Relative Prefix Distance Between Languages

Timothy Ng, David Rappaport, and Kai Salomaa

School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada
{ng, daver, ksalomaa}@cs.queensu.ca

Abstract. The prefix distance between two words x and y is defined as the number of symbols in x and y that do not belong to their longest common prefix. The relative prefix distance from a language L_1 to a language L_2, if finite, is the smallest integer k such that for every word in L_1, there is a word in L_2 with prefix distance at most k. We study the prefix distance between regular, visibly pushdown, deterministic context-free, and context-free languages. We show how to compute the distance between regular languages and determine whether the distance is bounded. For deterministic context-free languages and visibly pushdown languages, we show that the relative prefix distance to and from regular languages is decidable.

1 Introduction

Distances on words are typically defined to compare the similarity between two words. The prefix distance between two words x and y is defined as the number of symbols in x and y that do not belong to their longest common prefix. In some sense, the prefix distance measures the distance of objects arranged in a hierarchical structure [19]. The edit distance, which counts the minimum number of insertions, deletions, and substitutions required to transform one word into another, is more commonly used for string comparisons. However, the prefix distance is often simpler to compute than the edit distance and may suffice for certain applications, such as defect measurement [12] and intrusion detection [5]. Beyond string comparisons, the prefix distance also has interesting topological properties and is used to characterize the subsequentiality of functions [2].

These distance measures can be extended to sets of words, or languages. The standard topological definition for a distance over words when extended to languages takes the minimum of the distances between a word in each language and has been well studied [11, 10, 13, 14, 17]. Choffrut and Pighizzini [8] consider an alternate definition for distance between languages, called the relative or Hausdorff distance. The relative distance from one language to another is defined as the supremum over all words in the first language of the distance to the second language and is non-symmetric. A symmetric distance can be attained by simply taking the minimum of the relative distance in each direction.

Choffrut and Pighizzini study the distance between languages from the point of view of relations on words. They study various distances on subclasses of deterministic rational relations, showing that questions about distances are

decidable for recognizable relations and undecidable for deterministic rational relations with respect to the prefix, suffix, and subword distances.

The relative distance has applications in verification, as a generalization of the language inclusion problem. For instance, Benedikt et al. [3, 4] consider the one-sided relative edit distance to measure the cost of repairing regular specifications. They give an algorithm for deciding when the distance between regular languages is bounded and give complexity results for computing the edit distance between regular languages. Chatterjee et al. [7] consider the same problems for a context-free language and a language belonging to a subclass of the context-free languages.

In this paper, we study the relative prefix distance between various classes of languages. We show that the relative prefix distance between DFAs (deterministic finite automata) can be computed in polynomial time while computing the relative prefix and suffix distance between NFAs (nondeterministic finite automata) is PSPACE-complete. We also consider the computational problem of computing the relative prefix distance between more general classes of languages. For a fixed value k, deciding whether the relative prefix distance from a context-free language to a DFA (respectively, an NFA) language is at most k can be done in polynomial time (respectively, is EXPTIME-complete). On the other hand, computing the relative prefix distance from a regular language to a context-free language is undecidable. We show that the prefix distance neighbourhood of a DCFL (deterministic context-free language) is deterministic and this yields an algorithm to compute the relative prefix distance from a regular language to a DCFL. Finally, we show that computing the relative prefix distance from one visibly pushdown language to another is EXPTIME-complete while computing the relative prefix distance from a DCFL to a visibly pushdown language, or vice versa, is undecidable.

2 Preliminaries

Here we briefly recall some definitions and notation used in the paper. For all unexplained notions on finite automata and regular languages the reader may consult the textbook by Shallit [18] or the survey by Yu [20]. A survey of distances is given by Deza and Deza [9].

In the following, Σ is always a finite alphabet, the set of words of Σ is denoted Σ^*, and ε denotes the empty word. The reversal of a word $w \in \Sigma^*$ is denoted by x^R. The length of a word w is denoted by $|w|$. The cardinality of a finite set S is denoted $|S|$ and the power set of S is 2^S. A word $w \in \Sigma^*$ is a *subword* or *factor* of x if and only if there exist words $u, v \in \Sigma^*$ such that $x = uwv$. If $u = \varepsilon$, then w is a *prefix* of x. If $v = \varepsilon$, then w is a *suffix* of x.

