CSE 105 (FA20, Jones/Meyer)

Homework 7: Computational Complexity Classes

Rachel Cai [redacted]

Jiahong Long [redacted]

Andy Young [redacted]

9 December 2020

1. *True/False.*

   (a) NP is closed under intersection.

   True. Suppose A and B are problems in NP. Their intersection is verifiable by a Turing machine M which runs verifiers for A and B (both running in polynomial time by definition) either in parallel or serially, and accepts if both verifiers accept. Since this machine is a composition of machines which run in polynomial time, M itself also runs in polynomial time. It follows that we have shown the existence of a machine (and consequently, an algorithm) verifying $A \cap B$ running in polynomial time. Hence NP is closed under intersection.

   (b) If P = NP, then NP is closed under complementation.

   True. Recall that P is closed under complementation:

   Let A be a problem in P. Let M be a polynomial-time decider thereof. Construct the machine M' as:

   M': On input z:

   - Run M on z.
   - If M accepts, then reject.
   - If M rejects, then accept.

   We show that M' decides $\overline{A}$: suppose $a \in A$. Then $a \notin \overline{A}$, and because $a \in L(M)$, by construction $a \notin L(M')$ as required. Then, suppose $b \notin A$, so $b \in \overline{A}$. It follows that $b \notin L(M)$, so $b \in L(M')$ as required. Hence, M' decides $\overline{A}$, and does so in polynomial time (as it is a composition of a machine with a polynomial runtime).

   Hence, given that NP = P and P is closed under complement, we conclude that NP is also closed under complement.

   (c) Any non-empty finite language is in P.

   True. Let L be any such non-empty, finite language. Since L is finite, we can define

   $$\limsup |L| = \text{the length of the longest string in } L$$

   Any string in L is guaranteed to be at most $\limsup |L|$ characters long. Then, a decider for L only needs to read at most $\limsup |L|$ characters before making a decision on an input z since

   - if $|z| > \limsup |L|$, clearly $z \notin L$, and
   - if $|z| \leq \limsup |L|$, the number of characters we need to check is less than $\limsup |L|$.

Hence, we can construct a decider for L with a runtime bounded above by $\limsup |L|$, which therefore runs, as $n \to \infty$, in constant time. For a constant k, $O(k) \subseteq O(n^0) \subset \bigcup_i O(n^i)$. Hence, L is in P.

(d) For some language L, if $PATH$ polynomial-time-reduces to L and L polynomial-time reduces to $PATH$ then L is in P.

True. Recall that $PATH \in$ P. If L polynomial-time reduces to $PATH$, then any polynomial-time decider for $PATH$ can be used to decide L.

(e) For any non-empty language A, Σ^* polynomial-time reduces to A.

True. With the assumption that all input strings will be over Σ, let A be any non-empty language decided by M. Let Σ^* be decided by M_Σ. Clearly all strings over Σ are accepted by M_Σ. Then, we can construct a computable function which, on any input, outputs $const_a$, a string in A such that $const_a \in A$. Vacuously, this computable function runs in polynomial time (since it runs in constant time, as all input strings are assured to be in Σ^*), so we have shown that Σ^* polynomial-time reduces to A.

(f) By Rice's theorem: The language $\{\langle M \rangle \mid M \text{ is a Turing machine and } |L(M)| = 1\}$ is undecidable.

True. Denote

$$ONE_{TM} = \{\langle M \rangle \mid M \text{ is a Turing machine and } |L(M)| = 1\}$$

We will now provide a polynomial-time mapping reduction from A_{TM} to ONE_{TM}. Let

F: On input z:

- Typecheck z. If z is not of the form $< M, w >$ where M is a machine and w is a string over Σ, output $const_{rej}$, an encoding of a machine that accepts more than one string.
- Construct the machine M': On input s:
  - Run M on w.
  - If M accepts w and s is the empty string, <u>accept</u>. If s is not the empty string, <u>reject</u>.
  - If M rejects w, <u>reject</u> s.
- Output $\langle M' \rangle$.

All steps taken by F run in polynomial time: typechecking, machine construction, and outputting are all 'simple' tasks. Additionally, F is a reduction from A_{TM} to ONE_{TM}: we will show that $x \in A_{TM} \leftrightarrow F(x) \in ONE_{TM}$:

($\rightarrow$) Let $x \in A_{TM}$. Then, $x =< M, w >$, where M is a machine that accepts w. Then, $F(x) = \langle M' \rangle$, the machine constructed by F, will accept only the empty string. It immediately follows that $|L(M')| = 1$, and hence $F(x) \in ONE_{TM}$ as required.

