On the difference set of two transductions*

Stavros Konstantinidis[a,*], Nelma Moreira[b], Rogério Reis[b], Juraj Šebej[c]

[a] *Saint Mary's University, Mathematics & CS, 923 Robie Str, B3H 3C3, Halifax, Nova Scotia, Canada*
[b] *CMUP & DM, DCC, Faculdade de Ciências da Universidade do Porto, Rua do Campo Alegre, 4169-007, Porto, Portugal*
[c] *Institute of Computer Science, Faculty of Science, P. J. Šafárik University, Košice, Slovakia*

Abstract

The difference set $\Delta_{\boldsymbol{s},\boldsymbol{t}}$ of two (nondeterministic, in general) transducers $\boldsymbol{s},\boldsymbol{t}$ is the set of all input words for which the output sets of the two transducers are not equal. When the two transducers realize homomorphisms, their difference set is the complement of the well known equality set of the two homomorphisms. However, we show that transducer difference sets result in Chomsky-like classes of languages that are different than the classes resulting from equality sets. We also consider the following word problem: given transducers $\boldsymbol{s},\boldsymbol{t}$ and input w, tell whether the output sets $\boldsymbol{s}(w)$ and $\boldsymbol{t}(w)$ are different. In general the problem is **PSPACE**-complete, but it becomes **NP**-complete when at least one of the given transducers has finite outputs. We also provide a PRAX (polynomial randomized approximation) algorithm for the word problem as well as for the NFA (in)equivalence problem. Our presentation of PRAX algorithms improves the original presentation.

Keywords: transducer, difference set, equality set, counter machine, approximation algorithm, randomized algorithm, NFA equivalence

1. Introduction

We are interested in the difference set $\Delta_{S,T}$ between two transductions S and T that have the same domain:

$$\Delta_{S,T} = \{w \in \mathsf{dom}S \mid S(w) \neq T(w)\};$$

*Research supported by NSERC, Canada (Discovery Grant of S.K.) and by CMUP through FCT project UIDB/00144/2020.

*Corresponding author

Email addresses: s.konstantinidis@smu.ca (Stavros Konstantinidis), nelma.moreira@fc.up.pt (Nelma Moreira), rogerio.reis@fc.up.pt (Rogério Reis), juraj.sebej@upjs.sk (Juraj Šebej)

that is, the set of input words for which the outputs of the two transducers are different. We also write $\Delta_{\boldsymbol{s},\boldsymbol{t}}$, for $\Delta_{S,T}$ when $\boldsymbol{s},\boldsymbol{t}$ are transducers realizing S,T:

$$\Delta_{\boldsymbol{s},\boldsymbol{t}} = \{w \in \mathsf{dom}\boldsymbol{s} \mid \boldsymbol{s}(w) \neq \boldsymbol{t}(w)\}.$$

The theme of this research is a generalization of the work in [3], where the authors study the language that distinguishes two states of a deterministic finite automaton. The research in [3] is inspired by older studies on word experiments that distinguish certain aspects of automata states [7]. Here we consider the language that distinguishes the behaviour of two transducers. It is not difficult to see that this is equivalent to distinguishing the behaviour of two states of one transducer. Here moreover, we deal with transducers that are nondeterministic in general. When we consider the complementary notion of the equality set of two transductions (or transducers)

$$\mathcal{E}_{S,T} = \{w \in \mathsf{dom}S \mid S(w) = T(w)\}, \qquad \mathcal{E}_{\boldsymbol{s},\boldsymbol{t}} = \{w \in \mathsf{dom}\boldsymbol{s} \mid \boldsymbol{s}(w) = \boldsymbol{t}(w)\},$$

then we have a generalization of the classic notion of the equality set of two homomorphisms [28, 14], as well as the equality set of two deterministic generalized sequential machines or functional transducers[1] [5, 6, 4]. We note that any homomorphism $h : \Sigma^\star \to \Gamma^\star$ is total on $\Sigma^\star$, that is, the domain of h is $\Sigma^\star$. Some authors exclude the empty word from the equality set $\mathcal{E}_{g,h}$ of two homomorphisms h, g, as $\mathcal{E}_{g,h}$ is directly connected to the Post correspondence problem. Here, however, we do allow the empty word to be in $\mathcal{E}_{g,h}$ (which is also the approach taken in [28, 5]).

Our generalization from deterministic automata, or functional transductions, to transductions comes with a high price: *Membership to the difference (or equality) set of two given transducers $\boldsymbol{s},\boldsymbol{t}$ can be a hard problem; so we are interested in various ways to get as much information as possible about the difference set in question, considering also cases where the transducers involved are of a certain type.* In particular, we consider the following questions.

(In all questions, the transductions/transducers involved in a difference set are supposed to have the same domain.)

Deciding the word problem Δ: Given two transducers $\boldsymbol{s}$, $\boldsymbol{t}$ and a word w, decide whether $\boldsymbol{s}(w) \neq \boldsymbol{t}(w)$, that is, $w \in \Delta_{\boldsymbol{s},\boldsymbol{t}}$. As stated, this is the *unrestricted* word problem. We are also interested in restricted versions of the word problem when we have as a promise that the two given transducers are of certain types. For example, the *word problem for length-preserving transducers* is to decide whether $\boldsymbol{s}(w) \neq \boldsymbol{t}(w)$ when we know that for both $\boldsymbol{s},\boldsymbol{t}$, the length of any output word is equal to the length of the word that was used as input. About the word problem, we want to know how hard (or simple) it is: is it decidable in linear time, is it in the class **NP**, is it

[1] Reference [5] does consider the equality set of two transductions but their definition is different from ours.

in the class **PSPACE**? The answer depends on the restrictions on the transducers involved.

Chomsky-like type of the languages $\Delta_{S,T}$: For any fixed, but arbitrary, transductions S, T, we want to know the type of the language $\Delta_{S,T}$: is it regular, is it context-free and non-regular, is it non-context-free? The answer depends on the two transductions involved. And if $\Delta_{S,T}$ is a subset of some class $\mathcal{C}$, for all transductions S, T of certain types (e.g., when S, T are functional), is every language $L \in \mathcal{C}$ equal to the difference set $\Delta_{S,T}$ of two transductions S, T of the said types?

Remark 1. *The difference set of any two functional transductions is a one-counter language [4], while their equality set is a context-sensitive language—this follows from the result of [6] that the fixed point language* $\{w \in \mathsf{dom}S : w \in S(w)\}$ *of any transduction* S *is context-sensitive and the observation that the equality set of any two functional transductions* F, G *is equal to the fixed point of the transduction* $G^{-1} \circ F$*. We also note that there are functional transductions* F, G *whose equality set is not context-free [4].*

Structure of the paper and main results. The next section contains basic terminology and notation. Section 3 shows a few examples of difference sets and shows that the word problem can be hard (Theorem 1). Section 4 shows that there is a PRAX algorithm for the word problem (Theorem 2), as well as for the problem of NFA (in)equivalence. Our presentation of PRAX algorithms improves the original presentation in [19]. Section 5 shows that the difference set of two recognizable transduction is always regular and can be effectively constructed (Theorem 3). Section 6 shows a Chomsky-like hierarchy of classes of difference sets related to each other or to known ones (like the context-sensitive languages) (Theorem 4). Finally Section 7 contains a few concluding remarks.

2. Basic Terms and Background

We assume the reader to be familiar with basics of formal languages, see e.g., [15], [22], [24], [27]. Some notation: ε = empty word; Σ, Γ: arbitrary alphabets; $\bar{L}$ = the complement of the language L. We also assume the reader to be familiar with basics of transductions and transducers, see e.g., [2], [25], [36]. A (finite-state) transducer is a 6-tuple $\boldsymbol{t} = (Q, \Sigma, \Gamma, E, s, F)$ such that Q is the set of states, $s \in Q$ is the start (initial) state, $F \subseteq Q$ is the set of final states, Σ, Γ are the input and output alphabets, respectively, and E is the finite set of transitions (edges). Without mention, we assume that all transducers are in standard form: in each transition $(p, x/y, q) \in E$, we have that the input label x is empty or a single symbol, and the same for the output label y. The set of outputs of $\boldsymbol{t}$ on input w is denoted by $\boldsymbol{t}(w)$. The relation $\mathcal{R}(\boldsymbol{t})$ realized by $\boldsymbol{t}$ is the set $\{(w, z) : z \in \boldsymbol{t}(w)\}$. A transduction T is any relation realized by a transducer. We write $T(w)$ to denote the set of outputs of T on input w; hence, $T(w) = \boldsymbol{t}(w)$ when $T = \mathcal{R}(\boldsymbol{t})$.