A *nondeterministic finite automaton* (NFA) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, δ is a transition function $\delta : Q \times \Sigma \to 2^Q$, $q_0 \in Q$ is a set of initial states, and $F \subseteq Q$ is a set of final states. A word $w \in \Sigma^*$ is *accepted* by A if for some $q_0 \in Q_0$, $\delta(q_0, w) \cap F \neq \emptyset$ and the language recognized by A consists of all words accepted by A. An NFA is a

deterministic finite automaton (DFA) if for all $q \in Q$ and $a \in \Sigma$, $\delta(q, a)$ either consists of one state or is undefined.

A *pushdown automaton* (PDA) is a tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, Γ is a stack alphabet, δ is a transition function $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \to 2^{Q \times \Gamma^*}$, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of final states. For a transition $(q', \beta) \in \delta(q, a, \alpha)$, the PDA pops the symbol α from the top of the stack and pushes the symbols β onto the stack.

A configuration of a PDA is a triple (q, w, π) where $q \in Q$ is the current state, $w \in \Sigma^*$ is the remaining input, and $\pi \in \Gamma^*$ is the contents of the stack. For a transition $(q', \beta) \in \delta(q, a, \alpha)$, we write $(q, aw, \alpha\pi) \vdash (q', w, \beta\pi)$. We denote by $\vdash^*$ a sequence of transitions between the two configurations. Then a word w is accepted by a PDA if $(q_0, w, \varepsilon) \vdash^* (q_f, \varepsilon, \pi)$ for $q_f \in F$ and $\pi \in \Gamma^*$.

A PDA is a *deterministic pushdown automaton* (DPDA) if the transition function satisfies $|\delta(q, a, \alpha)| \leq 1$ for all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $\alpha \in \Gamma$, and that for all $q \in Q$ and $\alpha \in \Gamma$, if $\delta(q, \varepsilon, \alpha) \neq \emptyset$, then $\delta(q, a, \alpha) = \emptyset$ for all $a \in \Sigma$. It is well known that the class of deterministic context-free languages is a proper subclass of the context-free languages.

A *visibly pushdown automaton* is a tuple $V = (Q, \Sigma, \Gamma, \delta, q_0, F)$ as in a pushdown automaton, with the additional constraint that the alphabet Σ and transition function δ are partitioned into three sets

- call actions Σ_c with the transition function $\delta_c : Q \times \Sigma_c \to 2^{Q \times \Sigma}$,
- return actions Σ_r with the transition function $\delta_r : Q \times \Sigma_r \times \Gamma \to 2^Q$,
- internal actions Σ_i with the transition function $\delta_i : Q \times \Sigma_i \to 2^Q$.

The stack operations of a VPA are determined entirely by the input symbols. Specifically, upon reading a call action, the VPA must push to the stack and upon reading a return action, the VPA must pop from the stack. It is well known that the class of languages accepted by VPAs, the *visibly pushdown languages*, is a proper subclass of the deterministic context-free languages [1].

A *finite state transducer*, or transducer, is a tuple $T = (Q, \Sigma, \Delta, \delta, q_0, F)$, where Q is a finite set of states, Σ and Δ are finite alphabets, $\delta \subseteq Q \times \Sigma^* \times \Delta^* \times Q$ is a finite set of transitions, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of accepting states. An accepting computation of T is a sequence of elements of δ

$$(q_0, x_1, y_1, q_1)(q_1, x_2, y_2, q_2) \cdots (q_{n-1}, x_n, y_n, q_n)$$

where $q_n \in F$. We say the transducer maps the input string $x = x_1 \cdots x_n$ to the output string $y = y_1 \cdots y_n$, which we denote by $x \to_T y$. The set $\{(x, y) \mid x \to_T y\}$ is the relation realized by T. We define the transduction realized by T by

$$T(x) = \{y \in \Delta^* \mid x \to_T y\}.$$

2.1 Distances

A function $d : \Sigma^* \times \Sigma^* \to \mathbb{N} \cup \{0\}$ is a *distance* if it satisfies for all $x, y \in \Sigma^*$

1. $d(x, y) = 0$ if and only if $x = y$,

2. $d(x, y) = d(y, x)$,
3. $d(x, z) \leq d(x, y) + d(y, z)$ for $z \in \Sigma^*$.

A distance between words can be extended to a distance between a word $w \in \Sigma^*$ and a language $L \subseteq \Sigma^*$ by

$$d(w, L) = \min\{d(w, w') \mid w' \in L\}.$$

We define the relative distance [8] from a language L_1 to language L_2 to be

$$d(L_1|L_2) = \sup\{d(w_1, L_2) \mid w_1 \in L_1\}.$$

In other words, $d(L_1|L_2)$ is the value of the maximum distance from any word in L_1 to the language L_2. Note that under this definition, $d(L_1|L_2)$ is not symmetric and can be unbounded.

