Hardness Results for Intersection Non-Emptiness

Michael Wehar
University at Buffalo
mwehar@buffalo.edu

January 16, 2015

Abstract

We carefully reexamine a construction of Karakostas, Lipton, and Viglas (2003) to show that the intersection non-emptiness problem for DFA's (deterministic finite automata) characterizes the complexity class NL. In particular, if restricted to a binary work tape alphabet, then there exist constants c_1 and c_2 such that for every k intersection non-emptiness for k DFA's is solvable in $c_1 k \log(n)$ space, but is not solvable in $c_2 k \log(n)$ space. We optimize the construction to show that for an arbitrary number of DFA's intersection non-emptiness is not solvable in $o(\frac{n}{\log(n)\log(\log(n))})$ space. Furthermore, if there exists a function $f(k) = o(k)$ such that for every k intersection non-emptiness for k DFA's is solvable in $n^{f(k)}$ time, then P $\neq$ NL. If there does not exist a constant c such that for every k intersection non-emptiness for k DFA's is solvable in n^c time, then P does not contain any space complexity class larger than NL.

1 Introduction

Let $\mathcal{A}$ denote a class of machines. The intersection non-emptiness problem for $\mathcal{A}$, denoted by $\text{IE}_{\mathcal{A}}$, consists of all finite lists of machines in $\mathcal{A}$ whose underlying languages have a non-empty intersection. By fixing the number of machines in the input to k, one obtains intersection non-emptiness for k machines which we denote by k-$\text{IE}_{\mathcal{A}}$. Intersection non-emptiness problems can be motivated by the following scenario. Consider that you are trying to construct an object x for a particular application. You propose a finite list of conditions for x to satisfy such that each condition can be decided by a machine in $\mathcal{A}$. An algorithm that solves intersection non-emptiness for $\mathcal{A}$ provides a method for checking if there exists an object x satisfying the proposed conditions.

Let $\mathrm{IE}_{\mathcal{D}}$ denote the intersection non-emptiness problem for DFA's. One can solve $\mathrm{IE}_{\mathcal{D}}$ by checking reachability in a product machine. Given an input consisting of k machines each of size at most m, the product machine has size at most m^k. Therefore, checking reachability takes at most $m^{O(k)}$ time. Is it possible to solve $\mathrm{IE}_{\mathcal{D}}$ more efficiently? In [5], it was shown that $\mathrm{IE}_{\mathcal{D}}$ is PSPACE-complete. Consider restricting the number of machines in the input of $\mathrm{IE}_{\mathcal{D}}$ by a function $g(n)$ where n is the total input length. In [6], it was shown that if $g(n)$ is sublinear and log-space-constructible, then such a restriction yields a complete problem for $\mathrm{NSPACE}(g(n)\log(n))$. In [4], it was shown that the existence of a more efficient algorithm for $\mathrm{IE}_{\mathcal{D}}$ would imply a separation result. In particular, if there exists a function $f(k) = o(k)$ such that $\mathrm{IE}_{\mathcal{D}}$ is solvable in $m_1 \cdot m_2^{f(k)}$ time where m_1 is the size of a designated largest machine and all other machines have size at most m_2, then $\mathrm{NL} \neq \mathrm{P}$.

In this paper, we carefully reexamine and optimize the construction from [4] in order to prove new results. We show that if restricted to a binary work tape alphabet, then there exist constants c_1 and c_2 such that for every k, $k\text{-}\mathrm{IE}_{\mathcal{D}} \in \mathrm{NSPACE}(c_1 k \log(n))$ and $k\text{-}\mathrm{IE}_{\mathcal{D}} \notin \mathrm{NSPACE}(c_2 k \log(n))$. Then, we introduce an optimized construction to show that $\mathrm{IE}_{\mathcal{D}} \notin \mathrm{NSPACE}(o(\frac{n}{\log(n)\log(\log(n))}))$. Finally, we combine these results with a diagonalization argument to show that if there exists a function $f(k) = o(k)$ such that for every k, $k\text{-}\mathrm{IE}_{\mathcal{D}} \in \mathrm{DTIME}(n^{f(k)})$, then $\mathrm{P} \neq \mathrm{NL}$. If there does not exist a constant c such that for every k, $k\text{-}\mathrm{IE}_{\mathcal{D}} \in \mathrm{DTIME}(n^c)$, then $\mathrm{NSPACE}(f(n)) \nsubseteq \mathrm{P}$ for all $f(n) = \omega(\log(n))$ such that f is space-constructible.