Some *classes of transductions*:

- FINOUT: transductions T having finite outputs: the set $T(w)$ is finite for all inputs w.
- FINVAL: finite valued transductions T: there is $k \in \mathbb{N}_0$ such that the set $T(w)$ has at most k elements for all inputs w.
- FUNC: functional transductions T: the set $T(w)$ has at most one element for all inputs w.
- We have that FUNC $\subsetneq$ FINVAL $\subsetneq$ FINOUT.

We use the same terms for transducer types as for transduction classes; e.g., a transducer $\boldsymbol{t}$ has finite outputs if the transduction $\mathcal{R}(\boldsymbol{t})$ has finite outputs. We note that the term "T has finite outputs" is not standard. It is referred to as "T is simply finitely ambiguous" in [23] and "T is finitely ambiguous" in [29]. On the other hand, it is rather standard to use the term "ambiguous" in connection with the different paths followed by a transducer on a given input word w, as opposed to the different outputs produced on w [12].

Many "natural" (types of) transducers have finite outputs: any transducer for the set of prefixes (or suffixes) of a given input word; for all $d \in \mathbb{N}$, any transducer $\boldsymbol{t}_d$ realizing the up-to-d Hamming, or Levenshtein, distance ($z \in \boldsymbol{t}_d(w)$ iff the distance of w, z is $\leq d$); any transducer realizing the strict radix order; any length-preserving transducer; any exponentially ambiguous transducer = a transducer whose input part (the NFA made if we drop the output labels of the transducer) has $O(2^{\mathrm{poly}|w|})$ paths for each input word w. Observe that any exponentially ambiguous transducer has finite outputs. (See, e.g., [13, 21] for NFA ambiguity.)

A nondeterministic finite automaton (NFA), is a 5-tuple $\boldsymbol{n} = (Q, \Sigma, E, s, F)$, where the components are as in the case of a transducer, except that a transition of $\boldsymbol{n}$ is a tuple (p, x, q); that is, it has only an input label $x \in \Sigma \cup \{\boldsymbol{\varepsilon}\}$. As usual, $\mathcal{L}(\boldsymbol{n})$ is the language accepted by $\boldsymbol{n}$ = the set of all words formed in the paths of $\boldsymbol{n}$ from the start state s to a final state in F.

A (nondeterministic) one counter automaton (or machine) is a pushdown automaton where the pushdown alphabet has only one symbol plus a special bottom symbol [2]. We denote by **OCL** the class of languages accepted by one counter automata. As the pushdown can only store one symbol and can be tested for emptiness via the special stack bottom, the pushdown is called a counter. A (nondeterministic) counter machine with parameters (c, r) is an automaton with c counters such that each counter can do at most r reversals [17]. We denote by $\mathbf{NCM}(c, r)$ the class of languages accepted by counter machines with parameters (c, r), and by **NCM** the union of all $\mathbf{NCM}(c, r)$. We note that $\mathbf{OCL} \neq \mathbf{NCM}$: the language $\{a^n b^n : n \geq 1\}^\star$ is in **OCL** but not in **NCM**, [16], and the language $\{a^n b^n c^n : n \geq 0\}$ is in $\mathbf{NCM}(2, 1)$ but not in **OCL**.

Probability distributions. Let X be a countable nonempty set. A probability distribution on X is a function $D : X \to [0, 1]$ such that $\sum_{x \in X} D(x) = 1$. The domain of D, denoted by dom$D$, is the subset $\{x \in X : D(x) > 0\}$ of X. If $X = \{x_1, \ldots, x_\ell\}$, for some $\ell \in \mathbb{N}$, then we write

$$D = \big(D(x_1), \ldots, D(x_\ell)\big).$$

If $X \subseteq \mathbb{N}_0$ then the distribution D is called a length distribution. Following [11], we have the below definition.

Definition 1. Let D be a probability distribution on X. For any subset S of X, we define the quantity

$$D(S) = \sum_{x \in S} D(x) \tag{1}$$

and refer to it as the probability that a randomly selected element from D is in S. The notation $x \xleftarrow{\$} D$, borrowed from cryptography, means that x is randomly selected from D.

The Dirichlet distibution on $\mathbb{N}_0$, [10]. For any $t > 1$, the Dirichlet distribution D_t is defined such that $\mathsf{D}_t(n) = (1/\zeta(t))(n+1)^{-t}$ for $n \in \mathbb{N}_0$, where ζ is the Riemann zeta function. In [10] the author considers the Dirichlet distribution to be the basis where *"many heuristic probability arguments based on the fictitious uniform distribution on the positive integers become rigorous statements."*

Augmented length distributions, [19]. Selecting from a length distribution D could return a very large length ℓ, which can be intractable from an algorithmic point of view. For this reason we define the augmented length distribution D^M whose domain consists of all lengths $\ell \in \mathsf{dom}D$ with $\ell \leq M$ plus a *special new symbol '$\perp$'*, so the distribution could select the outcome '$\perp$' instead of a very large length. We have that

$$D^M(\ell) = D(\ell), \text{ if } \ell \leq M, \quad D^M(\perp) = D(\mathbb{N}^{>M}). \tag{2}$$

Word distributions. A word distribution W is a probability distribution on $\Sigma^\star$, that is, $W : \Sigma^\star \to [0,1]$ such that $\sum_{w \in \Sigma^\star} W(w) = 1$. The domain of W is $\mathsf{dom}W = \{w \in \Sigma^\star : W(w) > 0\}$.

Definition 2. Let D be a length distribution. Then $\langle D \rangle$ is the word distribution such that

$$\mathsf{dom}\langle D \rangle = \{w \in \Sigma^\star : |w| \in \mathsf{dom}D\} \quad \text{and} \quad \langle D \rangle(w) \;=\; D(|w|)|\Sigma|^{-|w|}.$$

Any such word distribution is called a length-based distribution.

For any length distribution D and for all $\ell \in \mathbb{N}_0$, we have: $\langle D \rangle(\Sigma^\ell) = D(\ell)$ and $\langle D \rangle(\Sigma^{>\ell}) = D\big(\mathbb{N}^{>\ell}\big)$.

Let W be a word distribution and let $M \in \mathbb{N}_0$. The augmented distribution W^M is defined in a natural way:

$$\begin{aligned}
\mathsf{dom}(W^M) &= \big(\mathsf{dom}(W) \cap \Sigma^{\leq M}\big) \cup \{\perp\}, \text{ or } \big(\mathsf{dom}(W) \cap \Sigma^{\leq M}\big) \text{ if } W(\Sigma^{>M}) = 0; \\
W^M(w) &= W(w), \quad \text{for all } w \in \mathsf{dom}(W) \cap \Sigma^{\leq M}; \\
W^M(\perp) &= W(\Sigma^{>M}) = 1 - W(\Sigma^{\leq M}).
\end{aligned}$$

3. Examples and Basic Results

In this section we first give a few examples that illustrate to some extent the types of the languages $\Delta_{S,T}$. Then we turn to the main result of this section, Theorem 1, where we show the upper bound **PSPACE** on the complexity of the (unrestricted) word problem Δ as well as the upper bound **NP** for $\Delta_{\text{fin}} =$ the version of the word problem where the first (at least) transducer has finite outputs. In fact, not surprisingly, Δ is **PSPACE**-hard, but Δ_{fin} is **NP**-hard.

Example 1. Let PX, SX be the prefix and suffix transductions—thus, $\mathsf{PX}(w) =$ the set of prefixes of w. Their difference set is equal to the set of all words containing at least two distinct letters. This follows when we note that, if a word w contains at least two distinct letters, then there is a prefix of w that is not a suffix of w. Thus, $\Delta_{\mathsf{PX},\mathsf{SX}}$ is a regular language: $\Delta_{\mathsf{PX},\mathsf{SX}} \in \mathbf{REG}$.

Example 2. Consider the finite valued transductions S, T with domain $a^\star b^\star a^\star b^\star$ such that $S(a^{n_1}b^{m_1}a^{m_2}b^{n_2}) = \{a^{n_1}, b^{m_1}\}$ and $T(a^{n_1}b^{m_1}a^{m_2}b^{n_2}) = \{a^{n_2}, b^{m_2}\}$. We have that $\Delta_{S,T} = a^\star b^\star a^\star b^\star \cap \{a^n b^m a^m b^n\}_{m,n\in\mathbb{N}_0}$ and $\mathcal{E}_{S,T} = \{a^n b^m a^m b^n\}_{m,n\in\mathbb{N}_0}$. The language $\mathcal{E}_{S,T}$ is context-free but not in **OCL** [35]. On the other hand, we have that $\Delta_{S,T}$ is in **OCL**, using the facts that

- $\Delta_{S,T}$ is the union of four languages: one of them consists of all words $a^{n_1}b^{m_1}a^{m_2}b^{n_2}$ with $n_1 > n_2$;
- the other three languages correspond to the three constraints $n_1 < n_2$, $m_1 > m_2$, $m_1 < m_2$;
- all four languages are in **OCL**; and **OCL** is closed under union.

