On the Complexity of Intersecting Regular, Context-free, and Tree Languages

Joseph Swernofsky
Independent Researcher
joseph.swernofsky@gmail.com

Michael Wehar
University at Buffalo
mwehar@buffalo.edu

September 3, 2016

Abstract

We apply a construction of Cook (1971) to show that the intersection non-emptiness problem for one PDA (pushdown automaton) and a finite list of DFA's (deterministic finite automata) characterizes the complexity class P. In particular, we show that there exist constants c_1 and c_2 such that for every k, intersection non-emptiness for one PDA and k DFA's is solvable in $O(n^{c_1 k})$ time, but is not solvable in $O(n^{c_2 k})$ time. Then, for every k, we reduce intersection non-emptiness for one PDA and 2^k DFA's to non-emptiness for multi-stack pushdown automata with k-phase switches to obtain a tight time complexity lower bound. Further, we revisit a construction of Veanes (1997) to show that the intersection non-emptiness problem for tree automata also characterizes the complexity class P. We show that there exist constants c_1 and c_2 such that for every k, intersection non-emptiness for k tree automata is solvable in $O(n^{c_1 k})$ time, but is not solvable in $O(n^{c_2 k})$ time.

1 Introduction

To determine whether a mathematical object exists one could start by listing constraints for the proposed object to satisfy. Then, for each constraint one could build a verifier to computationally determine whether an input satisfies the constraint. If each constraint can be verified by an automaton, does that mean one could efficiently determine whether there exists an object that satisfies all of the constraints?

We will investigate problems where we are given an encoding of a finite list of automata and want to determine whether there exists a string that satisfies each automaton in the list. A problem of this form is referred to as an intersection non-emptiness problem because it is equivalent to determining whether the languages associated with the automata have a non-empty intersection.

Each of the intersection non-emptiness problems that we investigate will be viewed as an infinite family of problems indexed on the natural numbers by the number of machines k. For each problem in such a family, we will prove a time complexity lower bound. One may be tempted to view each family of problems as a single parameterized problem. Such an interpretation is fine as long as one realizes that we aren't simply proving a single parameterized complexity lower bound. Rather, we are proving a lower bound for each of the infinitely many fixed levels of the parameterized problem.

In Section 2, we introduce some basic results that allow us to compare infinite families of problems. These results will be used to put all of our findings into a general framework to efficiently present our complexity lower bounds and shed light on the relationship between types of automata and complexity classes.

The intersection non-emptiness problem for DFA's, which we denote by $\text{IE}_{\mathcal{D}}$, is a well known PSPACE-complete problem [7]. Consider fixing the number of machines in the input. Let k-$\text{IE}_{\mathcal{D}}$ denote the restricted version of $\text{IE}_{\mathcal{D}}$ such that only inputs with at most k machines are accepted. In [18], the second author proved a tight non-deterministic space complexity lower bound for the k-$\text{IE}_{\mathcal{D}}$ problems. He showed that there exist c_1 and c_2 such that for every k, k-$\text{IE}_{\mathcal{D}} \in \text{NSPACE}(c_1 k \log(n))$ and k-$\text{IE}_{\mathcal{D}} \notin \text{NSPACE}(c_2 k \log(n))$ where space is measured relative to a fixed work tape alphabet. Therefore, we say that intersection non-emptiness for DFA's characterizes the complexity class NL.

First, we will investigate intersection non-emptiness for one PDA and a finite list of DFA's which we denote by $\text{IE}_{1\mathcal{P}+\mathcal{D}}$. We will show that there exist constants c_1 and c_2 such that for every k, k-$\text{IE}_{1\mathcal{P}+\mathcal{D}} \in \text{DTIME}(n^{c_1 k})$ and k-$\text{IE}_{1\mathcal{P}+\mathcal{D}} \notin \text{DTIME}(n^{c_2 k})$. In order to show the lower bound, we will reduce the acceptance problem for $k\log(n)$-space bounded auxiliary pushdown automata to k-$\text{IE}_{1\mathcal{P}+\mathcal{D}}$. Then, we will apply results from [4] to get that $k\log(n)$-space bounded auxiliary pushdown automata can be used to simulate $k\log(n)$-space bounded alternating Turing machines. Finally, we will apply results from [2] to get that $k\log(n)$-space bounded alternating Turing machines can be used to simulate n^k-time bounded deterministic Turing machines.

Next, we will investigate non-emptiness for multi-stack pushdown automata with k-phase switches which we denote by k-MPDA. From [12], we know that k-$\text{MPDA} \in \text{DTIME}(n^{O(2^k)})$. This result was shown by reducing k-MPDA to non-emptiness for graph automata with bounded tree width and then further reducing to non-emptiness for tree automata. We will show that this upper bound is tight. In particular, we will show that there exist constants c_1 and c_2 such that for every k, k-$\text{MPDA} \in \text{DTIME}(n^{c_1 2^k})$ and k-$\text{MPDA} \notin \text{DTIME}(n^{c_2 2^k})$. In order to show the lower bound, we will reduce 2^k-$\text{IE}_{1\mathcal{P}+\mathcal{D}}$ to k-MPDA and then apply the lower bound for 2^k-$\text{IE}_{1\mathcal{P}+\mathcal{D}}$ from the preceding section. In addition, we will present a lower bound for the dual of the k-MPDA problem.