2 Notation and Conventions

The input for $\mathrm{IE}_{\mathcal{D}}$ is an encoding of a finite list of DFA's. For each encoding, n will denote the length and k will denote the number of machines that are represented. For each natural number k, $k\text{-}\mathrm{IE}_{\mathcal{D}}$ denotes a restriction of the $\mathrm{IE}_{\mathcal{D}}$ problem such that we only accept inputs that encode at most k machines.

Whenever we use the term Turing machine, we refer to a deterministic or non-deterministic machine with a two-way read only input tape and a two-way read/write work tape. For our purposes, we will only consider Turing machines where the work tape alphabet is binary. A work tape over a binary alphabet will be referred to as a binary work tape. A cell on a binary work tape will be referred to as a bit cell.

For each k, there are acceptance problems for space and time bounded Turing machines denoted by $N^S_{k\log}$ and $D^T_{n^k}$, respectively. $N^S_{k\log}$ refers to the problem where we are given an encoding of a non-deterministic Turing machine M with a binary work tape and an input s. We accept (M, s) if and only if M accepts s using at most $k\log(n)$ work tape bit cells where n denotes the length of s. $D^T_{n^k}$ is defined similarly for n^k deterministic time. We denote by $\text{NSPACE}^2(h(n))$ the set of problems solvable by a non-deterministic Turing machine using at most $h(n)$ work tape bit cells. Such classes are used to measure the binary space complexity of problems [2]. We associate $N^S_{k\log}$ with $\text{NSPACE}^2(k\log(n))$ and $D^T_{n^k}$ with $\text{DTIME}(n^k)$.

3 Binary Space Complexity

We introduce a function $\text{S}_{\text{NL}}(k)$ that measures the actual space complexities of the $N^S_{k\log}$ problems. In particular, $\text{S}_{\text{NL}}(k)$ is defined as follows:

$$\text{S}_{\text{NL}}(k) := \min\{\, d \in \mathbb{N} \mid N^S_{k\log} \in \text{NSPACE}^2(d\log(n))\,\}. \tag{1}$$

In this section, we sketch how one could apply standard techniques from the space hierarchy theorem to prove that there exist constants c_1 and c_2 such that for every k sufficiently large, $N^S_{k\log} \in \text{NSPACE}^2(c_1 k\log(n))$ and $N^S_{k\log} \notin \text{NSPACE}^2(c_2 k\log(n))$. Using the function $\text{S}_{\text{NL}}(k)$, we express this result as $\text{S}_{\text{NL}}(k) = \Theta(k)$.

Proposition 1. $\text{S}_{\text{NL}}(k) = O(k)$.

Sketch of proof. Using the simulation found in any common proof of the space hierarchy theorem, one shows that $N^S_{\log} \in \text{NL}$. Further, one shows $\text{S}_{\text{NL}}(k) = O(k)$ by using padding to reduce $N^S_{k\log}$ to $N^S_{\log}$ for every k. □

Proposition 2. $\text{S}_{\text{NL}}(k) = \Omega(k)$.

Sketch of proof. Using the standard diagonalization argument found in any common proof of the non-deterministic space hierarchy theorem, one shows $\text{S}_{\text{NL}}(k)$ $= \Omega(k)$. Notice that in order to carry out the diagonalization one needs to show there exists c such that for all k,

$$\mathrm{NSPACE}^2(k\log(n)) \subseteq co\text{-}\mathrm{NSPACE}^2(ck\log(n)). \tag{2}$$

First, one applies the result NL $=$ co-NL to show that there exists c such that $N^S_{\log} \in co\text{-}\mathrm{NSPACE}^2(c\log(n))$. Further, one shows (2) by using padding to reduce $N^S_{k\log}$ to $N^S_{\log}$ for every k. □

Corollary 3. $\mathrm{S}_{\mathrm{NL}}(k) = \Theta(k)$.