Example 3. Consider the functional transductions S, T with domain $(a + b)^\star c(a + b)^\star$ such that $S(w_1cw_2) = \{w_1\}$ and $T(w_1cw_2) = \{w_2\}$. Then we have that $\Delta_{S,T} = (a+b)^\star c(a+b)^\star \cap \{wcw\}_{w\in(a+b)^\star}$ and $\mathcal{E}_{S,T} = \{wcw\}_{w\in(a+b)^\star}$. The language $\mathcal{E}_{S,T}$ is not context-free. On the other hand, we have that $\Delta_{S,T}$ is in **OCL** by Remark 1.

Example 4. Consider the finite valued transductions S, T with domain $a^+b^+c^+d^+$ such that $S(a^{n_1}b^{m_1}c^{n_2}d^{m_2}) = \{a^{n_1}, a^{m_1}\}$ and $T(a^{n_1}b^{m_1}c^{n_2}d^{m_2}) = \{a^{n_2}, a^{m_2}\}$. Then,

$$\Delta_{S,T} = \{a^{n_1}b^{m_1}c^{n_2}d^{m_2} \mid (n_1 \neq n_2 \wedge n_1 \neq m_2) \vee (m_1 \neq n_2 \wedge m_1 \neq m_2) \\ \vee (n_2 \neq n_1 \wedge n_2 \neq m_1) \vee (m_2 \neq n_1 \wedge m_2 \neq m_1)\},$$

which is a **NCM** language (shown in Theorem 4). On the other hand, the language is *not context-free:* this follows from the fact that the language is bounded (being a subset of $a^\star b^\star c^\star d^\star$) and that the Parikh map of the language is not a finite union of stratified linear sets [8, pg 160].

Example 5. The languages $\Delta_{S,T}$ are in **OCL**, in both of the following cases

- $S(a^{n_1}b^{m_1}c^{n_2}) = \{a^{n_1}, a^{m_1}\}$ and $T(a^{n_1}b^{m_1}c^{n_2}) = \{a^{n_2}\}$.

- $S(a^{n_1}b^{m_1}c^{n_2}) = \{a^{n_1}, a^{m_1}\}$ and $T(a^{n_1}b^{m_1}c^{n_2}) = \{a^{n_1}, a^{n_2}\}$.

In the first case, $\Delta_{S,T} = \{a^{n_1}b^{m_1}c^{n_2} \mid (n_1 = m_1 \wedge n_1 \neq n_2) \vee (n_1 \neq m_1)\} = \overline{\{a^n b^n c^n \mid n \geq 0\}} \cap a^\star b^\star c^\star$. The language is in **OCL** because one of the transductions is functional (see Theorem 4). In the second case, $\Delta_{S,T} = \{a^{n_1}b^{m_1}c^{n_2} \mid m_1 \neq n_2\}$.

Next we determine the complexity of the word problem in Theorem 1. In case the two given transducers realize homomorphisms, the word problem can be decided in deterministic logarithmic space (this is because our word problem is the complement of the word problem for the equality set of two homomorphisms which is in deterministic logarithmic space [14]). The proof of Theorem 1 uses the below lemma which is rather folklore, but we include it here for completeness.

Lemma 1. *The following statements hold true.*

1. *For any NFAs $\boldsymbol{n}_1, \boldsymbol{n}_2$, we have that $\mathcal{L}(\boldsymbol{n}_1) \subseteq \mathcal{L}(\boldsymbol{n}_2)$ iff $\mathcal{L}(\boldsymbol{n}_1) \cap \Sigma^{\leq 2^{s_1+s_2}} \subseteq \mathcal{L}(\boldsymbol{n}_2)$, where s_1, s_2 are the numbers of states of the two NFAs.*
2. *The problem of deciding whether $\mathcal{L}(\boldsymbol{n}_1) \subseteq \mathcal{L}(\boldsymbol{n}_2)$, for given NFAs $\boldsymbol{n}_1, \boldsymbol{n}_2$, is in* **PSPACE**.

Proof. For the first statement, first note that there are DFAs $\boldsymbol{d}_1, \boldsymbol{d}_2$ having at most $2^{s_1}, 2^{s_2}$ states, which are equivalent to $\boldsymbol{n}_1, \boldsymbol{n}_2$. Consider the product DFA $\boldsymbol{d}_1 \cap \overline{\boldsymbol{d}_2}$, which has at most $2^{s_1+s_2}$ states and accepts $\mathcal{L}(\boldsymbol{n}_1) \cap \overline{\mathcal{L}(\boldsymbol{n}_2)}$. Suppose that $\mathcal{L}(\boldsymbol{n}_1) \cap \Sigma^{\leq 2^{s_1+s_2}} \subseteq \mathcal{L}(\boldsymbol{n}_2)$, but there is a minimal length word $w \in \mathcal{L}(\boldsymbol{n}_1) - \mathcal{L}(\boldsymbol{n}_2)$. Then w has length $> 2^{s_1+s_2}$ and the accepting path of $\boldsymbol{d}_1 \cap \overline{\boldsymbol{d}_2}$ with label w has a cycle. If we remove the cycle, we get a shorter accepting path with some label $w' \in \mathcal{L}(\boldsymbol{n}_1) - \mathcal{L}(\boldsymbol{n}_2)$, which is impossible. Hence, $\mathcal{L}(\boldsymbol{n}_1) \subseteq \mathcal{L}(\boldsymbol{n}_2)$. The second statement follows by combining the results of [31, 30, 32, 18]. However, we can also show it directly using the first statement and the following polynomial space nondeterministic algorithm that decides whether $\mathcal{L}(\boldsymbol{n}_1) \not\subseteq \mathcal{L}(\boldsymbol{n}_2)$: initialize the set variables $V_1 = \{p_0\}$ and $V_2 = \{q_0\}$, where p_0, q_0 are the initial states of $\boldsymbol{n}_1, \boldsymbol{n}_2$. Guess up to $2^{s_1+s_2}$ alphabet symbols; for each symbol σ_i guessed, compute the next values of V_1 and V_2, which are the next sets of states of $\boldsymbol{n}_1, \boldsymbol{n}_2$ when the input σ_i is consumed. After the last symbol σ_ℓ is processed, return Yes iff V_1 contains a final state of $\boldsymbol{n}_1$ and V_2 contains no final state of $\boldsymbol{n}_2$—thus, the algorithm decides whether a word $\sigma_1 \cdots \sigma_\ell \in \mathcal{L}(\boldsymbol{n}_1) - \mathcal{L}(\boldsymbol{n}_2)$. The decidability of $\mathcal{L}(\boldsymbol{n}_1) \subseteq \mathcal{L}(\boldsymbol{n}_2)$ in polynomial space follows from the fact that **PSPACE** is closed under complementation.

Theorem 1. *The following statements hold true.*

1. *The word problem Δ is* **PSPACE**-*complete.*
2. *The word problem Δ_{fin} (where the first, at least, transducer has finite outputs) is* **NP**-*complete.*

Proof. First statement: The word problem Δ is to decide whether $\boldsymbol{s}(w) \neq \boldsymbol{t}(w)$, given transducers $\boldsymbol{s}, \boldsymbol{t}$ and word w. The problem is **PSPACE**-hard because we can reduce to it the NFA universality problem: given NFA $\boldsymbol{n}$ over

some alphabet Σ, decide whether $\mathcal{L}(\boldsymbol{n}) = \Sigma^\star$. Indeed, we have that $\mathcal{L}(\boldsymbol{n}) = \Sigma^\star$ iff $\boldsymbol{s}(w) \neq \boldsymbol{t}(w)$, where $\boldsymbol{s}, \boldsymbol{t}, w$ are constructed in polynomial time as follows: $\boldsymbol{s}$ realizes $\{(w, x) : x \in \mathcal{L}(\boldsymbol{n})\}$, $\boldsymbol{t}$ realizes $\{w\} \times \Sigma^\star$, and w is any chosen word over Σ. Now we show that the word problem is in the class **PSPACE**. First compute NFAs accepting $\boldsymbol{s}(w)$ and $\boldsymbol{t}(w)$. These NFAs are of sizes $O(|\boldsymbol{s}||w|)$ and $O(|\boldsymbol{t}||w|)$. Then decide within polynomial space whether these NFAs are equivalent—see Lemma 1.