Finally, we will investigate intersection non-emptiness for tree automata which we denote by $\text{IE}_{\mathcal{T}}$. In [16], it was shown that $\text{IE}_{\mathcal{T}}$ is EXPTIME-complete. We will show that there exist constants c_1 and c_2 such that for every k, k-$\text{IE}_{\mathcal{T}} \in$

DTIME($n^{c_1 k}$) and k-IE$_{\mathcal{T}} \notin$ DTIME($n^{c_2 k}$). In order to show the lower bound, we will reduce the acceptance problem for $k\log(n)$-space bounded alternating Turing machines to k-IE$_{\mathcal{T}}$. Then, we will again apply results from [2] to get that $k\log(n)$-space bounded alternating Turing machines can be used to simulate n^k-time bounded deterministic Turing machines.

2 Preliminaries

2.1 Complexity Classes

Each of the following complexity classes is associated with a machine class. In particular, a language X is in the complexity class if and only if there exists a machine M in the associated machine class such that M accepts X.

NL : Logarithmic space bounded non-deterministic Turing machines
AL : Logarithmic space bounded alternating Turing machines
AuxL : Logarithmic space bounded auxiliary pushdown automata
P : Polynomial time bounded deterministic Turing machines

Although NL $\subseteq$ P, it is not known if P = NL. However, using machine simulations, it was proven that P = AL in [2] and P = AuxL in [4].

If one carefully looks at the simulations from [2], one will notice that there are universal constants c_1 and c_2 such that for every k, each n^k-time bounded deterministic Turing machine can be simulated by a $c_1 k\log(n)$-space bounded alternating Turing machine and each $k\log(n)$-space bounded alternating Turing machine can be simulated by a $n^{c_2 k}$-time bounded deterministic Turing machine. Hence, not only are P and AL equivalent, but the DTIME(n^k) classes that make up P and the ASPACE($k\log(n)$) classes that make up AL are in some sense level-by-level equivalent to each other. This example should motivate the notion of level-by-level equivalence that we introduce in Section 2.3.

2.2 Acceptance Problems

We will specify a Turing machine model and introduce acceptance problems for the machine classes associated with NL, AL, AuxL, and P.

By a Turing machine, we are referring to a machine with a single two-way read-only input tape and a single two-way read/write binary work tape. The condition on the work tape being binary is significant. In particular, for space complexity, constants matter when the alphabet is fixed. By an $f(n)$-time bounded Turing machine, we mean a Turing machine that runs for at most $f(n)$ steps on all inputs of length n. By an $f(n)$-space bounded Turing machine, we mean a Turing machine that uses at most $f(n)$ cells on the binary work tape for all inputs of length n. By uses at most $f(n)$ cells, we mean that the tape head never moves to the right of

the $f(n)$th cell. Time and space bounded auxiliary pushdown automata can be defined similarly where the space bounds only apply to the auxiliary work tape. The space bounds do not apply to the stack.

The general form of an acceptance problem is as follows. Given an encoding of a machine M and an input x, does M accept x? For each k and each machine class that we discussed in Section 2.1, we can define an acceptance problem. Consider the following acceptance problems and their associated machine classes.

$$\begin{aligned} N^S_{k\log} &: k\log(n)\text{-space bounded non-deterministic Turing machines} \\ A^S_{k\log} &: k\log(n)\text{-space bounded alternating Turing machines} \\ Aux^S_{k\log} &: k\log(n)\text{-space bounded auxiliary pushdown automata} \\ D^T_{n^k} &: n^k\text{-time bounded deterministic Turing machines} \end{aligned}$$

2.3 Level-By-Level Equivalence

For each k and each machine class from Section 2.1, we defined an acceptance problem. In other words, for each machine class, we defined an infinite family of acceptance problems. These infinite families of problems characterize their associated machine classes and their associated complexity classes.

Let's look at an example. Consider the family $\{D^T_{n^k}\}_{k\in\mathbb{N}}$. The proof of the time hierarchy theorem has two parts: universal simulation and diagonalization. From universal simulation of deterministic Turing machines, we get a constant c_1 such that for every k, $D^T_{n^k} \in \text{DTIME}(n^{c_1 k})$. From diagonalization, we get a smaller constant c_2 such that for every k, $D^T_{n^k} \notin \text{DTIME}(n^{c_2 k})$. Therefore, we say that this family characterizes the complexity class P.

The notion of an infinite family characterizing a complexity class leads us to the concept of LBL (level-by-level) reducibility. This concept will allow us to compare the complexity of infinite families of problems.[1]

Given two infinite families of problems $X := \{X_k\}_{k\in\mathbb{N}}$ and $Y := \{Y_k\}_{k\in\mathbb{N}}$, we say that X is (polynomial time) LBL-reducible to Y if there exists a constant c such that for every k, there exists an $O(n^c)$-time bounded reduction from X_k to Y_k where k is treated as a constant. If X is LBL-reducible to Y, then we write $X \leq_L Y$. If X is LBL-reducible to Y and Y is LBL-reducible to X, then we say that X and Y are LBL-equivalent and write $X \equiv_L Y$. Notice that $\leq_L$ is transitive and $\equiv_L$ is an equivalence relation.