4 Reductions

We introduce a function $\mathrm{S}_{\mathrm{IE}}(k)$ that measures the actual space complexities of the k-$\mathrm{IE}_{\mathcal{D}}$ problems. In particular, $\mathrm{S}_{\mathrm{IE}}(k)$ is defined as follows:

$$\mathrm{S}_{\mathrm{IE}}(k) := \min\{\, d \in \mathbb{N} \mid k\text{-}\mathrm{IE}_{\mathcal{D}} \in \mathrm{NSPACE}^2(d\log(n))\,\}. \tag{3}$$

In this section, we carefully reexamine the construction from [4] to show that there exist constants c_1 and c_2 such that for every k sufficiently large, k-$\mathrm{IE}_{\mathcal{D}}$ $\in \mathrm{NSPACE}^2(c_1 k\log(n))$ and k-$\mathrm{IE}_{\mathcal{D}} \notin \mathrm{NSPACE}^2(c_2 k\log(n))$. Using the function $\mathrm{S}_{\mathrm{IE}}(k)$, we can express this result as $\mathrm{S}_{\mathrm{IE}}(k) = \Theta(\mathrm{S}_{\mathrm{NL}}(k)) = \Theta(k)$.

Proposition 4. $\mathrm{S}_{\mathrm{IE}}(k) = O(k)$.

Sketch of proof. As was previously discussed, one can solve $\mathrm{IE}_{\mathcal{D}}$ by checking reachability in a product machine. A state of the product machine can be stored as a string of $k\log(n)$ bits. Given such a state, we can non-deterministically guess which state comes next. There exists a path from an initial state to a final state if and only if there exists a path from an initial state to a final state of length at most n^k. Therefore, k-$\mathrm{IE}_{\mathcal{D}}$ is solvable using at most $ck\log(n)$ bits for some constant c. □

Theorem 5. $\mathrm{S}_{\mathrm{IE}}(k) = \Omega(\mathrm{S}_{\mathrm{NL}}(k))$.

Proof. We will describe a reduction from $N^S_{k\log}$ to k-$\mathrm{IE}_{\mathcal{D}}$. Then, we will discuss encoding details to show that this is a log-space reduction.

Let a $k\log(n)$ space bounded non-deterministic Turing machine M of size n_{M} and an input string s of length n_s be given. Together, an encoding of M and s

represent an arbitrary input for $N^S_{k\log}$. Let n denote the total size of M and s combined i.e. $n := n_{\mathrm{M}} + n_s$.

Our first task is to construct k DFA's, denoted by $< D_i >_{i\in[k]}$, each of size at most $p(n)$ for some fixed polynomial p such that M accepts s if and only if $\bigcap_{i\in[k]} \mathrm{L}(D_i)$ is non-empty. The DFA's will read in a string that represents a computation of M on s and verify that the computation is valid and accepting. The work tape of M will be split into k sections each consisting of $\log(n_s)$ sequential bits of memory. The ith DFA, D_i, will keep track of the ith section and verify that it is managed correctly. In addition, all of the DFA's will keep track of the input and work tape head positions. We will achieve a better simulation in Theorem 7 where we split up the management of the tape head positions to separate DFA's. The following two concepts are essential to our construction.

A *section i configuration* of M is a tuple of the form

$$(\text{state, input position, work position, } i\text{th section of work tape}).$$

A *forgetful configuration* of M is a tuple of the form

$$(\text{state, input position, work position, write bit}).$$

The states of D_i are identified with section i configurations. The alphabet characters are identified with forgetful configurations. Intuitively, D_i reads in forgetful configurations that represent where to move the tape heads next and how the current bit cell should be manipulated.