($\leftarrow$) Let $F(x) \in ONE_{TM}$. Then, by definition, $F(x) = \langle M' \rangle$ where M' accepts just one string. By construction, if M' accepts just one string, that string can only be the empty string. For this to be the case, we must have that M accepts w. It then immediately follows that $x =< M, w > \in A_{TM}$ as required.

Hence we have shown both directions, and F is indeed a reduction as required. Because A_{TM} is undecidable, and is reduced to ONE_{TM}, we have the ordering condition

$$A_{TM} \leq_M ONE_{TM}$$

and it necessarily follows that ONE_{TM} is also undecidable.

(g) If $A \in$ P then $A \in$ co–NP.

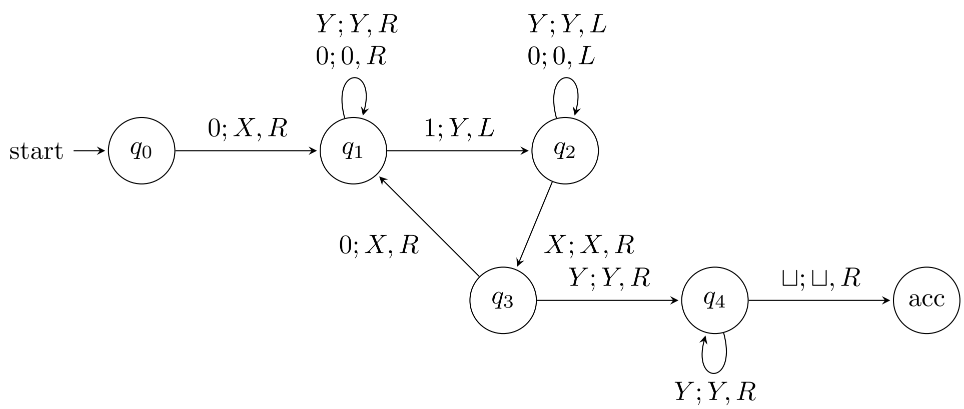
True. If $A \in$ P, then $\overline{A} \in$ P since P is closed under complementation. Additionally, since P $\subseteq$ NP, we have that $\overline{A} \in$ NP. Hence it follows by definition that $A \in$ co–NP.

(h) Suppose that $A \in$ NP–Complete and A polynomial-time-reduces to B, then $B \in$ NP–Complete.

False. Recall that NP-Complete is a subset of NP–Hard, a class of problems which are only at least as hard as the those in NP-Complete. Then, it is possible for B to be in NP–Hard $\setminus$ NP–Complete.

2. *Time complexity*

Let M be the Turing machine displayed below:



This machine decides the language $\{0^k 1^k \mid k \geq 1\}$.

(a) Consider the statement

> For all even positive integers n, the input string of length n for which M takes the most steps before halting is $0^{n/2}1^{n/2}$.

Give an argument about why the above statement is true.

We begin by conjecturing intuitively that, the statement is true because, for an input of the form $0^j 1^k$ where $j + k = n$, the machine will 'short-circuit' out after running out of either 1s or 0s, whichever happens first. Hence, to maximise runtime, we also conjecture that we must set $j = k = \frac{n}{2}$. We can concretely prove this by tracing the computation:

Note that, for any input that does not start with a 0, the machine immediately rejects after just one transition from q_0 to q_{rej}. Additionally, for any input that contains j zeroes and no ones, the machine rejects during the transition from q_1 to q_{rej}, as there is no 1 to transition to q_2, meaning $j + 1$ steps are taken.

For well-formed inputs (ie. those of the form $0^j 1^k$ and $j, k > 0$), note that we can break the computation into steps:

- Transition $q_0 \to q_1$: Cross off the first zero **(1 step)**
- Transition $q_1 \to q_2 \to q_3$: Cycle $\min(j-1, k-1)$ times: **(**$\min(j-1, k-1) \cdot (2j+1)$ **steps)**
  - Transition $q_1 \to q_1$: Skip as many zeroes and Ys as necessary to get to the first 1 **(**$j - 1$ **steps)**