To simplify how one shows that an infinite family X is LBL-reducible to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$, we have the following proposition.

Proposition 1. *Let an infinite family X be given. If there exists c such that for every k, $X_k \in \text{DTIME}(n^{ck})$, then X is LBL-reducible to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$.*

[1] An LBL-reduction is an infinite family of reductions. Such a family of reductions can be viewed as the non-uniform analogue of an fpt-reduction [5]. We introduce the distinct notion of an LBL-reduction to emphasize that our lower bounds will apply to each problem in the respective family of problems.

To simplify how one shows a (near) tight time complexity lower bound for an infinite family X, we have the following proposition.

Proposition 2. *If an infinite family X is LBL-equivalent to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$, then there exist c_1 and c_2 such that for every k, $X_k \in \mathrm{DTIME}(n^{c_1 k})$ and $X_k \notin \mathrm{DTIME}(n^{c_2 k})$.*

The simulations that we mentioned in Section 2.1 lead to two significant examples of LBL equivalence. In particular, from the simulations in [2], we have that $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ is LBL-equivalent to $\{A^S_{k\log}\}_{k\in\mathbb{N}}$. Also, from the simulations in [4], we have that $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ is LBL-equivalent to $\{Aux^S_{k\log}\}_{k\in\mathbb{N}}$. Now, we can apply Proposition 2 and the LBL equivalences to obtain (near) tight time complexity lower bounds. For example, consider the LBL equivalence between $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ and $\{A^S_{k\log}\}_{k\in\mathbb{N}}$. By applying Proposition 2, we get that there exist c_1 and c_2 such that for every k, $A^S_{k\log} \in \mathrm{DTIME}(n^{c_1 k})$ and $A^S_{k\log} \notin \mathrm{DTIME}(n^{c_2 k})$.

We will further use the equivalences for deterministic time, alternating space, and auxiliary space to prove equivalences for intersection non-emptiness problems. Then, we will apply Proposition 2 to obtain (near) tight time complexity lower bounds for these problems.

3 One PDA and k DFA's

It is well known that the general intersection non-emptiness problem for DFA's is PSPACE-complete [7]. Further work has shown that variations of this problem are hard as well [9]. We consider the problem where in addition to a finite list of DFA's, we are also given a single pushdown automaton. Notice that it doesn't make sense to consider more than one PDA because the intersection non-emptiness problem for two pushdown automata is undecidable.

We will show that intersection non-emptiness for one PDA and k DFA's is equivalent to acceptance for n^k-time bounded deterministic Turing machines. In particular, we will show that $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ is LBL-equivalent to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$.

Using the product construction, one can solve each k-$\mathrm{IE}_{1\mathcal{P}+\mathcal{D}}$ problem in $O(n^{ck})$ time for some constant c. Further, one can apply Proposition 1 to get the following result.

Proposition 3. $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ *is LBL-reducible to* $\{D^T_{n^k}\}_{k\in\mathbb{N}}$.

In the following theorem, we reduce acceptance for space bounded auxiliary pushdown automata to intersection non-emptiness for one PDA and a finite list of DFA's. The reduction that we present is based on reductions from [6] and [7]. Our presentation is in the same format as that from the second author's previous work where he reduces acceptance for non-deterministic space bounded Turing machines to intersection non-emptiness for a finite list of DFA's [18].

Theorem 4. $\{Aux^S_{k\log}\}_{k\in\mathbb{N}}$ *is LBL-reducible to* $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$.

Proof. An auxiliary pushdown automaton has a stack, a two-way read-only input tape, and a single read/write work tape. We will restrict the read/write work tape to be binary and bound the amount of cells that the automaton can use in terms of the input length. In addition, we will only consider auxiliary pushdown automata where the stack alphabet is binary. Such restricted auxiliary PDA's are sufficient for carrying out the simulation in [4].

Let k be given. We will describe a reduction from $Aux^S_{k\log}$ to k-IE$_{1\mathcal{P}+\mathcal{D}}$. Let a $k\log(n)$-space bounded auxiliary pushdown automaton M of size n_{M} and an input string x of length n_x be given. Together, an encoding of M and x represent an arbitrary input for $Aux^S_{k\log}$. Let n denote the total size of M and x combined i.e. $n := n_{\mathrm{M}} + n_x$.

Our task is to construct one PDA and k DFA's, denoted by PD and $\{D_i\}_{i\in[k]}$, each of size at most $O(n^c)$ for some fixed constant c such that M accepts x if and only if $\text{Ł}(PD) \cap \bigcap_{i\in[k]} \text{Ł}(D_i)$ is non-empty.

The automata will read in a string that represents a computation of M on x and verify that the computation is valid and accepting. The PDA PD will verify that the stack is managed correctly while the DFA's will verify that the work tape is managed correctly. In particular, the work tape of M will be split into k sections each consisting of $\log(n_x)$ 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 tape head positions.