Formally, the transitions for the DFA D_i are defined as follows. Let a forgetful configuration a and section i configurations r_1 and r_2 be given. It's possible that either the work tape position of r_1 is in the ith section, or the work tape position is in another section. In the first case, there is a transition from state r_1 with alphabet character a to state r_2 if (1a) going from r_1 to r_2 represents a valid transition of M on input s, (1b) the ith section of r_2 appropriately changes according to the write bit of a, and (1c) a and r_2 agree on state, input position, and work position. In the second case, there is a transition from state r_1 with alphabet character a to state r_2 if (2a) r_1 and r_2 agree on the ith section of the work tape, and (2b) a and r_2 agree on state, input position, and work position.

We assert without proof that for every string x, x represents a valid accepting computation of M on s if and only if $x \in \bigcap_{i\in[k]} \mathrm{L}(D_i)$. Therefore, M accepts s if and only if $\bigcap_{i\in[k]} \mathrm{L}(D_i)$ is non-empty.

We show that the D_i's have size at most $p(n)$ for some fixed polynomial p. Each D_i consists of a start state, a list of final states, and a list of transitions where each transition consists of two states and an alphabet character. Each state is represented by a section i configuration and each alphabet character is represented by a forgetful configuration. Therefore, in total there are $n_{\mathrm{M}} \cdot n_s \cdot k\log(n_s) \cdot 2^{\log(n_s)}$ section i configurations and $n_{\mathrm{M}} \cdot n_s \cdot k\log(n_s)$ forgetful configurations. Hence, there exists a fixed two variable polynomial q such that each D_i has size at most $q(n,k)$. Since k is fixed, one can blow up the degree of q to get a polynomial p such that p doesn't depend on k and each D_i has size at most $p(n)$.

It should be clear from the preceding that there is a fixed polynomial $t(n)$ such that for every k, $N^S_{k\log}$ is $t(n)$-time reducible to k-$\mathrm{IE}_{\mathcal{D}}$. However, we want to show that there is a constant c such that for every k, $N^S_{k\log}$ is $c\log(n)$-space reducible to k-$\mathrm{IE}_{\mathcal{D}}$. We accomplish this by describing how to print the string encoding of the D_i's to an auxiliary write only output tape using at most $c\log(n)$ space for some constant c.

We will describe how to print the transitions for each D_i and leave the remaining encoding details to the reader. We use a bit string i to represent the current DFA and two bit strings j_1 and j_2 to represent section i configurations. We iterate through every combination of i, j_1, and j_2. If D_i has a transition from j_1 to j_2, then we print (i, j_1, a, j_2) where a is the forgetful configuration that agrees with j_2. We assert that checking whether to print (i, j_1, a, j_2) requires no more than $d\log(k) + d\log(n)$ bits for some constant d. Therefore, in printing the encoding of the D_i's, we use no more than $c\log(k) + c\log(n)$ bits for some constant c. For each k, when n is sufficiently large, the $\log(k)$ term goes away. It follows that for every k, $N^S_{k\log}$ is $c\log(n)$-space reducible to k-$\mathrm{IE}_{\mathcal{D}}$. □

Corollary 6. $\mathrm{S}_{\mathrm{IE}}(k) = \Theta(\mathrm{S}_{\mathrm{NL}}(k)) = \Theta(k)$.

Proof. By Corollary 3, we have $\mathrm{S}_{\mathrm{NL}}(k) = \Theta(k)$. Applying Proposition 4 and Theorem 5, we get that $\mathrm{S}_{\mathrm{IE}}(k) = \Theta(\mathrm{S}_{\mathrm{NL}}(k)) = \Theta(k)$. □

Theorem 7. $\mathrm{IE}_{\mathcal{D}} \notin \mathrm{NSPACE}(o(\frac{n}{\log(n)\log(\log(n))}))$.

Proof. By the non-deterministic space hierarchy theorem, we may choose a problem Q such that $Q \in \mathrm{NSPACE}(n)$, but $Q \notin \mathrm{NSPACE}(o(n))$. Choose $c \in \mathbb{N}$ and a non-deterministic Turing machine M that solves Q using at most cn bit cells.