  - Transition $q_1 \to q_2$: Cross off the next one with a Y, and move left **(1 step)**
  - Transition $q_2 \to q_2$: Move left until we arrive at the rightmost X **($j-1$ steps)**
  - Transition $q_2 \to q_3$: Move right to the next character **(1 step)**
  - Transition $q_3 \to q_1$: Cross off the next zero **(1 step)**
- Then:
  - If $j \le k$: Repeat the cycle once more, without transitioning back to q_1. From q_3, transition to q_4, consuming a Y. **($2j+1$ steps)** Then:
    * If $j < k$: Since j ones were crossed off, there are $k-j > 0$ remaining. Hence, transition from q_4 to itself $j-1$ times to read off the remaining Ys, and then transition once more to q_{rej}. **(j steps)**
    * If $j = k$: All the ones were crossed off. Hence, transition from q_4 to itself $j-1$ times to read off the remaining Ys, and then transition once more to q_{acc}. **(j steps)**
  - If $j \ge k$: Repeat the cycle once more. There are now no more ones left on the tape. **($2j+1$ steps)** Then:
    * If $j > k$: Transition from q_1 to itself $j-1-k$ times to read off the remaining zeroes, and then k times to read off the Ys, and then once more to read a $\sqcup$ and reject. **($1+j$ steps)**.
    * If $j = k$: Transition from q_1 to itself $j-1-k$ times to read off the remaining zeroes, and then $k = j$ times to read off the Ys, and then once more to read a $\sqcup$ and reject. **(j steps)**.

Thus, in total, for each non-degenerate input of length $n > 0$ where j, k denote the number of zeroes and ones respectively, the runtime is

	j, k	runtime
1	$j > 1,\ k = 0$	j
2	$j, k > 1,\ j \le k$	$j(2j+1)+j+1$
3	$j, k > 1,\ j \ge k$	$k(2j+1)+j+1$
4	other	1

Since $j + k = n$, we can rewrite (3) as

$$(n-j)(2j+1)+j+1 = -2j^2+2nj+1+n$$

First, we consider (3). Differentiating (3) with respect to j, we find

$$\frac{d}{dj}\left(-2j^2+2nj+1+n)+j+1\right) = -4j+2n \qquad \text{Equating to zero:}$$

$$0 = -4j+2n$$

$$j = \frac{n}{2}$$

Hence, if (3) is the maximum[1] runtime, it is maximised where $j = \frac{n}{2}$, yielding a runtime of

$$\frac{n}{2}(n+1)+\frac{n}{2}+1 = \frac{1}{2}n^2+n+1$$

Considering (2), we note that since j is the only independent variable, and $j+k=m$, j must be bounded above by $\frac{n}{2}$. Additionally, since the expression for (2) is a monotonically increasing function of j, maximising j also maximises the runtime. Then, if (2) is the maximum runtime, it is maximised where $j = \frac{n}{2}$, yielding a runtime of

$$\frac{n}{2}(n+1)+\frac{n}{2}+1 = \frac{1}{2}n^2+n+1$$

Hence (2) is consistent with (3), are both maximised where $j = k = \frac{n}{2}$, and run longer than (1) or (4).

[1] The second derivative is vacuously and uniformly negative.

(b) Suppose that $f_M(n)$ is the running time of M, ie.

$f_m(n) =$ the maximum number of steps M takes before halting over all inputs of size n.

Assuming the statement from part (a), compute

i. $f_M(2)$

With results from (a), we obtain

$$f_M(2) = \frac{1}{2}2^2 + 2 + 1 = 5$$

and hence $f_M(2) = 5$.

ii. $f_M(4)$

With results from (a), we obtain

$$f_M(4) = \frac{1}{2}4^2 + 4 + 1 = 13$$

and hence $f_M(4) = 13$.

iii. $f_M(6)$

With results from (a), we obtain

$$f_M(6) = \frac{1}{2}6^2 + 6 + 1 = 25$$

and hence $f_M(6) = 25$.

iv. $f_M(8)$

With results from (a), we obtain

$$f_M(8) = \frac{1}{2}8^2 + 8 + 1 = 41$$

and hence $f_M(8) = 41$.

(c) What is the tightest Big-O class that $f_M(n)$ belongs to? Justify your answer by discussing how the machine processes inputs.