The *prefix distance* of x and y counts the number of symbols which do not belong to the longest common prefix of x and y. It is defined by

$$d_p(x, y) = |x| + |y| - 2 \cdot \max_{z \in \Sigma^*}\{|z| \mid x, y \in z\Sigma^*\}.$$

The suffix distance and subword distance can be similarly defined by considering the number of symbols of x and y which do not belong to the longest common suffix (subword, respectively) of x and y.

The *neighbourhood* of a language L of radius k with respect to a distance d is the set of all words $v \in \Sigma^*$ such that $d(u, v) \leq k$ for some $u \in L$ [6]. More formally,

$$E(L, d, k) = \{w \in \Sigma^* \mid d(w, L) \leq k\}.$$

It has been shown that the neighbourhoods of a regular language with respect to the prefix, suffix, and subword distances are regular [16].

3 Relative Prefix Distance Between Regular Languages

Choffrut and Pighizzini [8] showed that the main questions about the almost-reflexivity of a recognizable relation is decidable. Here, we show that these questions are computable in polynomial time if the languages are given as DFAs and that they are PSPACE-complete for NFAs.

Since the relative distance can be unbounded, we would like to characterize when, for two given languages, the distance is finite. In the following result, we show that the distance is either bounded by a function of the state complexity of the languages, or it is unbounded. First, we establish a simple lemma.

Lemma 1. *Let A_1 and A_2 be two NFAs recognizing L_1 and L_2 with n_1 and n_2 states respectively. Suppose $u \in L_1, v \in L_2$ and let p be the longest word satisfying $u = pu', v = pv'$. Then there exists a word $pw \in L_2$ such that $|w| < n_2 - 1$.*

Proof. This follows directly from the Pumping Lemma [18]. □

Theorem 2. *Let L_1, L_2 be regular languages recognized by NFAs A_1 and A_2 with n_1 and n_2 states, respectively. Suppose $d_p(L_1|L_2)$ is bounded. Then,*

$$d_p(L_1|L_2) \leq n_1 + n_2 - 2$$

Proof. Let $u \in L_1$ and $v \in L_2$ such that

$$k = d_p(u, v) = d_p(u, L_2) = d_p(L_1, L_2) < \infty$$

and suppose $k > n_1 + n_2 - 2$. We write $u = pu'$ and $v = pv'$, with p be being the longest common prefix of u and v. Since $d_p(u, v) = \min_{w \in L_2}\{d_p(u, w)\}$, by Lemma 1, we have $|v'| \leq n_2 - 1$, which implies that $|u'| > n_1 - 1$.

By the Pumping Lemma [18], we can write $u' = xyz$ where $|yz| < n_1 - 1$, $|y| > 0$ and $pxy^iz \in L_1$ for any $i \in \mathbb{N}$. Consider a word $u_2 = pxy^2z \in L_1$ and let $v_2 \in L_2$ be a word such that $d_p(u_2, v_2) = d_p(u_2, L_2)$. That is, v_2 is the word in L_2 which is closest to u_2. By our assumption, we have $d_p(u_2, v_2) \leq k$. Now let q be the longest word such that $u_2 = qu_2'$ and $v_2 = qv_2'$.

First, suppose that $|q| \leq |pxy|$. Let $\ell = |pxy| - |q|$. Then,

$$k \leq d_p(u, v_2) = \ell + |z| + |v_2'| < \ell + |y| + |z| + |v_2'| = d_p(u_2, v_2) = k$$

which is a contradiction. Next, suppose that $|q| > |pxy|$. Recall that $u = pxyz$ and let q' be such that $v_2 = pxyq'v_2'$. By Lemma 1, let w be such that $|w| < n_2 - 1$ and $pxyw \in L_2$. Then we have

$$k \leq d_p(u, pxyw) = |z| + |w| \leq n_1 - 1 + n_2 - 1 < k$$

which again is a contradiction. Therefore, $k \leq n_1 + n_2 - 2$. □

Example 3. We will show that this bound is reachable. Let $\Sigma = \{a, b\}$ and let $L_1 = \Sigma^* a \Sigma^{n_1 - 2}$ and $L_2 = \Sigma^* b \Sigma^{n_2 - 2}$. Note that L_1 can be recognized by an NFA with n_1 states and L_2 can be recognized by an NFA with n_2 states. We observe that for any word in $w \in L_1$, we have $d_p(w, L_2) \leq n_1 + n_2 - 2$.

If the distance is bounded, then it is possible construct a neighbourhood of finite radius with respect to the given distance.