We optimize the construction from the proof of Theorem 5 to show that if $\mathrm{IE}_{\mathcal{D}} \in \mathrm{NSPACE}(o(\frac{n}{\log(n)\log(\log(n))}))$, then $Q \in \mathrm{NSPACE}(o(n))$. Since we know that $Q \notin \mathrm{NSPACE}(o(n))$, it follows that $\mathrm{IE}_{\mathcal{D}} \notin \mathrm{NSPACE}(o(\frac{n}{\log(n)\log(\log(n))}))$.

Let an input string s for M of length n be given. Our task is to construct $(c+1)\cdot n$ DFA's each with at most $d\log(n)$ states for some constant d such that M accepts s if and only if the DFA's have a non-empty intersection. The DFA's will read in a bit string that represents a computation of M on s and verify that the computation is valid and accepting. In this construction, we split up the management of the tape head positions to separate DFA's. There are n DFA's, denoted by $< I_i >_{i\in[n]}$, that manage the input tape and there are cn DFA's, denoted by $< W_i >_{i\in[cn]}$, that manage the work tape. The following concept is essential to our construction.

An *informative configuration* of M is a tuple of the form

(state, input position, current input bit, work position, current work bit).

The DFA's will read in a sequence of informative configurations that are encoded as bit strings. In contrast to the previous construction, the DFA's will have a binary input alphabet.

Each DFA is assigned to manage a bit position of either the input tape or work tape. Each I_i stores the ith input tape bit and operates as follows. It reads each informative configuration and checks if it represents the input position i. If it does not, then it ignores the informative configuration and moves on to the next one. However, if it does represent the input position i, then it checks that the stored bit matches the current input bit and uses the current work bit to check that the input position and state validly transition to the next informative configuration. Each W_i stores the ith work tape bit and operates as follows. It reads each informative configuration and checks if it represents the work position i. If it does not, then it ignores the informative configuration and moves on to the next one. However, if it does represent position i, then it checks that the stored bit matches the current work bit and uses the current input bit to both modify the stored bit and check that the work position and state validly transition to the next informative configuration. It's important to remark that DFA's for boundary positions such as I_1, I_n, W_1, and W_{cn} cannot allow the input position or work position to go outside $[n]$ or $[cn]$, respectively.

We assert without proof that for every bit string x, x represents a valid accepting computation of M on s if and only if $x \in \bigcap_{i\in[n]} \mathrm{L}(I_i)$ and $x \in \bigcap_{i\in[cn]} \mathrm{L}(W_i)$.

Therefore, M accepts s if and only if there exists a string x such that $x \in \bigcap_{i\in[n]} \mathrm{L}(I_i)$ and $x \in \bigcap_{i\in[cn]} \mathrm{L}(W_i)$.

A DFA with $\log(cn)$ states can be constructed to recognize a fixed binary number $i \in [cn]$. Since a tape position i could only transition to $i-1$, i, or $i+1$ in one step, it follows that a DFA with $d\log(n)$ states for some constant d can be constructed to check the validity of transitioning to the next informative configuration. Therefore, we can construct each DFA with at most $d\log(n)$ states for some constant d.

We described how to construct $(c+1)\cdot n$ DFA's each with at most $d\log(n)$ states for some constant d whose intersection is non-empty if and only if M accepts s. Since the total length of the string encoding of $< I_i >_{i\in[n]}$ combined with $< W_i >_{i\in[cn]}$ is at most $n\log(n)\log(\log(n))$, it follows that $\mathrm{IE}_{\mathcal{D}} \in \mathrm{NSPACE}(o(\frac{n}{\log(n)\log(\log(n))}))$ implies $Q \in \mathrm{NSPACE}(o(n))$. We obtain the desired result because $Q \notin \mathrm{NSPACE}(o(n))$. □