With results from part (a):

Note that, for any input that does not start with a 0, the machine immediately rejects after just one transition from q_0 to q_{rej}. Additionally, for any input that contains j zeroes and no ones, the machine rejects during the transition from q_1 to q_{rej}, as there is no 1 to transition to q_2, meaning $j + 1$ steps are taken.
For well-formed inputs (ie. those of the form $0^j 1^k$ and $j, k > 0$), note that we can break the computation into steps:

- Transition $q_0 \to q_1$: Cross off the first zero **(1 step)**
- Transition $q_1 \to q_2 \to q_3$: Cycle $\min(j-1, k-1)$ times: **($\min(j-1, k-1) \cdot (2j+1)$ steps)**
  - Transition $q_1 \to q_1$: Skip as many zeroes and Ys as necessary to get to the first 1 **($j - 1$ steps)**

  - Transition $q_1 \to q_2$: Cross off the next one with a Y, and move left **(1 step)**
  - Transition $q_2 \to q_2$: Move left until we arrive at the rightmost X **($j-1$ steps)**
  - Transition $q_2 \to q_3$: Move right to the next character **(1 step)**
  - Transition $q_3 \to q_1$: Cross off the next zero **(1 step)**
- Then:
  - If $j \leq k$: Repeat the cycle once more, without transitioning back to q_1. From q_3, transition to q_4, consuming a Y. **($2j+1$ steps)** Then:
    - If $j < k$: Since j ones were crossed off, there are $k - j > 0$ remaining. Hence, transition from q_4 to itself $j-1$ times to read off the remaining Ys, and then transition once more to q_{rej}. **(j steps)**
    - If $j = k$: All the ones were crossed off. Hence, transition from q_4 to itself $j-1$ times to read off the remaining Ys, and then transition once more to q_{acc}. **(j steps)**
  - If $j \geq k$: Repeat the cycle once more. There are now no more ones left on the tape. **($2j+1$ steps)** Then:
    - If $j > k$: Transition from q_1 to itself $j-1-k$ times to read off the remaining zeroes, and then k times to read off the Ys, and then once more to read a $\sqcup$ and reject. **($1+j$ steps)**.
    - If $j = k$: Transition from q_1 to itself $j-1-k$ times to read off the remaining zeroes, and then $k = j$ times to read off the Ys, and then once more to read a $\sqcup$ and reject. **(j steps)**.

Thus, in total, for each non-degenerate input of length $n > 0$ where j, k denote the number of zeroes and ones respectively, the runtime is

	j, k	runtime
1	$j > 1,\ k = 0$	j
2	$j, k > 1,\ j \leq k$	$j(2j+1)+j+1$
3	$j, k > 1,\ j \geq k$	$k(2j+1)+j+1$
4	other	1

Since $j + k = n$, we can rewrite (3) as

$$(n-j)(2j+1)+j+1 = -2j^2 + 2nj + 1 + n$$

First, we consider (3). Differentiating (3) with respect to j, we find

$$\frac{d}{dj}\left(-2j^2 + 2nj + 1 + n) + j + 1\right) = -4j + 2n \qquad \text{Equating to zero:}$$

$$0 = -4j + 2n$$

$$j = \frac{n}{2}$$

Hence, if (3) is the maximum[2] runtime, it is maximised where $j = \frac{n}{2}$, yielding a runtime of

$$\frac{n}{2}(n+1) + \frac{n}{2} + 1 = \frac{1}{2}n^2 + n + 1$$

Considering (2), we note that since j is the only independent variable, and $j + k = m$, j must be bounded above by $\frac{n}{2}$. Additionally, since the expression for (2) is a monotonically increasing function of j, maximising j also maximises the runtime. Then, if (2) is the maximum runtime, it is maximised where $j = \frac{n}{2}$, yielding a runtime of

$$\frac{n}{2}(n+1) + \frac{n}{2} + 1 = \frac{1}{2}n^2 + n + 1$$

Hence (2) is consistent with (3), are both maximised where $j = k = \frac{n}{2}$, and run longer than (1) or (4).

[2] The second derivative is vacuously and uniformly negative.

Then the runtime of an input of size n is bounded above by the polynomial

$$f(n) = \frac{1}{2}n^2 + n + 1$$

We posit that $f(n) \in \Theta(n^2)$. First, we show $f(n) \in O(n^2)$ with the witnesses $c = 1$ and $n_0 = 3$:

$$\begin{aligned} f(n) &\leq cn^2 & n > n_0 \\ \frac{1}{2}n^2 + n + 1 &\leq n^2 \\ 0 &\leq \frac{1}{2}n^2 - n - 1 = g(n) \end{aligned}$$

Note that

$$\frac{dg}{dn} = n - 1 > 0 \text{ for } n > 1 \qquad \text{and} \qquad g(\sqrt{3}+1) = 0$$