Lemma 4. *Let L_1 and L_2 be languages. Then $d_p(L_1|L_2) \leq k$ if and only if $L_1 \subseteq E(L_2, d_p, k)$.*

Theorem 5. *Let A_1 and A_2 be DFAs. Then it is decidable in polynomial time whether $d_p(L(A_1)|L(A_2))$ is bounded.*

Proof. By Theorem 2, we know that $d_p(L(A_1)|L(A_2)) \leq n_1 + n_2 - 2$ if it is bounded. Otherwise, it is unbounded. Therefore, it is enough to check

$$L(A_1) \subseteq E(L(A_2), d_p, n_1 + n_2 - 2).$$

It is known that the size of a DFA for $E(L(A_2), d_p, n_1 + n_2 - 2)$ is at most $\frac{n_2(n_2-1)}{2} + n_1 + n_2 - 1$ states, which is polynomial in n_2 and can be constructed in polynomial time [16]. Then since the inclusion problem for DFAs is decidable in polynomial time, checking the above inclusion can also be done in polynomial time. □

Theorem 6. *Let A_1 and A_2 be DFAs. Then $d_p(L(A_1)|L(A_2))$ is computable in polynomial time.*

We will now show that the same questions are PSPACE-complete when we are given nondeterministic finite automata. First, we will make use of the following observation.

Lemma 7. *Consider languages L_1 and L_2 over an alphabet Σ. Let $\#$ be a symbol not in Σ and $k \in \mathbb{N}$. Then*

$$d_p(L_1\#^k|L_2) \leq k \text{ iff } L_1 \subseteq L_2.$$

Theorem 8. *Let $k \in \mathbb{N}$ be fixed. For given NFAs A_1 and A_2, deciding whether or not $d_p(L(A_1)|L(A_2)) \leq k$ is PSPACE-complete.*

Proof. First, we note that given an n-state NFA A, we can construct an NFA A' for $E(L(A), d_p, k)$ with at most $n + k$ states [16]. To see that the problem is in PSPACE, we note that the problem is equivalent to deciding

$$L(A_1) \subseteq E(L(A_2), d_p, k).$$

To see that the problem is PSPACE-hard, we reduce from NFA universality. Suppose we are given an NFA A. Then by Lemma 7, $L(A) = \Sigma^*$ if and only if $d_p(\Sigma^*\#^k|L(A)) \leq k$. □

Corollary 9. *Let A_1 and A_2 be NFAs. Then the problem of deciding whether $d_p(L(A_1)|L(A_2))$ is bounded is PSPACE-complete.*

We can derive some results for the suffix distance as well, by using its symmetry with the prefix distance. One might assume that this means the complexity of questions regarding the relative suffix distance follow straightforwardly from our results on the relative prefix distance. However, computing the neighbourhood with respect to the suffix distance is much more difficult than for the prefix distance. First, as a corollary of the bound from Theorem 2, we get:

Corollary 10. *Let L_1, L_2 be regular languages recognized by NFAs A_1 and A_2 with n_1 and n_2 states, respectively. Then either $d_s(L_1|L_2) \leq n_1 + n_2 - 2$ or $d_s(L_1|L_2)$ is unbounded.*

Proposition 11. *Let A_1 and A_2 be DFAs with n_1 and n_2 states, respectively. Then deciding whether $d_s(L(A_1)|L(A_2))$ is bounded is in PSPACE.*

Proof. We can construct an NFA for the language $E(L(A_2), d_s, n_1 + n_2 - 2)$ that has at most $n_2 + (n_1 + n_2 - 2)$ states. Thus, we can decide the inclusion $L_1 \subseteq E(L_2, d_s, n_1 + n_2 - 2)$ in PSPACE. □

We note that the current best known DFA construction for $E(L(A_2), d_s, n_1+n_2-2)$ has at most $n_1 + 2^{n_2}$ states, and is therefore not known to be polynomial in n_2 [15].

Corollary 12. *Let A_1 and A_2 be NFAs. Then the problem of deciding whether $d_s(L(A_1)|L(A_2))$ is bounded is PSPACE-complete.*

4 Relative Prefix Distance and Context-free Languages

Here, we consider the relative distance on non-regular languages. The distance from one context-free language to another is undecidable by Choffrut and Pighizzini [8]. Thus, we consider the distance between context-free languages and regular languages. First, we define the following useful finite-state transducer.