The following two concepts are essential to our construction.
A *section i configuration* of M is a tuple of the form:

(state, input position, work position, ith section of work tape).

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

(state, input position, work position, write bit, stack action, top bit).

The alphabet symbols are identified with forgetful configurations. The PDA PD only has two states. When it reads a forgetful configuration a, if a represents the top of the stack correctly, then PD loops in the initial/accepting state and pushes or pops based on the stack instruction that a represents. Otherwise, PD goes to the dead/rejecting state.

The states for the D_i's are identified with section i configurations. Each D_i has a single initial state. We identify this initial state with the section i configuration of M that represents the initial input and work positions, a blank ith section of the work tape, and the initial state of M. The final states of D_i represent accepting configurations of M.

Informally, the transitions are defined as follows. For each D_i, there is a transition from state r_1 to state r_2 with symbol a if a validly represents how the state and partial tapes for r_1 and r_2 could be manipulated in one step for the computation of M on input x. It's important to notice that in order to determine if there is a transition, the stack action and top bit of the stack must be taken into account.

We assert without proof that for every string y, y represents a valid accepting computation of M on x if and only if $y \in \text{Ł}(PD) \cap \bigcap_{i\in[k]} \text{Ł}(D_i)$. Therefore, M accepts x if and only if $\text{Ł}(PD) \cap \bigcap_{i\in[k]} \text{Ł}(D_i)$ is non-empty. By bounding the total number of section i configurations, one can show there exists a fixed two variable polynomial q such that each D_i has at most $q(n, k)$ states. Therefore, there is a constant d that does not depend on k such that each D_i has size at most $O(n^d)$ where k is treated as a constant. Further, we can compute each D_i's transition table by looping through every combination of a pair of states and an alphabet symbol, and marking the valid combinations. The number of possible combinations is a fixed polynomial blow-up from n^d. Therefore, we can compute the transition tables in $O(n^c)$ time for some slightly larger constant c that does not depend on k.

Since k was arbitrary, we have that for every k, there is an $O(n^c)$-time reduction from $Aux^S_{k\log}$ to k-$\text{IE}_{1\mathcal{P}+\mathcal{D}}$. □

In the preceding reduction, it was surprising that the PDA had a fixed number of states. Even more surprisingly, one could convert the automata constructed in the reduction to automata with a binary input alphabet. In doing so, the PDA can be made fixed. In other words, there is a fixed deterministic pushdown automaton for which the intersection non-emptiness problem is hard.

Corollary 5. *$\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ and $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ are LBL-equivalent.*

Proof. From Section 2, we know that $\{Aux^S_{k\log}\}_{k\in\mathbb{N}} \equiv_L \{D^T_{n^k}\}_{k\in\mathbb{N}}$. Further, we have $\{Aux^S_{k\log}\}_{k\in\mathbb{N}} \leq_L \{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}} \leq_L \{D^T_{n^k}\}_{k\in\mathbb{N}}$ from Proposition 3 and Theorem 4. Combine to obtain the desired result. □

Corollary 6. *$\exists c_1\, \exists c_2\, \forall k$ $k\text{-IE}_{1\mathcal{P}+\mathcal{D}} \in \text{DTIME}(n^{c_1 k})$ and $k\text{-IE}_{1\mathcal{P}+\mathcal{D}} \notin \text{DTIME}(n^{c_2 k})$.*

Proof. Combine Corollary 5 with Proposition 2. □

4 MPDA's with k-Phase Switches

A two-stack pushdown automaton can simulate a Turing machine. Therefore, the non-emptiness problem for such machines is undecidable. However, we can restrict how and when the machines can access their stacks to obtain classes of machines whose non-emptiness problems are decidable [12]. In particular, we will discuss the k-phase switches restriction. This restriction forces a machine to designate a stack for popping. In other words, a restricted machine can push to any stack, but only pop from the designated stack. The k refers to how many times the machine can switch which stack is designated. We refer to a machine with such a restriction as a multi-stack pushdown automaton with k-phase switches. For background on such machines, we refer the reader to [14]. We also investigate what we refer to as the dual machines. These machines can pop from any stack, but can only push to the designated stack.

We will denote the non-emptiness problem for multi-stack pushdown automata with k-phase switches by k-MPDA. Similarly, we will denote the non-emptiness

problem for the dual machines by k-*co*-MPDA. We will show that $\{k\text{-MPDA}\}_{k\in\mathbb{N}}$, $\{k\text{-}co\text{-MPDA}\}_{k\in\mathbb{N}}$, and $\{D^T_{n^{2^k}}\}_{k\in\mathbb{N}}$ are LBL-equivalent. As a result, we will obtain tight lower bounds for these non-emptiness problems.

Recently, the non-emptiness problem for a related class of infinite automata was shown to have a double exponential time lower bound [8]. In addition, the non-emptiness problem for ordered multi-stack pushdown automata was shown to have a double exponential time lower bound [1]. Our lower bound may be suggested by such sources, but we elegantly prove it using a novel reduction found in the proof of Theorem 8.