5 Space vs Time

We introduce functions $\mathrm{R}_{\mathrm{NL}}(k)$ and $\mathrm{R}_{\mathrm{IE}}(k)$ that measure the actual time complexities of $N^S_{k\log}$ and k-$\mathrm{IE}_{\mathcal{D}}$, respectively. In particular, $\mathrm{R}_{\mathrm{NL}}(k)$ and $\mathrm{R}_{\mathrm{IE}}(k)$ are defined as follows:

$$\mathrm{R}_{\mathrm{NL}}(k) := \min\{\, d \in \mathbb{N} \mid N^S_{k\log} \in \mathrm{DTIME}(n^d)\,\} \tag{4}$$

$$\mathrm{R}_{\mathrm{IE}}(k) := \min\{\, d \in \mathbb{N} \mid k\text{-}\mathrm{IE}_{\mathcal{D}} \in \mathrm{DTIME}(n^d)\,\}. \tag{5}$$

In this section, we show that if there exists a function $f(k) = o(k)$ such that for every k, $N^S_{k\log} \in \mathrm{DTIME}(n^{f(k)})$, then $\mathrm{P} \neq \mathrm{NL}$. Using the function $\mathrm{R}_{\mathrm{NL}}(k)$ we can express this result as if $\mathrm{R}_{\mathrm{NL}}(k) = o(k)$, then $\mathrm{P} \neq \mathrm{NL}$. Notice that by using the reduction from Theorem 5, we also have $\mathrm{R}_{\mathrm{IE}}(k) = \Theta(\mathrm{R}_{\mathrm{NL}}(k))$. It follows that if $\mathrm{R}_{\mathrm{IE}}(k) = o(k)$, then $\mathrm{P} \neq \mathrm{NL}$.

Proposition 8. $\mathrm{R}_{\mathrm{IE}}(k) = \Theta(\mathrm{R}_{\mathrm{NL}}(k))$.

Theorem 9. *If* $\mathrm{R}_{\mathrm{NL}}(k) = o(k)$, *then* $\mathrm{NL} \neq \mathrm{P}$.

Proof. Suppose that $\mathrm{NL} = \mathrm{P}$. Since $D^T_n \in \mathrm{P}$, we have $D^T_n \in \mathrm{NL}$. Choose $d \in \mathbb{N}$ such that $D^T_n \in \mathrm{NSPACE}^2(d\log(n))$. Further, by using padding to reduce $D^T_{n^k}$ to D^T_n for every k, one can show that there exists d' such that for all k, $D^T_{n^k}$

$\in \text{NSPACE}^2(d'k\log(n))$. Choose such a constant d' satisfying for all k, $D^T_{n^k} \in \text{NSPACE}^2(d'k\log(n))$.

Suppose for sake of contradiction that $\text{R}_{\text{NL}}(k) = o(k)$. By Proposition 2, we may choose c such that for all k sufficiently large

$$N^S_{k\log} \notin \text{NSPACE}^2(\left\lfloor \frac{k}{c} \right\rfloor \log(n)). \tag{6}$$

Since $\text{R}_{\text{NL}}(k) = o(k)$, for all k sufficiently large

$$\text{R}_{\text{NL}}(k) < \left\lfloor \frac{k}{cd'} \right\rfloor. \tag{7}$$

Choose m satisfying $N^S_{m\log} \notin \text{NSPACE}^2(\lfloor \frac{m}{c} \rfloor \log(n))$ and $\text{R}_{\text{NL}}(m) < \lfloor \frac{m}{cd'} \rfloor$. Therefore,

$$N^S_{m\log} \in \text{DTIME}(o(n^{\lfloor \frac{m}{cd'} \rfloor})). \tag{8}$$

Since $D^T_{n^k} \in \text{NSPACE}^2(d'k\log(n))$ for all k,

$$D^T_{n^{\lfloor \frac{m}{cd'} \rfloor}} \in \text{NSPACE}^2(d' \left\lfloor \frac{m}{cd'} \right\rfloor \log(n)) \subseteq \text{NSPACE}^2(\left\lfloor \frac{m}{c} \right\rfloor \log(n)). \tag{9}$$