Let $P_k = (Q_k, \Sigma, \Sigma, \delta_k, I, F_k)$ be a finite state transducer, shown in Figure 1, with $Q_k = \{0, \ldots, k\}$, $I = \{0\}$, $F_k = Q_k$, and transitions

- $(0, a, a, 0)$ for all $a \in \Sigma$,
- $(i, a, \varepsilon, i+1)$ for all $a \in \Sigma$ and $0 \leq i \leq k-1$,
- $(i, \varepsilon, a, i+1)$ for all $a \in \Sigma$ and $0 \leq i \leq k-1$,
- $(i, a, b, i+2)$ for all $a, b \in \Sigma$ with $a \neq b$ and $0 \leq i \leq k-2$.

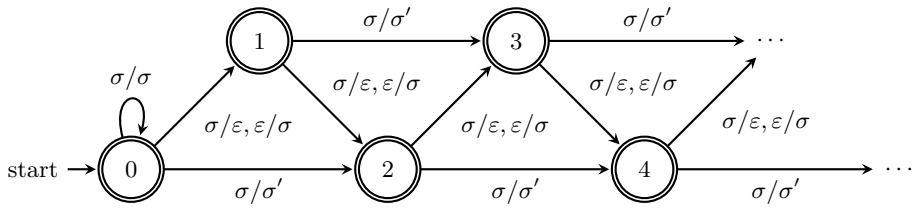

Fig. 1. The transducer P_k with $\sigma, \sigma' \in \Sigma$

Lemma 13. *Let $w \in \Sigma^*$. Then $P_k(w) = E(w, d_p, k)$.*

It is not difficult to see that for a language L, we have $P_k(L) = E(L, d_p, k)$. A similar transducer S_k with respect to the suffix distance can be defined by replacing the transition $(0, a, a, 0)$ for all $a \in \Sigma$ with (k, a, a, k) for all $a \in \Sigma^*$. We make use of the fact that context-free languages are closed under rational transductions to get the following result.

Proposition 14. *Let L be a context-free language. Then for every $k \geq 0$, the neighbourhood $E(L, d_p, k)$ is context-free.*

Proposition 15. *Let $k \in \mathbb{N}$ be fixed. Given a regular language L_1 and a context-free language L_2, determining whether or not $d_p(L_1|L_2) \leq k$ is undecidable.*

On the other hand, computing the relative distance from a context-free language to a regular language becomes decidable, and when the regular language is given by a DFA, this problem can even be decided in polynomial time. First, we will state the following useful fact from Chatterjee et al. [7].

Proposition 16 ([7]). *Given a PDA P and an NFA A, the inclusion $L(P) \subseteq L(A)$ can be decided in EXPTIME. Given a deterministic PDA P and an NFA A, it is EXPTIME-hard to decide whether or not $L(P) \subseteq L(A)$.*

Proposition 17. *Let $k \in \mathbb{N}$ be fixed.*

1. *Given an NFA A and a PDA P, deciding whether or not $d_p(L(P)|L(A)) \leq k$ is EXPTIME-complete.*
2. *Given a DFA B and a PDA P, it can be decided in polynomial time whether or not $d_p(L(P)|L(B)) \leq k$.*

It is clear that by symmetry, we would attain the same results for the suffix distance between a context-free language and a regular language. However, if we consider the distance between a DCFL and a regular language, we get different results for the prefix and suffix distance. First, we show that the neighbourhood of a DCFL with respect to the suffix distance need not be a DCFL.

Lemma 18. *There exist a deterministic context-free language L and integer k for which $E(L, d_s, k)$ is not a deterministic context-free language.*

This lemma, together with Proposition 15, leads to some fairly straightforward results.

Proposition 19. *Let $k \in \mathbb{N}$ be fixed. Given a DPDA P and an NFA A,*

1. *deciding whether or not $d_s(L(A)|L(P)) \leq k$ is undecidable.*
2. *deciding whether or not $d_s(L(P)|L(A)) \leq k$ is EXPTIME-complete.*

Differing from the case of the suffix distance, we show that neighbourhoods of DCFLs with respect to the prefix distance are also DCFLs.

Theorem 20. *Let L be a deterministic context-free language. Then for every $k \geq 0$, the neighbourhood $E(L, d_p, k)$ is a deterministic context-free language.*

Proof. Given a DPDA A recognizing L, we will construct a DPDA A' that recognizes the neighbourhood $E(L, d_p, k)$. We need to determine whether the input word w has prefix distance at most k from some word in L. We can simulate a computation of w on A and based on the current state and the top k symbols of the pushdown stack, we can determine the length of a path to or from a closest final state of A. If such a path of length less than k exists, then w has a prefix distance of less than k from some word in L. This requires that we know what the top k symbols of the pushdown stack are, so we simulate the top of the stack via the finite state memory and store the rest of the stack on the pushdown stack as normal.