Second statement: The word problem Δ_{fin} is **NP**-hard because we can reduce to it the complement of the following **coNP**-complete problem: given a block NFA $\boldsymbol{b}$, that is an NFA accepting fixed-length words of some length ℓ, decide whether $\mathcal{L}(\boldsymbol{b}) = \Sigma^\ell$, [19]. Indeed, for any block NFA $\boldsymbol{b}$, we have that $\mathcal{L}(\boldsymbol{b}) \neq \Sigma^\ell$ iff $\boldsymbol{s}(w) \neq \boldsymbol{t}(w)$, where $\boldsymbol{s}, \boldsymbol{t}, w$ are constructed in polynomial time as follows: $\boldsymbol{s}$ realizes $\{(w, x) : x \in \mathcal{L}(\boldsymbol{b})\}$, $\boldsymbol{t}$ realizes $\{w\} \times \Sigma^\ell$, and w is any chosen word in Σ^ℓ. We now show that Δ_{fin} is in **NP**. Given instance $\boldsymbol{s}, \boldsymbol{t}, w$, where we know that $\boldsymbol{s}$ has finite outputs, we describe a nondeterministic polynomial time algorithm deciding whether $w \in \Delta_{\boldsymbol{s},\boldsymbol{t}}$.

1. construct NFAs $\boldsymbol{m}$, $\boldsymbol{n}$ accepting $\boldsymbol{s}(w)$, $\boldsymbol{t}(w)$;
2. let n be the number of states of $\boldsymbol{m}$; // any word in $\mathcal{L}(\boldsymbol{m})$ has length $< n$
3. construct DFA $\boldsymbol{d}$ accepting all words of length $\geq n$;
4. construct NFA $(\boldsymbol{n} \cap \boldsymbol{d})$ accepting all words in $\boldsymbol{t}(w)$ that are of length $\geq n$;
5. if $(\boldsymbol{n} \cap \boldsymbol{d})$ accepts at least one word return Yes;
   // next test whether there is a word in $\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)$ that is shorter than n
6. guess a word z of length $< n$;
7. if $\big(z \in \boldsymbol{s}(w)$ and $z \notin \boldsymbol{t}(w)\big)$ or $\big(z \notin \boldsymbol{s}(w)$ and $z \in \boldsymbol{t}(w)\big)$ return Yes;
8. return No

All operations in the above algorithm can be done in polynomial time. Any word in $\boldsymbol{s}(w)$ cannot be longer than $n-1$, so steps 1–5 decide deterministically whether there is a word in $\boldsymbol{t}(w)$ that is too long to be in $\boldsymbol{s}(w)$. Steps 6–8 use nondeterminism to decide whether $\boldsymbol{s}(w) \triangle \boldsymbol{t}(w) \neq \emptyset$, knowing that any word in $\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)$ must be of length $< n$.

Remark 2. *In the proof of the claim that Δ_{fin} is* **NP***-hard, both transducers $\boldsymbol{s}, \boldsymbol{t}$ are length preserving, that is, $|z| = |w|$ for all $z \in \boldsymbol{s}(w)$, and the same for $\boldsymbol{t}$. Hence, the restriction of Δ_{fin} to length preserving transducers does not make the word problem easier.*

Remark 3. *In [34], the author shows that there is a double exponential algorithm that computes, for any given finite valued transducer $\boldsymbol{s}$, a set $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N$ of functional transducers such that $\mathcal{R}(\boldsymbol{s}) = \cup \mathcal{R}(\boldsymbol{f}_i)$. The time complexity of this problem is reduced to single exponential in [26]. This result can be used to decide in exponential time the version of the word problem restricted to finite valued transducers. However, the nondeterministic algorithm in the proof of Theorem 1 is applicable to the proper superclass of transducers with finite outputs and entails an exponential time deterministic algorithm.*

4. PRAX Algorithms & the PRAX Algorithm for Δ

As the word problem Δ is hard, in Theorem 2 of this section we provide a polynomial time randomized approximation (PRAX) algorithm for Δ. We adapt the PRAX method introduced in [19] which applies to hard NFA universality problems. In fact in Lemma 2, we make more clear the concept of PRAX algorithms so that they also apply to the complements of the problems considered in [19].

The PRAX method of [19]. Let v be a $[0,1]$-valued function, that is a function that maps each problem instance[2] x to a value in $[0,1]$. Define the language

$$L_v = \{x : v(x) = 1\}.$$

For the NFA universality problem (whether $\mathcal{L}(\boldsymbol{n}) = \Sigma^\star$ for given NFA $\boldsymbol{n}$), we have $v(\boldsymbol{n}) = W(\mathcal{L}(\boldsymbol{n}))$, where W is any word distribution with domain $\Sigma^\star$. Indeed we have that $\mathcal{L}(\boldsymbol{n}) = \Sigma^\star$ iff $W(\mathcal{L}(\boldsymbol{n})) = 1$. Each real $\varepsilon \in (0,1)$ defines the approximation language

$$L_{v,\varepsilon} = \{x : v(x) \geq 1 - \varepsilon\}.$$

The idea here is that, as it is hard to tell whether $v(x) = 1$, we might be *happy to know that* $v(x) \geq 1-\varepsilon$, where ε is called the (approximation) tolerance. As $L_{v,\varepsilon}$ can be harder than L_v, [19] defines a PRAX algorithm for L_v to be a randomized decision algorithm $A(x,\varepsilon)$ satisfying the following conditions:

- if $x \in L_v$ then $A(x,\varepsilon) = \mathsf{True}$;
- if $x \notin L_{v,\varepsilon}$ then $\mathrm{P}[A(x,\varepsilon) = \mathsf{False}] \geq 3/4$;
- $A(x,\varepsilon)$ works within polynomial time w.r.t. $1/\varepsilon$ and the size of x.

When $A(x,\varepsilon)$ gives the answer False, this answer is correct: $x \notin L_v$. If $A(x,\varepsilon)$ returns True then probably $x \in L_{v,\varepsilon}$, in the sense that $x \notin L_{v,\varepsilon}$ would imply $\mathrm{P}[A(x,\varepsilon) = \mathsf{False}] \geq 3/4$. Thus, when the algorithm returns True, the answer is *correct within the tolerance* ε ($x \in L_{v,\varepsilon}$) with probability $\geq 3/4$. The algorithm returns the wrong answer exactly when it returns True and $x \notin L_{v,\varepsilon}$, but this happens with probability $\leq 1/4$.

The PRAX method for both 0-1 and non-0-1 problems. Let again v be a $[0,1]$-valued function. We denote by $\bar{v}$ the $[0,1]$-valued function with $\bar{v}(x) = 1 - v(x)$. While the method of [19] seems to apply only to universality problems, we see that the language L_v is also equal to $\{x : \bar{v}(x) = 0\}$. Thus, we call the language L_v a 0-1 problem. On the other hand, for given v, we define the non-0-1 problem to be the language

$$K_v = \{x : v(x) > 0\},$$

[2] Following the presentation style of [9, pg 193] and [19], we refrain from cluttering the notation with the use of a variable for the set of instances.

which is also equal to $\{x : \bar{v}(x) < 1\}$. Our word problem Δ can be written as

$$\Delta = \{\boldsymbol{s}, \boldsymbol{t}, w : \big(\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)\big) \neq \emptyset\} \;=\; K_v \;=\; \{\boldsymbol{s}, \boldsymbol{t}, w : W\big(\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)\big) > 0\},$$

where we use the value function $v(\boldsymbol{s}, \boldsymbol{t}, w) = W\big(\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)\big)$. As before, each tolerance $\varepsilon \in (0, 1)$ defines an approximation language

$$K_{v,\varepsilon} = \{x : v(x) > \varepsilon\}.$$

Thus, Δ_ε consists of instances for which the symmetric difference of $\boldsymbol{s}(w)$ and $\boldsymbol{t}(w)$ is significant and should be detected by a randomized algorithm with high probability.