Proposition 7. *$\{k\text{-MPDA}\}_{k\in\mathbb{N}}$ and $\{k\text{-}co\text{-MPDA}\}_{k\in\mathbb{N}}$ are LBL-reducible to $\{D^T_{n^{2^k}}\}_{k\in\mathbb{N}}$.*

Sketch of proof. In [12], it was shown that k-MPDA and k-*co*-MPDA $\in$ DTIME($n^{O(2^k)}$) by a reduction to non-emptiness for graph automata with bounded tree width and further to non-emptiness for tree automata. Then, one can apply a variation of Proposition 1 to get the desired result. □

In the following theorem, we reduce intersection non-emptiness for one PDA and 2^k DFA's to non-emptiness for multi-stack pushdown automata with k-phase switches.

Theorem 8. *$\{2^k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ is LBL-reducible to $\{k\text{-MPDA}\}_{k\in\mathbb{N}}$.*

Sketch of proof. Let an input for 2^k-IE$_{1\mathcal{P}+\mathcal{D}}$ consisting of a PDA and 2^k DFA's be given. We will describe how to construct a multi-stack pushdown automaton M with k-phase switches whose language is non-empty if and only if the PDA and DFA's languages have a non-empty intersection.

The machine M will have k stacks. It will read its input and copy it onto all of the stacks besides the first stack. While it is reading the input, the first stack will be used to simulate the PDA on the input. Then, it will repeat the following procedure until each of the stacks have been designated once.

The procedure consists of popping from the designated stack and pushing what is being popped onto all of the other stacks. While it is popping, it is also simulating one DFA per copy of the input string or simulating one DFA in reverse per copy of the reversal of the input string. This will eventually create exponentially many copies of the input string followed by the reversal of the input string and lead to simulating each DFA or reversal on one of the copies.

If the PDA and all of the DFA's accept, then M will accept. Otherwise, M will reject. In total, we are able to simulate one PDA and $O(2^k)$ DFA's using only k-phase switches. Also, the size of M will be approximately the sum of the sizes of the PDA and DFA's. □

A related reduction can be given for the dual machines. The proof of Theorem 9 can be found in Appendix D.

Theorem 9. *$\{2^k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ is LBL-reducible to $\{k\text{-}co\text{-MPDA}\}_{k\in\mathbb{N}}$.*

Corollary 10. *$\{k\text{-MPDA}\}_{k\in\mathbb{N}}$, $\{k\text{-}co\text{-MPDA}\}_{k\in\mathbb{N}}$, and $\{D^T_{n^{2^k}}\}_{k\in\mathbb{N}}$ are LBL-equivalent.*

Sketch of proof. From Corollary 5, we have $\{2^k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}} \equiv_L \{D^T_{n^{2^k}}\}_{k\in\mathbb{N}}$. Further, we have $\{2^k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}} \leq_L \{k\text{-MPDA}\}_{k\in\mathbb{N}} \leq_L \{D^T_{n^{2^k}}\}_{k\in\mathbb{N}}$ from Proposition 7 and Theorem 8. Similarly, we have reductions for the dual machines. Combine to obtain the desired result. □

Corollary 11. *There exist c_1 and c_2 such that for every k:*

$$i)\ k\text{-MPDA } \textit{and } k\textit{-co-MPDA} \in \text{DTIME}(n^{c_1 2^k})$$
$$ii)\ k\text{-MPDA } \textit{and } k\textit{-co-MPDA} \notin \text{DTIME}(n^{c_2 2^k}).$$

Sketch of proof. Combine Corollary 10 and a variation of Proposition 2. □

5 k Tree Automata

It is known that the general intersection non-emptiness problem for deterministic top-down tree automata is EXPTIME-complete [3]. We will show that intersection non-emptiness for k deterministic top-down tree automata is equivalent to acceptance for n^k-time bounded deterministic Turing machines. In particular, we will show that $\{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}}$ is LBL-equivalent to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$. For background on decision problems for tree automata, we refer the reader to [3] and [13].

Using the product construction, one can solve each k-IE$_{\mathcal{T}}$ problem in $O(n^{ck})$ time for some constant c. Further, one can apply Proposition 1 to get the following result.

Proposition 12. *$\{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}}$ is LBL-reducible to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$.*

In the following theorem, we reduce acceptance for alternating Turing machines to intersection non-emptiness for tree automata. The reduction that we present is similar to that found in [16] and briefly described in [3]. Our presentation is in the same format as Theorem 4.

Theorem 13. *$\{A^S_{k\log}\}_{k\in\mathbb{N}}$ is LBL-reducible to $\{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}}$.*

Proof. An alternating Turing machine has existential states and universal states. Therefore, there are existential configurations and universal configurations. An existential configuration c leads to an accepting configuration if and only if there exists a valid transition out of c that leads to an accepting configuration. A universal configuration c leads to an accepting configuration if and only if every valid transition out of c leads to an accepting configuration. We will only consider alternating machines such that no universal configuration can have more than two valid outgoing transitions. We assert without proof that any alternating machine can be unraveled with intermediate universal states to satisfy this property in such a way that there is no more than a polynomial blow-up in the number of states.

