, our function δ^*_{inv} and δ^* would not both end on the same state h. This is because of the deterministic nature of our original DFA. If $\delta^*(yy^R)$ does not end in an accepting state, it is *impossible* for $\delta^*(s, y)$ and $\delta^*_{inv}(a \in A, y)$ to have paths ending on the same state. Proof by contradiction:
    - Suppose $yy^R \notin L(M)$ but $\delta^*(h, \mathrm{y}) = h$ and $h \in \delta^*_{inv}(a \in A, \mathrm{y})$.
    - By our earlier proof we have shown that our NFA checks to see if y working forward from the start to the middle and y^R working from the finish to the middle meet each other at some specific state
    - If they met each other at the same state then their would be a path from the start to the accepting states of M
    - $\delta^*(s, yy^R)$ does not end in an accepting state because $yy^R \notin L(M)$
    - This is a contradiction
    - Therefore, they cannot meet in a middle state h
- 8. We have shown that for the language L, our NFA accepts all strings contained in it and does not accept strings not in L.

…I'm sure this proof will be as fun to grade as it was to write…