Definition 3. Let v be a $[0, 1]$-valued function. A PRAX algorithm for K_v is a randomized decision algorithm $A(x, \varepsilon)$ such that

1. If $x \notin K_v$ then $A(x, \varepsilon) = \mathsf{False}$.
2. If $x \in K_{v,\varepsilon}$ then $\mathrm{P}[A(x, \varepsilon) = \mathsf{True}] \geq 3/4$.
3. $A(x, \varepsilon)$ works within polynomial time w.r.t. $1/\varepsilon$ and the size of x.

A PRAX algorithm is a randomized algorithm which is a PRAX for a 0-1 or a non-0-1 problem.

Explanation. In the above definition, if $A(x, \varepsilon)$ returns True then $x \in K_v$. If $A(x, \varepsilon)$ returns False then probably $x \notin K_{v,\varepsilon}$, in the sense that $x \in K_{v,\varepsilon}$ would imply $\mathrm{P}[A(x, \varepsilon) = \mathsf{True}] \geq 3/4$. Thus, whenever the algorithm returns the answer True, this answer is correct: $x \in K_v$; when the algorithm returns False, the answer is *correct within the tolerance* ε ($x \notin K_{v,\varepsilon}$) with probability $\geq 3/4$. The algorithm returns the wrong answer exactly when it returns False and $x \in K_{v,\varepsilon}$, but this happens with probability $< 1/4$.

How are PRAX algorithms for 0-1 and for non-0-1 problems related to each other? Their intuitive duality can be formalized in the following result whose proof follows from the definitions without complications.

Lemma 2. [PRAX duality.] *For any decision algorithm $A(\cdots)$ we denote by $\bar{A}(\cdots)$ the algorithm that results by simply negating all decisions (truth outputs) made by A. Let v be a $[0, 1]$-valued function. We have that $A(x, \varepsilon)$ is a PRAX algorithm for K_v iff $\bar{A}(x, \varepsilon)$ is a PRAX algorithm for $L_{\bar{v}}$.*

We now turn to the PRAX algorithm for the word problem (Theorem 2). The following lemma is analogous to Lemma 4 of [19]. However, we note that the proof of the present lemma is simpler and the upper bound is smaller than that of [19]. We also recall from [19] that the application of the Chebyshev inequality to a binomial random variable B entails the following inequality for $a > 0$.

$$\mathrm{P}[\,|B - E(B)| \geq a] \;\leq\; n/(4a^2). \tag{3}$$

$\mathsf{EstSetSize}(x, n, M)$

cnt := 0;
repeat n times:
  $z \xleftarrow{\$} W^M$;
  if $\big(z \neq \perp$ and $z \in S(x)\big)$
    cnt := cnt+1;
return cnt / n;

Note: If the domain of the word distribution W is finite and its words are of length $\leq M$, then we can simply use $z \xleftarrow{\$} W$ instead of $z \xleftarrow{\$} W^M$ and we can omit the condition $z \neq \perp$.

Figure 1: This random process refers to a particular word distribution W. It is assumed that each input x describes a language $S(x)$ that can be infinite—e.g., x can be an NFA and $S(x)$ would be the language accepted by x; or x can be an instance $(\boldsymbol{s}, \boldsymbol{t}, w)$ of our word problem Δ and $S(x)$ would be $\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)$. The returned value is an estimate of the "size" of $S(x)$ w.r.t. $\mathsf{dom}W$, or mathematically an estimate of the probability that a word selected from W is in $S(x)$—see Lemma 3.

Lemma 3. *Consider the random process in Fig. 1, and let* $\mathtt{Cnt}$ *be the random variable for the value of* cnt *when the process returns. Let* $\delta, q \in [0, 1]$. *If* $q < \delta$ *and* $W(S(x)) > \delta + W(\Sigma^{>M})$ *then* $\mathrm{P}[\mathtt{Cnt}/n \leq q] < \frac{1}{4n(\delta - q)^2}$.

Proof. Assume $q < \delta$ and $W(S(x)) > \delta + W(\Sigma^{>M})$. Let $S^M = S(x) \cap \Sigma^{\leq M}$. First note that each selection z is either a word in $\mathsf{dom}W$ of length $\leq M$ or $\perp$. Thus, $\mathtt{Cnt}$ is binomial: the number of successes = "selections in S^M" in n trials. Hence, $E(\mathtt{Cnt}) = nW(S^M)$. Thus, we have

$$\begin{aligned} \mathrm{P}[\mathtt{Cnt} \leq nq] &= \mathrm{P}[\mathtt{Cnt} - E(\mathtt{Cnt}) \leq nq - nW(S^M)] \\ &\leq \mathrm{P}[\,|\mathtt{Cnt} - E(\mathtt{Cnt})| \geq nW(S^M) - nq\,] \\ &\leq \frac{1}{4n\big(W(S^M) - q\big)^2} < \frac{1}{4n(\delta - q)^2}, \end{aligned}$$

where we note that $W(S^M) = W(S(x)) - W\big(S(x) \cap \Sigma^{>M}\big) \geq W(S(x)) - W(\Sigma^{>M}) > \delta$.

Theorem 2. $\mathsf{DiffSet}\big(\boldsymbol{s}, \boldsymbol{t}, w, \varepsilon\big)$ *in Fig. 2 is a PRAX algorithm, with respect to the Dirichlet word distribution, for the word problem* Δ.

Proof. For brevity, we use $A(\alpha, \varepsilon)$ to refer to $\mathsf{DiffSet}\big(\boldsymbol{s}, \boldsymbol{t}, w, \varepsilon\big)$. The algorithm constructs NFAs $\boldsymbol{m}, \boldsymbol{n}$ accepting $\boldsymbol{s}(w), \boldsymbol{t}(w)$ and selects n elements from D_t^M, where M is such that $\mathsf{D}_t(\Sigma^{>M}) \leq \varepsilon/2$—Lemma 6 of [19] says that $\mathsf{D}_t(\Sigma^{>M}) \leq \delta$, if $M \geq \sqrt[t-1]{1/\delta}$. Each selection ℓ is either $\perp$ (corresponding to a word length that would be too large), or a word length $\ell \leq M$. In the latter case, a word of length ℓ is selected uniformly at random. Next we need to verify the three conditions about $A(\alpha, \varepsilon)$ in Definition 3. If $\alpha \notin \Delta$ then $\boldsymbol{s}(w) \triangle \boldsymbol{t}(w) = \emptyset$, so

$\mathsf{DiffSet}\big(\boldsymbol{s}, \boldsymbol{t}, w, \varepsilon\big)$
  compute $\boldsymbol{m} :=$ NFA accepting $\boldsymbol{s}(w)$;
  compute $\boldsymbol{n} :=$ NFA accepting $\boldsymbol{t}(w)$;
  $n := \lceil 4/\varepsilon^2 \rceil$;
  $M := \lceil \sqrt[t-1]{2/\varepsilon} \rceil$;
  $D := \big(\mathsf{D}_t(0), \ldots, \mathsf{D}_t(M), 1 - \sum_{\ell=0}^{M} \mathsf{D}_t(\ell)\big)$;
  repeat n times:
    $\ell :=$ $\mathsf{selectFin}(D)$;
    if $(\ell \neq \perp)$ $z :=$ $\mathsf{selectUnif}(\Sigma, \ell)$;
    if $\big(\ell \neq \perp$ and $z \in \mathcal{L}(\boldsymbol{m}) \triangle \mathcal{L}(\boldsymbol{n})\big)$
      return True;
  return False;

Figure 2: This is the PRAX algorithm for the word problem Δ—see Theorem 2. The word distribution used is $\langle \mathsf{D}_t \rangle$, that is the distribution based on the Dirichlet length distribution D_t, for some $t > 1$. The function $\mathsf{selectFin}(D)$ selects an element from the finite distribution $D = \mathsf{D}_t^M$. The function $\mathsf{selectUnif}(\Sigma, \ell)$ selects uniformly a word of length ℓ over Σ.

the algorithm will return False. For the second condition, assume $\alpha \in \Delta_\varepsilon$; then $\langle \mathsf{D}_t \rangle\big(\boldsymbol{s}(w) \triangle \boldsymbol{t}(w)\big) > \varepsilon$. Consider the random process in Fig. 1 and assume that it selects exactly the same words z as $A(\alpha, \varepsilon)$ does. Using Lemma 3 for $\delta = \varepsilon/2$ and $q = 0$, we have

$$\mathrm{P}[A(\alpha, \varepsilon) = \mathsf{False}] = \mathrm{P}[\mathtt{Cnt} = 0] = \mathrm{P}[\mathtt{Cnt}/n \leq 0] < \frac{1}{4n\delta^2} \leq \frac{1}{4}.$$

Hence, $\mathrm{P}[A(\alpha, \varepsilon) = \mathsf{True}] > 3/4$, as required. The third condition requires that $A(\alpha, \varepsilon)$ works in polynomial time. This follows from standard automata constructions and the fact that $\mathsf{selectFin}(D)$ and $\mathsf{selectUnif}(\Sigma, \ell)$ can also be done in polynomial time, [19].