Let k be given. We will describe a reduction from $A^S_{k\log}$ to k-IE$_{\mathcal{T}}$. Let a $k\log(n)$-space bounded alternating Turing machine M of size n_{M} and an input

string x of length n_x be given. Together, an encoding of M and x represent an arbitrary input for $A^S_{k\log}$. Let n denote the total size of M and x combined i.e. $n := n_{\mathrm{M}} + n_x$.

Our task is to construct k top-down deterministic tree automata, denoted by $\{T_i\}_{i\in[k]}$, each of size at most $O(n^c)$ for some fixed constant c such that M accepts x if and only if $\bigcap_{i\in[k]} Ł(T_i)$ is non-empty.

The tree automata will read in a labeled tree that represents a computation of M on x and verify that the computation is valid and accepting. The work tape of M will be split into k sections each consisting of $\log(n_x)$ sequential bits of memory. The ith tree automaton, T_i, will keep track of the ith section and verify that it is managed correctly. In addition, all of the tree automata will keep track of the tape head positions.

The following two concepts are essential to our construction.
A *section i configuration* of M is a tuple of the form:

(state, input position, work position, ith section of work tape).

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

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

The alphabet consists of symbols of arity 0, 1, and 2 such that each arity 0 symbol represents an accepting forgetful configuration, each arity 1 symbol represents an arbitrary forgetful configuration, and each arity 2 symbol represents a pair of forgetful configurations. We won't need any symbols of arity larger than 2 because each universal configuration has at most two outgoing transitions.

The states of T_i are identified with section i configurations. Each T_i has a single initial state. We identify this initial state with the section i configuration of M that represents the initial input and work positions, a blank ith section of the work tape, and the initial state of M.

We say that a section i configuration r extends a forgetful configuration a if r agrees with a on state, input position, and work position.

We say that a section i configuration r_1 transitions to a section i configuration r_2 on input x if either (a) the work position for r_1 is in the ith section and r_2 correctly represents how the tape positions and the ith section could change in one step of the computation on x, or (b) r_1 is not in the ith section and r_1 and r_2 agree on the ith section of the work tape.

For each T_i, we have the following transitions. Each arity 0 symbol a accepts on a state r if and only if r extends a and a represents an accepting state of M. Each arity 1 symbol a transitions from a state r_1 to a state r_2 if and only if (i) r_1 transitions to r_2 on input x (consistently with a's write bit), (ii) r_2 extends a, and (iii) if r_1 is a universal configuration and the work position of r_1 is in the ith section, then r_1 can only transition to r_2 on input x. Each arity 2 symbol (a_1, a_2) transitions from a state r to a pair of distinct states (r_1, r_2) if and only if

r transitions to r_1 on input x, r transitions to r_2 on input x, r_1 extends a_1, and r_2 extends a_2.

We assert without proof that for every labeled tree y, y represents a valid accepting computation of M on x if and only if $y \in \bigcap_{i\in[k]} \text{Ł}(T_i)$. Therefore, M accepts x if and only if $\bigcap_{i\in[k]} \text{Ł}(T_i)$ is non-empty. By bounding the total number of section i configurations, one can show there exists a fixed two variable polynomial q such that each T_i has at most $q(n,k)$ states. Therefore, there is a constant d that does not depend on k such that each T_i has size at most $O(n^d)$ where k is treated as a constant. Further, we can compute each T_i's transition table by looping through every combination of a pair of states and an alphabet symbol, and marking the valid combinations. The number of possible combinations is a fixed polynomial blow-up from n^d. Therefore, we can compute the transition tables in $O(n^c)$ time for some slightly larger constant c that does not depend on k.

Since k was arbitrary, we have that for every k, there is an $O(n^c)$-time reduction from $A^S_{k\log}$ to k-IE$_{\mathcal{T}}$. □

Corollary 14. *$\{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}}$ and $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ are LBL-equivalent.*

Proof. From Section 2, we know that $\{A^S_{k\log}\}_{k\in\mathbb{N}} \equiv_L \{D^T_{n^k}\}_{k\in\mathbb{N}}$. Further, we have $\{A^S_{k\log}\}_{k\in\mathbb{N}} \leq_L \{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}} \leq_L \{D^T_{n^k}\}_{k\in\mathbb{N}}$ from Proposition 12 and Theorem 13. Combine to obtain the desired result. □

Corollary 15. *$\exists c_1\, \exists c_2\, \forall k$ $k\text{-IE}_{\mathcal{T}} \in \text{DTIME}(n^{c_1 k})$ and $k\text{-IE}_{\mathcal{T}} \notin \text{DTIME}(n^{c_2 k})$.*

Proof. Combine Corollary 14 with Proposition 2. □

6 Conclusion

We introduced the notions of LBL reducibility and LBL equivalence for infinite families of problems.[2] We then used existing simulations to show that $\{D^T_{n^k}\}_{k\in\mathbb{N}}$, $\{A^S_{k\log}\}_{k\in\mathbb{N}}$, $\{Aux^S_{k\log}\}_{k\in\mathbb{N}}$, $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$, and $\{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}}$ are all polynomial time LBL-equivalent. Further, we applied that $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ and $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ are LBL-equivalent to show that $\{D^T_{n^{2^k}}\}_{k\in\mathbb{N}}$, $\{2^k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$, $\{k\text{-MPDA}\}_{k\in\mathbb{N}}$, and $\{k\text{-}co\text{-MPDA}\}_{k\in\mathbb{N}}$ are all polynomial time LBL-equivalent. By combining these equivalences with Proposition 2, we get (near) tight time complexity lower bounds for all of these problems.