Since the derivative is positive for $n > 1$, the difference between the two functions grows for all $n > 3 = n_0 > \sqrt{3} + 1 \approx 2.73 > 1$. Hence

$$\begin{aligned} 0 &\leq \frac{1}{2}n^2 - n - 1 & n > n_0 = 3 \\ \frac{1}{2}n^2 + n + 1 &\leq n^2 & n > 3 \end{aligned}$$

as requested. It follows that $f(n) \in O(n^2)$ as shown by the witnesses c and n_0. Then we show $f(n) \in \Omega(n^2)$ with the witnesses $c = \frac{1}{4}$ and $n_0 = 0$:

$$\begin{aligned} f(n) &\geq cn^2 & n > n_0 \\ \frac{1}{2}n^2 + n + 1 &\geq \frac{1}{4}n^2 \\ \frac{1}{4}n^2 + n + 1 &\geq 0 \end{aligned}$$

Note that

$$\frac{dg}{dn} = \frac{1}{2}n + 1 > 0 \text{ for } n > -2 \qquad \text{and} \qquad g(-2) = 0$$

Since the derivative is positive for $n > -2$, the difference between the two functions grows for all $n > 0 = n_0 > -2$. Hence

$$\begin{aligned} \frac{1}{4}n^2 + n + 1 &\geq 0 & n > 0n_0 = 0 \\ \frac{1}{2}n^2 + n + 1 &\geq \frac{1}{4}n^2 & n > 0 \end{aligned}$$

as requested. It follows that $f(n) \in \Omega(n^2)$ as shown by the witnesses c and n_0, and hence $f(n) \in \Theta(n^2)$. It follows by definition of Big-Theta that we have

$$\exists k_a\ \exists k_b\ \exists n_0\ \forall n > n_0\ \left(k_a n^2 \leq f(n) \leq k_b n^2\right)$$

Therefore, $O(n^2)$ must be the tightest Big-O bound on $f(n)$, since any stricter Big-O class will fail to bound $f(n)$ from above as it will grow slower than $O(n^2)$.

3. *Polynomial-time mapping reductions*

   Consider the sets

$$\begin{aligned} EMP_{NFA} &= \{\langle N \rangle \mid \text{such that } N \text{ is an NFA over } \{0,1\} \text{ that accepts the string } \varepsilon.\} \\ Z_{NFA} &= \{\langle N \rangle \mid \text{such that } N \text{ is an NFA over } \{0,1\} \text{ that accepts the string } 0.\} \end{aligned}$$

(a) Design a polynomial-time mapping reduction F from EMP_{NFA} to Z_{NFA}. (The runtime is in terms of the number of states of the input NFA.)

Define the mapping-reduction function:

F: On input s:

i. Typecheck s. If s is not of the form $\langle N \rangle$ where N is a NFA over $\{0,1\}$ output $const_{rej}$, an encoding of some constant NFA over $\{0,1\}$ that does not accept the string 0. Otherwise, $s = \langle N \rangle$ and let $N = (Q_0, \{0,1\}, q_{00}, \delta_0, F_0)$.

ii. Construct the NFA $N' = (Q, \Sigma, q_0, \delta, F)$ as

$$\begin{aligned}
Q &= Q_0 \cup \{q_{accnew}\} \\
\Sigma &= \{0,1\} \\
q_0 &= q_{00} \\
\delta : Q \times \Sigma \mapsto \mathcal{P}(Q) &= \begin{cases} \delta(q,s) = \delta_0(q,s) & q \notin F_0 \\ \delta(q,s) = \delta_0(q,s) & q \in F_0 \text{ and } s \neq 0 \\ \delta(q,s) = \delta_0(q,s) \cup \{q_{accnew}\} & q \in F_0 \text{ and } s = 0 \end{cases} \\
F &= \{q_{accnew}\}
\end{aligned}$$

iii. Output $\langle N' \rangle$.

(b) Justify why it is polynomial time.

Each step of the mapping-reduction runs either in constant time, or scales linearly with n, the number of states in the NFA. Recall that in a graph with n nodes, there can be at most

$$\frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

edges. Since NFAs can be represented as graphs where each state is a node and each transition is an edge, the sum of the number of transitions and states is bounded above by

$$\underbrace{\frac{1}{2}n^2 - \frac{1}{2}n}_{\text{no. transitions}} + \underbrace{n}_{\text{no. states}} = \frac{1}{2}n^2 + \frac{1}{2}n$$

Any reasonable encoding of an NFA should scale linearly in length with the number of transitions and states, and as such typechecking occurs in $O(\frac{1}{2}n^2 + \frac{1}{2}n) = O(n^2)$ time, which is polynomial. Additionally, the construction, encoding, and output of the NFA N' occurs in $O(n)$ time since we need only alter fixed and trivial aspects of N (and N is known, and already on the tape) to generate N': specifically, adding new transitions from $f \in F_0$ to q_{accnew}, adding the state q_{accnew}, and setting $F = \{q_{accnew}\}$ must occur in at most $O(n)$ time to account for shifting parts of the encoding and adding new states and transitions, for any nonpathological encoding.