The NFA inequivalence problem is to decide, for given NFAs $\boldsymbol{m}, \boldsymbol{n}$, whether $\mathcal{L}(\boldsymbol{m}) \neq \mathcal{L}(\boldsymbol{n})$, which is equivalent to $\big(\mathcal{L}(\boldsymbol{m}) \triangle \mathcal{L}(\boldsymbol{n})\big) \neq \emptyset$, and also equivalent to $\langle \mathsf{D}_t \rangle\big(\mathcal{L}(\boldsymbol{m}) \triangle \mathcal{L}(\boldsymbol{n})\big) > 0$. This problem is **PSPACE**-complete. Clearly, if we omit the first two lines from the PRAX algorithm $\mathsf{DiffSet}\big(\boldsymbol{s}, \boldsymbol{t}, w, \varepsilon\big)$ we get a PRAX algorithm $\mathsf{IneqNFA}(\boldsymbol{m}, \boldsymbol{n})$ for the NFA inequivalence problem. Moreover, by Lemma 2 (PRAX duality) we have that $\overline{\mathsf{IneqNFA}}(\boldsymbol{m}, \boldsymbol{n})$ is a PRAX algorithm for the NFA equivalence problem.

Corollary 1. *There are PRAX algorithms, with respect to the Dirichlet word distribution, for both, the NFA inequivalence and the NFA equivalence problems.*

5. Difference Sets of Recognizable transductions

A nonempty transduction T is called recognizable, if it is a finite union of cross products of regular languages, that is,

$$T = \bigcup_{i=1}^{n} A_i \times B_i, \tag{4}$$

where $n \geq 1$ and all A_i's and B_i's are regular languages. *We assume that, unless T is empty and unless stated otherwise, all A_i's and all B_j's are nonempty.* A natural representation of recognizable transductions is as follows. An NFA pair set is a set

$$\boldsymbol{A} = \{(\boldsymbol{a}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{a}_n, \boldsymbol{b}_n)\}, \tag{5}$$

where the $\boldsymbol{a}_i$'s and $\boldsymbol{b}_i$'s are NFAs. If $A_i = \mathcal{L}(\boldsymbol{a}_i)$ and $B_i = \mathcal{L}(\boldsymbol{b}_i)$, for all i, then we write $\mathcal{R}(\boldsymbol{A}) = T$ and we say that $\boldsymbol{A}$ describes (or represents) T. If for all i, the languages $A_i = \mathcal{L}(\boldsymbol{a}_i)$ are nonempty and mutually disjoint and the languages $B_i = \mathcal{L}(\boldsymbol{b}_i)$ are nonempty and distinct then the expression in (4) is said to be in disjoint canonical form, in which case any NFA pair set $\boldsymbol{A}$ that describes T is also said to be in disjoint canonical form. It turns out that every recognizable transduction T can be written as in (4) in disjoint canonical form: [25, Exercise IV.1.22], [20]. Below in Lemma 4, we provide an explicit construction of this fact which shows that the disjoint canonical form of T is unique and how large it can be. The main result here is that the difference set of two recognizable transductions is a regular language and can be effectively constructed:

Theorem 3. *Let $S = \bigcup_{i=1}^{n} A_i \times B_i$ and $T = \bigcup_{j=1}^{m} C_j \times D_j$ be recognizable transductions with the same domains. The following statements hold true.*

1. *The difference set $\Delta_{S,T}$ is a regular language.*
2. *If S, T are given via NFA pair sets then an NFA accepting $\Delta_{S,T}$ can be effectively constructed.*
3. *If S, T are given via NFA pair sets $\boldsymbol{A} = \{(\boldsymbol{a}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{a}_n, \boldsymbol{b}_n)\}$ and $\boldsymbol{C} = \{(\boldsymbol{c}_1, \boldsymbol{d}_1), \ldots, (\boldsymbol{c}_m, \boldsymbol{d}_m)\}$ in disjoint canonical form then there is an NFA of size $O\big(\sum |\boldsymbol{a}_i| \cdot \sum |\boldsymbol{c}_j|\big)$ accepting $\Delta_{S,T}$. Moreover, there are NFA pair sets $\boldsymbol{A}$ and $\boldsymbol{C}$ as above such that the constructed NFA for $\Delta_{S,T}$ is of size $\Theta\big(\sum |\boldsymbol{a}_i| \cdot \sum |\boldsymbol{c}_j|\big)$.*

The proof is shown further below and uses the fact that it is always possible to express a recognizable T as in (4) in disjoint canonical form.

Lemma 4. *Let $T = \bigcup_{i=1}^{n} A_i \times B_i$ be a recognizable transduction such that none of the languages A_i, B_i is empty. The following statements hold true.*

1. *There is a recognizable transduction $R = \bigcup_{j=1}^{m} E_j \times F_j$ in disjoint canonical form such that $m \leq 2^n - 1$.*
2. *If T is given by an NFA pair set then we can construct an NFA pair set describing R.*

3. *The disjoint canonical form is unique: if* $R = \bigcup_{\ell=1}^{h} G_\ell \times H_\ell$ *in disjoint canonical form then we have that* $\ell = m$ *and the set of pairs* (E_j, F_j) *is equal to the set of pairs* (G_ℓ, H_ℓ).

PROOF. We prove each statement in turn.

1. Let $N = \{1, \ldots, n\}$. For any word $w \in \cup A_i$, we have the following mutually exclusive cases:
   - w belongs to all n of the A_i's
   - w belongs to exactly $n-1$ of the A_i's: $\binom{n}{n-1}$ cases
   - ......
   - w belongs to exactly k of the A_i's: $\binom{n}{k}$ cases
   - ......
   - w belongs to exactly 1 of the A_i's: $\binom{n}{1}$ cases.

   Based on the above n mutually exclusive cases for a $w \in \cup A_i$, we can now define the required E_j's and F_j's in n *steps* as follows: In step 1, if $\cap_{i\in N} A_i \neq \emptyset$ then $E_1 = \cap_{i\in N} A_i$ and $F_1 = \cup_{i\in N} B_i$. In the general step k, the next group of E_j's are the nonempty sets of the form $(\cap_{i\in I} A_i) - (\cup_{\ell\in N-I} A_\ell)$, for each choice of an $I \subseteq N$ with $|I| = k$, and the corresponding F_j's are the languages $\cup_{i\in I} B_i$. For example, if $n = 4$ then step 3 would define the next nonempty sets E_j from the list: $(A_1 \cap A_2) - (A_3 \cup A_4)$, $(A_1 \cap A_3) - (A_2 \cup A_4)$, $(A_1 \cap A_4) - (A_2 \cup A_3)$, $(A_2 \cap A_3) - (A_1 \cup A_4)$, $(A_2 \cap A_4) - (A_1 \cup A_3)$, $(A_3 \cap A_4) - (A_1 \cup A_2)$.
2. If T is given by an NFA pair set then also R can be described by an NFA pair set, as the above definition of the sets E_j, F_j involves regularity preserving operations and the efficient test for emptiness on NFAs.
3. Now suppose that R can also be written in disjoint canonical form as $R = \bigcup_{\ell=1}^{h} G_\ell \times H_\ell$ such that $\ell \leq m$. If $\ell < m$ then there are disjoint languages E_{j_1} and E_{j_2} and two elements $w_1 \in E_{j_1}, w_2 \in E_{j_2}$ that must belong to the same language G_ℓ. This is impossible, however, as the languages F_{j_1} and F_{j_2} are distinct and they cannot both be equal to H_ℓ. Hence, $\ell = m$. Now consider any pair (E_j, F_j). Each $w \in E_j$ belongs to exactly one G_ℓ and this forces $F_j = H_\ell = R(w)$, and also that all elements of E_j must belong to G_ℓ. Moreover, G_ℓ cannot contain an element u outside of E_j, as otherwise $R(u) \neq F_j$ while also $R(u) = R(w)$.