Since we can trivially reduce every problem in $\text{DTIME}(o(n^{\lfloor \frac{m}{cd'} \rfloor}))$ to $D^T_{n^{\lfloor \frac{m}{cd'} \rfloor}}$,

$$N^S_{m\log} \in \text{DTIME}(o(n^{\lfloor \frac{m}{cd'} \rfloor})) \subseteq \text{NSPACE}^2(\left\lfloor \frac{m}{c} \right\rfloor \log(n)) \tag{10}$$

which is a contradiction because $N^S_{m\log} \notin \text{NSPACE}^2(\lfloor \frac{m}{c} \rfloor \log(n))$. □

Corollary 10. *If* $\text{R}_{\text{IE}}(k) = o(k)$*, then* $\text{NL} \neq \text{P}$*.*

Next, we show that if $\text{R}_{\text{NL}}(k)$ is unbounded, then P does not contain any space complexity class larger than NL. Since $\text{R}_{\text{IE}}(k) = \Theta(\text{R}_{\text{NL}}(k))$, it follows that if $\text{R}_{\text{IE}}(k)$ is unbounded, then P does not contain any space complexity class larger than NL.

For every function f, let N^S_f denote the acceptance problem for $f(n)$-space bounded non-deterministic Turing machines. N^S_f is of particular interest to us if it is non-deterministically solvable in $f(n)$ space.

Theorem 11. *If* $\text{R}_{\text{NL}}(k)$ *is unbounded, then* $N^S_f \notin \text{P}$ *for all functions* $f(n) = \omega(\log(n))$*.*

Proof. We will prove the contrapositive. Suppose that $N_f^S \in \mathrm{P}$ for some function $f(n) = \omega(\log(n))$. By assumption, we may choose $c \in \mathbb{N}$ and a deterministic Turing machine T such that T solves N_f^S in at most $O(n^c)$ time. Let $k \in \mathbb{N}$ be given. Choose a non-deterministic Turing machine M that solves $N_{k\log}^S$ using at most $O(\log(n))$ bit cells. We can deterministically solve $N_{k\log}^S$ in at most $O(n^c)$ time by feeding T an encoding of M and the input string. Since k is arbitrary, $N_{k\log}^S$ is solvable in $O(n^c)$ time for every k. It follows that $\mathrm{R_{NL}}(k)$ is bounded. □

Corollary 12. *If* $\mathrm{R_{NL}}(k)$ *is unbounded, then* $\mathrm{NSPACE}(f(n)) \nsubseteq \mathrm{P}$ *for all* $f(n) = \omega(\log(n))$ *such that* f *is space-constructible.*

Proof. Suppose $\mathrm{R_{NL}}(k)$ is unbounded. Let a function $f(n) = \omega(\log(n))$ such that f is space-constructible be given. Apply the preceding theorem to get that $N_f^S \notin \mathrm{P}$. Since f is space-constructible, one can use the simulation found in any common proof of the space hierarchy theorem to show that $N_f^S \in \mathrm{NSPACE}(f(n))$. Since $N_f^S \notin \mathrm{P}$ and $N_f^S \in \mathrm{NSPACE}(f(n))$, it follows that $\mathrm{NSPACE}(f(n)) \nsubseteq \mathrm{P}$. □

Corollary 13. *If* $\mathrm{R_{IE}}(k)$ *is unbounded, then* $\mathrm{NSPACE}(f(n)) \nsubseteq \mathrm{P}$ *for all* $f(n) = \omega(\log(n))$ *such that* f *is space-constructible.*

6 Conclusion

In Section 4, we showed that $\mathrm{S_{NL}}(k) = \mathrm{S_{IE}}(k) = \Theta(k)$. Therefore, we think of intersection non-emptiness for DFA's as characterizing the complexity class NL. Further, we showed that $\mathrm{IE}_\mathcal{D} \notin \mathrm{NSPACE}(o(\frac{n}{\log(n)\log(\log(n))}))$. In Section 5, we showed that if $\mathrm{R_{IE}}(k) = o(k)$, then $\mathrm{NL} \neq \mathrm{P}$ and if $\mathrm{R_{IE}}(k)$ is unbounded, then $\mathrm{NSPACE}(f(n)) \nsubseteq \mathrm{P}$ for all $f(n) = \omega(\log(n))$ such that f is space-constructible. Therefore, the asymptotic complexity of $\mathrm{R_{IE}}(k)$ determines the relationship between space and time complexity classes.