We claim that all of the polynomial time LBL-reductions that we presented can be carefully optimized to become log-space LBL-reductions. Formally, we say that a family X if log-space LBL-reducible to a family Y if there exists a constant c and a function f such that for every k, there exists a $(c + o(1))\log(n)$-space reduction from X_k to Y_k where k is treated as a constant.

[2] These concepts serve as the non-uniform analogues of fpt reducibility and fpt equivalence from the subject of parameterized complexity theory [5].

The notion of log-space LBL equivalence can be used to express the P vs NL problem from structural complexity theory. Consider the machine classes consisting of polynomial time bounded deterministic Turing machines and log-space bounded non-deterministic Turing machines. These machine classes have associated acceptance problems $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ and $\{N^S_{k\log}\}_{k\in\mathbb{N}}$, respectively. One can show that P = NL if and only if $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ and $\{N^S_{k\log}\}_{k\in\mathbb{N}}$ are log-space LBL-equivalent.

Now, one might ask, "What's the relationship between P vs NL and intersection non-emptiness problems?" We know that $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ and $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ are log-space LBL-equivalent. In addition, from the second author's previous work [18], it can be shown that $\{N^S_{k\log}\}_{k\in\mathbb{N}}$ and $\{k\text{-IE}_{\mathcal{D}}\}_{k\in\mathbb{N}}$ are log-space LBL-equivalent. Therefore, we get that P = NL if and only if $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ and $\{k\text{-IE}_{\mathcal{D}}\}_{k\in\mathbb{N}}$ are log-space LBL-equivalent. In other words, P = NL if and only if adding a PDA does not increase the difficulty of the intersection non-emptiness problem for DFA's.

We showed that intersection non-emptiness problems for DFA's, PDA's, and tree automata characterize complexity classes. There are many more types of automata and we suggest that one may be able to prove more characterizations using the notion of LBL reducibility. From this perspective, we intend to investigate decision problems for tree automata with auxiliary memory.

Acknowledgments.

We greatly appreciate all of the help and suggestions that we received. We would especially like to thank Richard Lipton, Kenneth Regan, Atri Rudra, and all those from Carnegie Mellon University who were supportive of our research work while we were undergraduates.

References

[1] M. F. Atig, B. Bollig, and P. Habermehl. Emptiness of multi-pushdown automata is 2ETIME-Complete. *Developments in Language Theory*, 2008.

[2] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981.

[3] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, October 2007.

[4] Stephen A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *J. ACM*, 18(1):4–18, January 1971.

[5] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

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

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

[8] S. L. Torre, P. Madhusudan, and G. Parlato. An infinite automaton characterization of double exponential time. *CSL 2008*, pages 33–48, 2008.

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

[10] N. Limaye and M. Mahajan. Membership testing: Removing extra stacks from multi-stack pushdown automata. *LATA 2009*, pages 493–504, 2009.

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

[12] P. Madhusudan and Gennaro Parlato. The tree width of automata with auxiliary storage. *POPL 2011*, 2011.

[13] W. Martens and S. Vansummeren. Automata and logic on trees: Algorithms. ESSLLI 2007, 2007.

[14] S. L. Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. *LICS 2007*, pages 161–170, 2007.

[15] Leslie G. Valiant. *Decision Procedures for Families of Deterministic Pushdown Automata*. PhD thesis, University of Warwick, August 1973.

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

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

[18] Michael Wehar. Hardness results for intersection non-emptiness. *ICALP 2014 (Part II)*, pages 354–362, 2014.

A Appendix: Proof of Proposition 1

Let an infinite family X be given. Suppose that there exists c such that for every k, $X_k \in \text{DTIME}(n^{ck})$.

Let k be given. We will describe how to reduce X_k to $D^T_{n^k}$. Choose a deterministic Turing machine M that solves X_k in n^{ck} time. Therefore, M is n^{ck}-time bounded. We can modify M to get a machine M$'$ that is n^k-time bounded where

M′ accepts a string x if and only if x can be split into a m^c length padding and a length m string that is accepted by M for some natural number m.

Now, let a string y of length m be given. The reduction takes y and maps to an encoded pair consisting of M' and y padded with a string of length m^c. We have that $y \in X_k$ if and only if the encoded pair is in $D^T_{n^k}$. Since M′ is fixed, this is a $O(n^c)$-time bounded reduction where c does not depend on k. □

B Appendix: Proof of Proposition 2

Let an infinite family X be given. Suppose that X is LBL-equivalent to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$ with reduction constants c_1 and c_2.