Let L be recognized by a DPDA $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Then for each state $q \in Q$ and string of stack symbols $\pi \in \Gamma^{\leq k}$, we define the following function $\varphi_{A,k} : Q \times \Gamma^{\leq k} \to \mathbb{N}$,

$$\varphi_A(q, \pi) = \min_{w \in \Sigma^*} (\{|w| \mid (q, w, \pi) \vdash^* (q', \varepsilon, \pi')\} \cup \{k+1\})$$

for some $q' \in F$ and $\pi' \in \Gamma^*$. The function $\varphi_{A,k}(q, \pi)$ gives the length of a shortest word w such that for any word x that reaches the state q with π on the top of the stack, we have $xw \in L$ if $|w| \leq k$. Based on this function, we can construct a DPDA $A' = (Q', \Sigma, \Gamma, \delta', q_0', F')$ that recognizes the language $E(L, d_p, k)$.

Let $Q' = Q \times \Gamma^{\leq k} \times \{0, \ldots, k+1\} \cup \{p_1, \ldots, p_k\}$. We set the initial state to be $q_0' = (q_0, \varepsilon, \varphi_A(q_0, \varepsilon))$. The set of final states is defined

$$F' = Q \times \Gamma^{\leq k} \times \{0, \ldots, k\} \cup \{p_1, \ldots, p_k\}.$$

We describe the transition function of A' by first describing the operation of the stack. We keep track of the top k symbols of the stack "in memory" in the states and store the rest of the stack as normal. Consider a state q, a symbol $a \in \Sigma$, and let $(q', \beta) = \delta(q, a, \alpha)$, where α is the top of the pushdown stack of A and $\beta = \beta_1 \cdots \beta_{|\beta|}$ is the symbols to be pushed onto the pushdown stack of A. Let $\pi = \gamma_1 \cdots \gamma_{|\pi|}$ be the top of the pushdown stack with $|\pi| \leq k$ to be kept in memory.

Consider the state (q, π, i) in A' with $\gamma_1 = \alpha$. First, we consider $\beta = \varepsilon$ to demonstrate a pop action. First, we consider when $|\pi| \leq k$. This occurs when the size of the in-memory portion of the stack has at most k elements and therefore, the size of the entire stack has at most k elements. In this case, the stack of A' is empty and the stack operations are performed only on the in-memory portion of the stack. Thus, for a transition $\delta(q, a, \alpha)$ in A, we have the transition $\delta'((q, \pi, i), a, \varepsilon) = ((q', \gamma_2 \cdots \gamma_{|\pi|}, j), \varepsilon)$.

If $|\pi| > k$ and the stack contains $m > k$ symbols, then the top k symbols of the stack of A are stored as π and the rest of the $m - k$ stack symbols are on the pushdown stack of A'. In this case, A' will pop the topmost symbol α' on its stack to append to the end of π and remove the first symbol γ_1 of π to simulate a pop from the top of the in-memory portion of the stack. Formally, the transition is $\delta'((q, \pi, i), a, \alpha') = (q', \gamma_2 \cdots \gamma_k \alpha', j), \varepsilon)$, where α' is the top of the pushdown stack of A'.

Now, for $\beta \neq \varepsilon$, we demonstrate the push action. Let $\pi' = \beta \cdot \gamma_2 \cdots \gamma_{|\pi|}$. First we consider when $|\pi'| \leq k$. In this case, γ_1 is popped as above and now we need to push the symbols onto the stack. Since the size of the stack is less than k, we store the entire contents in memory and we have $\delta'((q, \pi, i), a, \varepsilon) = ((q', \pi', j), \varepsilon)$. If $|\pi'| > k$, then we keep the first k symbols in memory and push the rest onto the stack. Let $\pi' = \eta_1 \cdots \eta_{|\pi'|}$. Then we have the transition $\delta'((q, \pi, i), a, \alpha') = ((q', \eta_1 \cdots \eta_k, j), \eta_{k+1} \cdots \eta_{|\pi'|} \alpha')$, where α' is the top of the pushdown stack of A'.

To see that this is all deterministic, we recall that each transition of A is uniquely determined by the current state q, input symbol a, and the top of the stack α. Each of the above transitions of A' is still uniquely determined by the same items, noting that α is γ_1, the first symbol of π and that the stack additional stack manipulations are determined by π, which is part of the state, and α', the top of the pushdown stack of A'.