Remark 4. *Here we show an example of a transduction* $T = \bigcup_{i=1}^{n} A_i \times B_i$ *for which the disjoint canonical form has a number* m *of cross products that meets the upper bound* $2^n - 1$. *Let* $p_1, \ldots, p_n$ *be any distinct primes, let each* $A_i = (a^{p_i})^\star$, *and let each* B_i *be any nonempty language. One verifies that, for each nonempty subset* I *of* $\{1, \ldots, n\}$, *the language* $(\cap_{i\in I} A_i) - (\cup_{\ell\in N-I} A_\ell)$ *is nonempty, as it contains the word* a^{n_I} *with* $n_I = \Pi_{i\in I} p_i$

PROOF. (Of Theorem 3.) We prove each statement in turn.

1. By Lemma 4, we can assume that all A_i's are mutually disjoint, and the same for all C_j's. First we have the following observation: Any word w

in the common domain of S and T belongs to a unique A_i and a unique C_j, so we have that $S(w) \neq T(w)$ iff $B_i \neq D_j$. Based on this observation, the language $\Delta_{S,T}$ is equal to the finite union of the nonempty regular languages $(A_i \cap C_j)$, where $i = 1, \ldots, n$ and $j = 1, \ldots, m$ with $B_i \neq D_j$. Hence, $\Delta_{S,T}$ is regular.

2. This statement is simply a constructive version of the previous one: using Lemma 4, we can construct NFA pair sets for S and T in disjoint normal form, and then construct an NFA for $\Delta_{S,T}$ using the regular operations in the above paragraph.
3. The construction of the desired NFA $\boldsymbol{f}$ mimics the definition of $\Delta_{S,T}$ in the proof of the first statement: $\boldsymbol{f}$ is the union of NFAs $\boldsymbol{f}_{i,j}$ accepting nonempty languages $(A_i \cap C_j)$ with $B_i \neq D_j$. An example of two NFA pair sets $\boldsymbol{A}$ and $\boldsymbol{C}$ describing S, T, respectively, such that the constructed $\boldsymbol{f}$ has the desired size is as follows: Let $p_1, \ldots, p_n, q_1, \ldots, q_m$ be distinct primes, let each $\boldsymbol{a}_i$ accept $(a^{p_i})^\star$ and each $\boldsymbol{c}_j$ accept $(a^{q_j})^\star$. Then $(a^{p_i})^\star \cap (a^{q_j})^\star \neq \emptyset$. Moreover, set $B_i = a^{p_i}$ and $D_j = a^{q_j}$ which implies $B_i \neq C_j$ for all i, j.

6. Chomsky-like Hierarchy of Difference Sets

For any transductions S, T of certain types, the languages $\Delta_{S,T}$ form a language class. In this section, we investigate how these classes are related to each other and to known classes (like the classes of context-sensitive languages **CSL** and one counter languages **OCL**). We use a notation similar to that of [5]: if $\mathtt{Y}$ is a type of transductions then $\mathcal{E}(\mathtt{Y})$ is the class of all equality sets between transductions of type $\mathtt{Y}$. For example, $\mathcal{E}(\mathsf{HOM})$ is the class of languages of the form $\mathcal{E}_{g,h}$, for some homomorphisms g, h. Similarly here we write $\Delta(\mathtt{Y})$ for the class of all difference sets between transductions of type $\mathtt{Y}$. We also write $\Delta(\mathtt{Y}_1, \mathtt{Y}_2)$ for the class of all difference sets between a transduction of type $\mathtt{Y}_1$ and one of type $\mathtt{Y}_2$. For example, $\Delta(\mathsf{FUNC}, \mathsf{TR})$ is the class of languages of the form $\Delta_{S,T}$, for some functional transduction S and some transduction T.

Below we state the main theorem of this section, and further below we present a few lemmata that lead to the proof of the main theorem.

Theorem 4. *The subset relations shown in Fig. 3 are correct.*

Unlike the case of recognizable transductions, the difference sets of homomorphic transductions do not include all the regular languages.

Proposition 1. *The difference set of any two homomorphisms is either $\emptyset$ or an infinite language. Moreover, the languages abR are not in $\Delta(\mathsf{HOM})$, for any regular R and for any two distinct alphabet letters a, b.*

Proof. Let g, h be homomorphisms such that $\Delta_{g,h}$ is nonempty. If $\mathcal{E}_{g,h}$ is finite then $\Delta_{g,h}$ must be infinite. If $\mathcal{E}_{g,h}$ is infinite then also $\Delta_{g,h}$ must be infinite as $\mathcal{E}_{g,h}\Delta_{g,h} \subseteq \Delta_{g,h}$.

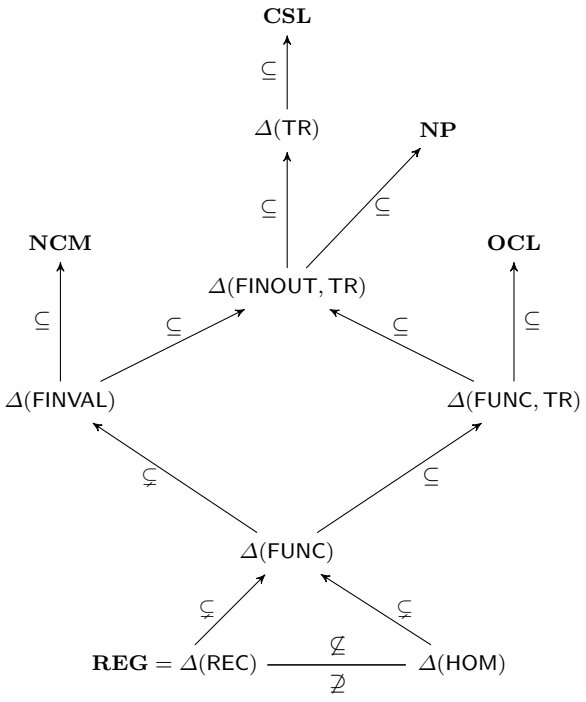

Figure 3: Subset relations between various classes of difference sets.

For the second statement, we use the fact that $\mathcal{E}_{g,h}$ is a star language [5]. We argue by contradiction: Assume that $abR = \Delta_{g,h}$, for some homomorphisms h, g; then $\Sigma^\star - abR = \mathcal{E}_{g,h}$ and $\Sigma^\star - abR = X^\star$, for some language X. But then $a, bx \in X^\star$, for any $x \in R$, which implies that $abx \in X^\star \cap abR$; a contradiction.

Lemma 5. *For all functional transductions $F_0, G_1, \ldots, G_k$, for $k \geq 1$, we have that*

$$\bigcap_{1 \leq j \leq k} \Delta_{F_0, G_j} \in \mathbf{NCM}(2k, 1).$$

Proof. We use the same notation $F_0, G_1, \ldots, G_k$ to denote transducers realizing the transductions. We adapt the proofs of [12, Theorem 2] and [4]. We construct a $(2k, 1)$-counter machine M accepting all words $w \in \mathsf{dom} F_0$ such that $F_0(w)$ is different from all $G_j(w)$. M has $2k$ counters and simulates the computations of F_0 on w and of G_j on w, using k counters b_j for F_0 and one counter c_j for each G_j, for $j = 1, \ldots, k$. Each pair b_j, c_j records the length of the output words during the computation. Each state of M records the current states of $F_0, G_1, \ldots, G_k$. Nondeterministically, M stops incrementing the counters and stores in the finite control the last symbols of the outputs σ_j and τ_j. At the end of the input, M checks, for each j, whether the proposition $b_j = c_j \wedge \sigma_j \neq \tau_j$ is

true—for the part $b_j = c_j$ the counters are decremented and tested if they are both zero. M accepts if and only if the propositions are true for all j.

PROOF. (Of Theorem 4.) To avoid cluttering in Fig. 3, we do not show the previously known class inclusions

$$\mathbf{OCL} \subseteq \mathbf{CSL} \quad \text{and} \quad \mathbf{NCM} \subseteq \mathbf{NP}, \mathbf{CSL}$$

where the last inclusion follows from [1, Theorem 5]. The following inclusions

$$\Delta(\mathsf{HOM}) \subseteq \Delta(\mathsf{FUNC}) \subseteq \Delta(\mathsf{FUNC}, \mathsf{TR}), \Delta(\mathsf{FINVAL}) \subseteq \Delta(\mathsf{FINOUT}, \mathsf{TR}) \subseteq \Delta(\mathsf{TR})$$

follow immediately from the fact that some transduction types are special cases of others, for example HOM is a special type of FUNC, FINVAL is a special type of FINOUT, and all are special types of TR. Next, we consider the rest of the inclusions in turn.

$\Delta(\mathsf{REC}) = \mathbf{REG}$: Follows from Theorem 3 and the fact that every regular language R is the difference set of the recognizable transductions $R \times \{0\}$ and $R \times \{1\}$.