(c) Show that $x \in EMP_{NFA}$ iff $F(x) \in Z_{NFA}$. (A deterministic Turing machine cannot simulate an NFA directly; it would have to first convert the NFA to a DFA. Recall that an NFA with n states corresponds to a DFA with at most 2^n states.)

To show

$$x \in EMP_{NFA} \leftrightarrow F(x) \in Z_{NFA}$$

we show both directions:

($\rightarrow$) If $x \in EMP_{NFA}$, then $x = \langle N \rangle$ where N is an NFA which accepts the empty string. Denote the accept states of N by F_0. It follows that in the computation of x by the computable function F, we obtain $F(x) = \langle N' \rangle$, where N' is an NFA with accept states F, otherwise identical to N, except $F = \{q_{accnew}\}$, and transitions have been added from every $f \in F_0$ to q_{accnew} upon reading a 0. Then it immediately follows that when running N' on the input 0, ε–transitions can be taken to at least one $f \in F_0$ (since N accepts ε), from which point we can read the input 0 and immediately transition to q_{accnew}. Since $q_{accnew} \in F$, we are assured that N' accepts 0, and it follows that $\langle N' \rangle = F(x) \in Z_{NFA}$ as required.

($\leftarrow$) We seek to prove

$$F(x) \in Z_{NFA} \rightarrow x \in EMP_{NFA}.$$

Towards this, we proceed using proof by contrapositive, and will demonstrate that

$$x \notin EMP_{NFA} \rightarrow F(x) \notin Z_{NFA}$$

If $x \notin EMP_{NFA}$, then we know that x is either an invalid encoding of an NFA, or x is a valid encoding of an NFA, but does not accept ε. We begin case analysis:

- *Case 1.* x is not an encoding of an NFA.
  In this case, our computable function F fails typechecking (step i) and outputs $const_{rej}$, an encoding of some NFA over $\{0, 1\}$ that does not accept 0. It immediately follows that $F(x) = const_{rej}$, and vacuously $F(x) \notin Z_{NFA}$ as required.

- *Case 2.* x is an encoding of an NFA that does not accept 0.
  In this case, our computable function outputs an encoding of the NFA N', which has been constructed to be identical to N, except all former accept states are no longer accept states, and have been connected such that reading a 0 transitions from each former accept state to the new accept state, q_{accnew}.

  We will prove by contradiction that $F(x) = \langle N' \rangle \notin Z_{NFA}$. Assume that in fact $F(x) \in Z_{NFA}$. Then, N' accepts 0. For this to be true, there must exist at least one ordered sequence of connected states S representing the computation of 0 by N' that begins at q_0 and ends at q_{accnew} composed exclusively of ε–transitions and exactly one 0 transition (ie. a transition taken by reading a single 0). Because the sequence S ends at q_{accnew} and it is only possible to transition to q_{accnew} by starting at one of the former accept states and reading a 0, we know that the sequence S must be composed of some number of ε–transitions to one of the former accept states, followed by a 0 transition to q_{accnew}:

  $$S = \{q_0,\ q_a \in Q,\ q_b \in Q, \qquad \ldots \qquad q_f \in F_0,\ q_{accnew}\}$$

  However, the existence of ε–transitions to the former accept states would imply that the original NFA N accepts ε, which is a direct contradiction. Hence the assumption was false, and $F(x)$ cannot possibly be in Z_{NFA}.

Hence we have proven that if $x \notin EMP_{NFA}$, then $F(x) \notin Z_{NFA}$. It follows that the contrapositive is true, and we have proven

$$F(x) \in Z_{NFA} \rightarrow x \in EMP_{NFA}$$

as required.

Thus we have shown both directions of the biconditional

$$x \in EMP_{NFA} \leftrightarrow F(x) \in Z_{NFA}$$

to be true. We have therefore also proven that $F(x)$ is indeed a mapping-reduction from EMP_{NFA} to Z_{NFA}.