Now, we consider the step counter in the third component of a state of A'. The counter either increments by one for each input symbol that is read, or takes

on the value $\varphi_A(q, \pi)$ if it is smaller than the number of steps. Formally, for a transition $\delta'((q, \pi, i), a, \alpha) = ((q', \pi', j), \beta)$ of A', we define j by

$$j = \min(i + 1, \varphi_A(q', \pi')).$$

Finally, we consider transitions that were undefined in A. We define a chain of k new states p_i, $1 \leq i \leq k$. For each p_i and $1 \leq i \leq k - 1$, reading any symbol will transition to state p_{i+1} and there are no outgoing transitions from p_k. If on a state (q, π, j) with the top symbol of the pushdown stack α we ever read an input symbol a such that the transition $\delta(q, a, \alpha)$ is undefined, then based on the step counter component j, the machine A' enters the chain of k states at p_{j+1}.

Now we show that a word $w \in \Sigma^*$ reaches a state (q, π, i) if and only if there exists some word $x \in L$ such that $d_p(w, x) = i$ if $i \leq k$. First suppose that w reaches (q, π, i). Then this means w reaches the state q on the original machine. One of three cases is possible.

1. If $i \leq k$ and $\varphi_A(q, \pi) = i$, then there exists some suffix x' of length i such that $wx' \in L(A)$. This gives us $d_p(w, x) = d_p(w, wx') = |x'| = i$.
2. If $i \leq k$ and $\varphi_A(q, \pi) < i$, then on some prefix p of w, the first case applied. That is, for $w = pw'$, there exists a word $x = px'$ and $d_p(p, x) = |x'| = i - |w'|$. This implies that $d_p(w, x) = |x'| + |w'| = i$.
3. If $i > k$, then $i = k + 1$ and neither of the two cases above apply. Then there is no word x such that $d_p(w, x) \leq k$.

From the above, we observe that if w didn't reach a state in the original DPDA A, then, on some prefix p with $w = pw'$, reading p takes A' to the state (q, π, i). Then reading w' takes the machine to a state $p_{|w'|+i}$ if $i + |w'| \leq k$.

Since all states except for states of the form $(q, \pi, k + 1)$ are accepting states, we have $L(A') = E(L(A), d_p, k)$ and A' has $O(nk|\Gamma|^k)$ states. □

Recall from Proposition 15 that the relative prefix distance from a regular language to a context-free language is undecidable. We get contrasting results for DCFLs using the construction from Theorem 20 and the fact that DCFLs are closed under complement.

Proposition 21. *Let $k \in \mathbb{N}$ be fixed.*

1. *Given an NFA A and a DPDA P, it can be decided in polynomial time whether or not $d_p(L(A)|L(P)) \leq k$.*
2. *Given a DFA B and a DPDA P, it can be decided in polynomial time whether or not $d_p(L(B)|L(P)) \leq k$.*

Now, we consider the class of visibly pushdown languages. The class of VPLs is known to be a proper subclass of DCFLs. First, we show that the relative prefix distance between a DCFL and VPL is undecidable.

Proposition 22. *Let $k \in \mathbb{N}$ be fixed. Given a visibly pushdown automaton A and a deterministic pushdown automaton P,*

1. *determining whether or not* $d_p(L(A)|L(P)) \leq k$ *is undecidable.*
2. *determining whether or not* $d_p(L(P)|L(A)) \leq k$ *is undecidable.*

Unlike DCFLs, the VPLs are closed under typical language operations and the standard questions involving VPLs are decidable. This will allow us to consider the problem of deciding whether the prefix distance from a VPL to another VPL is within k. First, we will show that the prefix neighbourhood of a VPL is also a VPL.

Theorem 23. *Let L be a visibly pushdown language. Then $E(L, d_p, k)$ is a visibly pushdown language for all $k \geq 0$.*

Proof. Let A be a visibly pushdown automaton that recognizes L. We will modify the construction of the prefix neighbourhood DPDA defined in the proof of Theorem 20 to construct a VPA A' that recognizes $E(L, d_p, k)$. To preserve the visibly pushdown property, we must push and pop from the stack as dictated by call and return symbols. This is only an issue when the stack of A has size less than k. In the DCFL construction, we simply ignored the stack, but we cannot do this for a VPA.

To solve this, we add a new symbol $\triangle$ to the stack alphabet. Let $q \in Q$, $a \in \Sigma_c$, and let $(q', \beta) = \delta_c(q, a)$. Consider the state (q, π, i) with $\pi = \gamma_1 \cdots \gamma_{|\pi|}$ and $|\pi| < k$. On this transition, the VPA A must push β onto the stack. In the VPA A', we add β to the top of the stack in memory and push a dummy symbol $\triangle$ onto the stack. Then our transition in A' is $\delta'_c((q, \pi, i), a) = ((q', \beta\pi, j), \triangle)$.