There are several related problems that appear to be harder than k-$\mathrm{IE}_\mathcal{D}$, but easier than $N_{k\log}^S$. For example, consider intersection non-emptiness for k NFA's, non-emptiness for k-turn 2DFA's, and intersection non-emptiness for k DFA's and a one-counter automaton. We can use $\mathrm{S_{IE}}(k) = \Theta(\mathrm{S_{NL}}(k))$ and $\mathrm{R_{IE}}(k) = \Theta(\mathrm{R_{NL}}(k))$ as squeeze theorems to show that all of these problems are of "equivalent" difficulty. Also, one could define a function that maps the k-$\mathrm{IE}_\mathcal{D}$ problems to their actual circuit complexities. The asymptotic complexity of such a function could determine the relationship between NL vs NP and P/poly vs space complexity classes [4].

Several related intersection non-emptiness problems have been studied. There are two such problems that we would like to mention. In [10], intersection non-emptiness for acyclic DFA's, which are DFA's without directed cycles, was shown to be NP-complete. We assert that one could modify the construction from the proof of Theorem 5 to reduce the acceptance problem for n-time and $k\log(n)$-space bounded non-deterministic Turing machines to intersection non-emptiness for k acyclic DFA's. Also, in [11], intersection non-emptiness for tree automata was shown to be EXPTIME-complete. In an upcoming paper, the author and Joseph Swernofsky introduce time complexity lower bounds for intersection non-emptiness for tree automata.

Acknowledgments

I greatly appreciate all of the help and suggestions that I received. In particular, I would like to thank Christos Kapoutsis for suggestions related to the constructions, Joseph Swernofsky for proof reading and many discussions, Richard Lipton and Kenneth Regan for calling attention to my results in an article on their blog [8], and the many anonymous referees. I would especially like to thank all those at Carnegie Mellon University who offered their help and support for my honors thesis on the same topic. In particular, I would like to thank my thesis advisor, Klaus Sutner, and my thesis committee members, Manuel Blum and Richard Statman.

References

[1] Michael Blondin, Andreas Krebs, and Pierre McKenzie. The complexity of intersecting finite automata having few final states. *Computational Complexity (CC)*, 2014 (to appear).

[2] Oded Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, New York, 2008.

[3] Neil D. Jones, Y. Edmund Lien, and William T. Laaser. New problems complete for nondeterministic log space. *Mathematical Systems Theory 10*, 1976.

[4] G. Karakostas, R. J. Lipton, and A. Viglas. On the complexity of intersecting finite state automata and NL versus NP. *Theoretical Computer Science*, 302:257–274, 2003.

[5] Dexter Kozen. Lower bounds for natural proof systems. *Proc. 18th Symp. on the Foundations of Computer Science*, pages 254–266, 1977.

[6] Klaus-Jörn Lange and Peter Rossmanith. The emptiness problem for intersections of regular languages. *Lecture Notes in Computer Science*, 629:346–354, 1992.

[7] R. J. Lipton. On the intersection of finite automata. *Gödel's Lost Letter and P=NP*, August 2009.

[8] R. J. Lipton and K. W. Regan. The power of guessing. *Gödel's Lost Letter and P=NP*, November 2012.

[9] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal*, 1959.

[10] Narad Rampersad and Jeffrey Shallit. Detecting patterns in finite regular and context-free languages. *Information Processing Letters*, 110, 2010.

[11] Margus Veanes. On computational complexity of basic decision problems of finite tree automata. *UPMAIL Technical Report 133*, 1997.

[12] Michael Wehar. Intersection emptiness for finite automata. Honors thesis, Carnegie Mellon University, 2012.