Let k be given. Consider the $O(n^{c_1})$-time bounded reduction from X_k to $D^T_{n^k}$. By universal simulation, there is a deterministic Turing machine M that solves $D^T_{n^k}$ in $n^{d_1 k}$ time for a constant d_1 that does not depend on k. If we combine the reduction with the machine M, we get a machine that solves X_k in $O(n^{c_1 d_1 k})$ time. Therefore, $X_k \in \text{DTIME}(n^{c_1 d_1 k})$.

Consider the $O(n^{c_2})$-time bounded reduction from $D^T_{n^k}$ to X_k. By diagonalization, we get a constant d_2 that does not depend on k such that $D^T_{n^k} \notin \text{DTIME}(n^{d_2 k})$. Hence, we can't solve X_k in $O(n^{\frac{d_2 k}{c_2}})$ time or else we would be able to combine such a solver with the reduction to show that $D^T_{n^k} \in \text{DTIME}(n^{d_2 k})$ which can't happen. Therefore, $X_k \notin \text{DTIME}(n^{\frac{d_2 k}{c_2}})$. □

C Appendix: Proof of Proposition 3

Non-emptiness for a single PDA is known to be solvable in polynomial time. Hence, we may choose c such that non-emptiness for PDA's is in $\text{DTIME}(n^c)$.

Let k be given. Let an input consisting of one PDA and k DFA's be given. Let m denote the number of states from the largest automaton. Let n denote the total length of the input's string encoding. The product of the PDA and k DFA's is a PDA with at most m^{k+1} states and the product can be encoded by a string of length at most n^{k+1}. Now, we use the algorithm that decides non-emptiness for PDA's to solve non-emptiness for the product automaton in $O(n^{c(k+1)})$ time.

Since k is arbitrary, we have for all k, $k\text{-IE}_{1\mathcal{P}+\mathcal{D}} \in \text{DTIME}(n^{c(k+1)})$. We can choose a larger constant c' so that for all k, $k\text{-IE}_{1\mathcal{P}+\mathcal{D}} \in \text{DTIME}(n^{c'k})$. Now, we can apply Proposition 1 to get that $\{k\text{-IE}_{1\mathcal{P}+\mathcal{D}}\}_{k\in\mathbb{N}}$ is LBL-reducible to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$. □

D Appendix: Proof of Theorem 9

Let an input for $2^k\text{-IE}_{1\mathcal{P}+\mathcal{D}}$ consisting of a PDA and 2^k DFA's be given. We will describe how to construct a dual machine with k-*co*-phase switches that accepts some input if and only if the PDA and DFA's languages have a non-empty intersection.

The MPDA will have k stacks. It will read an input consisting of 2^k separated strings and repeat the following procedure. As it is reading the input, it will copy the strings onto the designated stack. In addition, while reading, it will trade-off between simulating one DFA per string and simulating one DFA in reverse per string. Eventually, the MPDA will non-deterministically guess that it reached the midpoint i.e. the point when the designated stack's height is equal to the length of what still needs to be read on the input tape. When this happens, it will designate a new stack. Now, as it continues reading the input, it will pop the previously designated stacks and make sure that the strings on the input tape match the strings on these stacks while still repeating this procedure from the beginning on the newly designated stack.

Essentially, this procedure allows the MPDA to read in a sequence of exponentially many separated strings and use the stacks to verify that the strings are all equivalent modulo reversal. All the while, the MPDA is simulating one DFA per string or simulating one DFA in reverse per reversal of the string.

Finally, when the procedure is finished and the end of the input is reached, all the stacks are empty besides one stack with exactly one copy of the string. We can designate a new stack to simulate the PDA on the one remaining copy of the string. It is alright for us to designate a new stack to simulate the PDA because the end of the input was reached and we already verified that all the strings on the input tape are the same modulo reversal.

If the PDA and all of the DFA's accept, then the MPDA will accept. Otherwise, it will reject. In total, we are able to simulate one PDA and $O(2^k)$ DFA's using only k-*co*-phase switches. Also, the size of the MPDA will be approximately the sum of the sizes of the PDA and DFA's. □

E Appendix: Proof of Proposition 12

Non-emptiness for a single deterministic top-down tree automaton is known to be solvable in polynomial time. Hence, we may choose c such that 1-IE$_{\mathcal{T}} \in$ DTIME(n^c).

Let k be given. Let an input consisting of k tree automata with at most m states each be given. Let n denote the total length of the input's string encoding. Take the product of the k tree automata. The resulting automaton will have at most m^k states and can be encoded by a string of length at most n^k. Now, we use the algorithm that decides 1-IE$_{\mathcal{T}}$ to solve non-emptiness for the product tree automaton in $O(n^{ck})$ time.

Since k is arbitrary, we have for all k, k-IE$_{\mathcal{T}} \in$ DTIME(n^{ck}). Now, we can apply Proposition 1 to get that $\{k\text{-IE}_{\mathcal{T}}\}_{k\in\mathbb{N}}$ is LBL-reducible to $\{D^T_{n^k}\}_{k\in\mathbb{N}}$. □