Now let $a \in \Sigma_r$ be a return action and consider the transition $\delta_r(q, a, \alpha) = q'$ of A. The VPA A must pop α from the stack, but if $|\pi| < k$, the stack of A' contains only $\triangle$s. Then the corresponding pop action on A' is to pop a $\triangle$ off of the stack, use γ_1 to determine the transition, and remove γ_1 from the in-memory portion of the stack. Then our corresponding transition in A' is $\delta'_r((q, \gamma_2 \cdots \gamma_{|\pi|}, i), a, \triangle) = q'$. Since a $\triangle$ is only pushed onto the stack whenever a call action is read and popped whenever a return action is read, we are guaranteed to have exactly as many $\triangle$s as there are symbols in the in-memory portion of the stack.

Once the in-memory portion of the stack reaches k symbols, the VPA behaves exactly like the DPDA that was constructed in Theorem 20 until the stack size becomes less than k again. Furthermore, since the construction preserves the determinism of the DCFL, if the VPA A is deterministic, then the VPA A' that is constructed via this process will also be deterministic. □

Proposition 24. *Let $k \in \mathbb{N}$ be fixed. For given VPAs A_1 and A_2, deciding $d_p(L(A_1)|L(A_2)) \leq k$ is EXPTIME-complete.*

References

1. Alur, R., Madhusudan, P.: Adding nesting structure to words. Journal of the ACM 56(3) (2009)

2. Béal, M.P., Carton, O., Prieur, C., Sakarovitch, J.: Squaring transducers: an efficient procedure for deciding functionality and sequentiality. Theoretical Computer Science 292(1), 45–63 (2003)
3. Benedikt, M., Puppis, G., Riveros, C.: Bounded repairability of word languages. Journal of Computer and System Sciences 79(8), 1302–1321 (2013)
4. Benedikt, M., Puppis, G., Riveros, C.: The per-character cost of repairing word languages. Theoretical Computer Science 539, 38–67 (2014)
5. Bruschi, D., Pighizzini, G.: String Distances and Intrusion Detection: Bridging the Gap Between Formal Languages and Computer Security. RAIRO Informatique Théorique et Applications 40, 303–313 (2006)
6. Calude, C.S., Salomaa, K., Yu, S.: Additive Distances and Quasi-Distances Between Words. Journal of Universal Computer Science 8(2), 141–152 (2002)
7. Chatterjee, K., Henzinger, T.A., Ibsen-Jensen, R., Otop, J.: Edit Distance for Pushdown Automata. In: International Colloquium on Automata, Languages and Programming. pp. 121–133 (2015)
8. Choffrut, C., Pighizzini, G.: Distances between languages and reflexivity of relations. Theoretical Computer Science 286(1), 117–138 (2002)
9. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer Berlin Heidelberg (2009)
10. Han, Y.S., Ko, S.K.: Edit-Distance Between Visibly Pushdown Languages. In: SOFSEM 2017: Theory and Practice of Computer Science. pp. 387–401 (2017)
11. Han, Y.S., Ko, S.K., Salomaa, K.: The Edit-Distance Between a Regular Language and a Context-Free Language. International Journal of Foundations of Computer Science 24(07), 1067–1082 (2013)
12. Kutrib, M., Meckel, K., Wendlandt, M.: Parameterized Prefix Distance between Regular Languages. In: SOFSEM 2014: Theory and Practice of Computer Science. pp. 419–430 (2014)
13. Mohri, M.: Edit-distance of weighted automata: General definitions and algorithms. International Journal of Foundations of Computer Science 14(6), 957–982 (2003)
14. Ng, T.: Prefix Distance Between Regular Languages. In: Implementation and Application of Automata. pp. 224–235 (2016)
15. Ng, T., Rappaport, D., Salomaa, K.: Descriptional Complexity of Error Detection. In: Emergent Computation: A Festschrift for Selim G. Akl, pp. 101–119. Springer International Publishing (2017)
16. Ng, T., Rappaport, D., Salomaa, K.: State complexity of prefix distance. Theoretical Computer Science 679, 107–117 (2017)
17. Pighizzini, G.: How Hard Is Computing the Edit Distance? Information and Computation 165(1), 1–13 (2001)
18. Shallit, J.: A second course in formal languages and automata theory. Cambridge University Press, Cambridge, MA (2009)
19. Skala, M.: Counting distance permutations. Journal of Discrete Algorithms 7(1), 49–61 (2009)
20. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, chap. 2, pp. 41–110. Springer-Verlag, Berlin, Heidelberg (1997)