$\mathbf{REG} \subsetneq \Delta(\mathsf{FUNC})$: The subset relation follows from the fact that every regular language R is the difference set of the functional transductions $R \times \{0\}$ and $R \times \{1\}$. The transductions $S(a^m b^n) = c^m$ and $T(a^m b^n) = c^n$, for $m, n \in \mathbb{N}_0$ and alphabet symbols a, b, c, are functional and $\Delta_{S,T} = \{a^m b^n : m \neq n\}$, which is a non-regular language.

$\mathbf{REG} \not\subseteq \Delta(\mathsf{HOM})$: Follows from Proposition 1.

$\Delta(\mathsf{HOM}) \not\subseteq \mathbf{REG}$: Follows from Example 2 of [5] stating that $\mathcal{E}_{g,h} = \{w \in \{a, b\}^\star : |w|_a = |w|_b\}$, for homomorphisms g, h such that $g(a) = 0, g(b) = \varepsilon, h(a) = \varepsilon, h(b) = 0$.

$\Delta(\mathsf{HOM}) \subsetneq \Delta(\mathsf{FUNC})$: We already know that $\Delta(\mathsf{HOM}) \subseteq \Delta(\mathsf{FUNC})$. Example 4 of [5] shows two functional transductions F, G such that $\mathsf{dom}F = \mathsf{dom}G = (a^+b^+)^\star$ and $\mathcal{E}_{F,G} = \{a^n b^n \mid n \geq 1\}^\star$ but $\mathcal{E}_{F,G} \notin \mathcal{E}(\mathsf{HOM})$. We can extend F, G such that $\mathsf{dom}F = \mathsf{dom}G = \{a, b\}^\star$ and $F(w) = 0$, $G(w) = 1$, for all $w \notin (a^+b^+)^\star$. Then, the extended F, G are still functional and again $\mathcal{E}_{F,G} = \{a^n b^n \mid n \geq 1\}^\star$. Moreover, we have that $\Delta_{F,G} = \overline{\mathcal{E}_{F,G}}$ and we can verify that $\Delta_{F,G}$ cannot be equal to $\Delta_{g,h}$ for any homomorphisms g, h (else $\mathcal{E}_{g,h}$ would be equal to $\mathcal{E}_{F,G}$).

$\Delta(\mathsf{FUNC}, \mathsf{TR}) \subseteq \mathbf{OCL}$: First we note the fact that $\Delta(\mathsf{FUNC}) \subseteq \mathbf{OCL}$, which is essentially a rephrasing of the Corollary of [4] stating that the complement of the equality set of two functional transductions is a one-counter language. Next we note that, for any two functional transductions F, G, the Theorem of [4] constructs a one counter automaton that accepts w iff $F(w)$ is not a prefix of $G(w)$ and $G(w)$ is not a prefix of $F(w)$. For a functional transduction S and a transduction T, one can mimic the proof of the Theorem of [4] to construct a

one-counter automaton accepting $\Delta_{S,T}$ in view of the simple fact that, for any word w, $S(w) \neq T(w)$ iff $T(w)$ contains a word $z \neq S(w)$.

$\Delta(\mathsf{FINVAL}) \subseteq \mathbf{NCM}$: Consider any finite-valued transductions F, G. As stated in Remark 3, references [33, 26] imply that $F = F_1 \cup \cdots \cup F_k$ and $G = G_1 \cup \cdots \cup G_\ell$, for some functional transductions F_i, G_j. Then we have that

$$\Delta_{F,G} = \bigcup_i \bigcap_j \Delta_{F_i,G_j} \cup \bigcup_j \bigcap_i \Delta_{F_i,G_j}$$

The claim follows when we note that each set $\bigcap_j \Delta_{F_i,G_j}$ (and also each set $\bigcap_i \Delta_{F_i,G_j}$) is in **NCM**, by Lemma 5, and the fact that the class **NCM** is closed under intersection and union [17].

$\Delta(\mathsf{TR}) \subseteq \mathbf{CSL}$: First note that $\mathbf{CSL} = \mathbf{NPSPACE}[n]$ (see e.g., [15]). For any fixed, but arbitrary, transductions S, T we show that deciding whether a given word w is in $\Delta_{S,T}$ can be done nondeterministically in space $O(|w|)$. Consider any transducers $\boldsymbol{s}, \boldsymbol{t}$ realizing S, T. We construct NFAs $\boldsymbol{c}, \boldsymbol{d}$ accepting the languages $\boldsymbol{s}(w)$ and $\boldsymbol{t}(w)$. These NFAs are of size $O(|w|)$, as $\boldsymbol{s}, \boldsymbol{t}$ are fixed. Using the nondeterministic algorithm in the proof of Lemma 1, we can decide whether $\mathcal{L}(\boldsymbol{c}) \not\subseteq \mathcal{L}(\boldsymbol{d})$ or $\mathcal{L}(\boldsymbol{d}) \not\subseteq \mathcal{L}(\boldsymbol{c})$ using space $O(|\boldsymbol{c}| + |\boldsymbol{d}|) = O(|w|)$. Hence, we can also decide whether $\mathcal{L}(\boldsymbol{c}) = \mathcal{L}(\boldsymbol{d})$ in nondeterministic space $O(|w|)$.

$\Delta(\mathsf{FINOUT}, \mathsf{TR}) \subseteq \mathbf{NP}$: For any fixed, but arbitrary, transductions S, T with $S \in \mathsf{FINOUT}$, there are transducers $\boldsymbol{s}$ and $\boldsymbol{t}$ realizing S, T. These transducers can be used to decide whether $w \in \Delta_{\boldsymbol{s},\boldsymbol{t}}$, for any given word w, exactly as in the proof of Theorem 1, where now the time complexity is only in terms of $|w|$, as $\boldsymbol{s}, \boldsymbol{t}$ are fixed.

$\Delta(\mathsf{FUNC}) \subsetneq \Delta(\mathsf{FINVAL})$: As mentioned already, $\Delta(\mathsf{FUNC}) \subseteq \Delta(\mathsf{FINVAL})$. The proper inclusion follows from Example 4 and the fact that the class **OCL** is a subset of the context-free languages.

Remark 5. *Due to the closure of the class* **CSL** *under complementation, the above result* $\Delta(\mathsf{TR}) \subseteq \mathbf{CSL}$ *implies that* $\mathcal{E}(\mathsf{TR}) \subseteq \mathbf{CSL}$*, which strengthens the earlier known fact* $\mathcal{E}(\mathsf{FUNC}) \subseteq \mathbf{CSL}$ *mentioned in Remark 1.*

7. Concluding Remarks

We introduced the concept of difference set of two transductions, which is complementary to the concept of equality set of transductions. While the word problems of the two concepts are essentially the same, the language classes resulting from the two concepts are different. We have also expressed in clear terms the concept of a PRAX algorithm that is now applicable to a language and its complement. Hence, there are PRAX algorithms for the word problem pertaining to either of the difference and equality sets.

The class hierarchy in Fig. 3 is incomplete. As future research we propose to investigate whether some of the inclusions are proper. For example, is there a one counter language that is not in $\Delta(\mathsf{FUNC})$?

The PRAX algorithm in Theorem 2 is a "tail-cutting" algorithm, that is, for the given approximation tolerance ε, the algorithm determines via the length M the tail of the probability distribution that can be safely ignored when testing the amount of difference of the output sets of the two transducers. However, if we know that transducer $\boldsymbol{s}$ (at least) has finite outputs then the algorithm can be modified to sample words from the uniform distribution on the finite set $\boldsymbol{s}(w)$. Details of this and possibly other similar improvements can be investigated in future research.

Acknowledgement. We thank Professors Tero Harju (Turku University, Finland) and Ian McQuillan (University of Saskatchewan, Canada) for suggesting some key references used in this paper.

References

[1] Baker, B.S., Book, R.V., 1974. Reversal-bounded multipushdown machines. J. Comput. Syst. Sci. 8, 315–332. URL: https://doi.org/10.1016/S0022-0000(74)80027-9.

[2] Berstel, J., 1979. Transductions and Context-Free Languages. B.G. Teubner, Stuttgart.

[3] Câmpeanu, C., Moreira, N., Reis, R., 2016. Distinguishability operations and closures. Fundam. Informaticae 148, 243–266. doi:10.3233/FI-2016-1434.

[4] Engelfriet, J., Hoogeboom, H.J., 1988. Prefix and equality languages of rational functions are co-context-free. Inf. Process. Lett. 28, 77–79. URL: https://doi.org/10.1016/0020-0190(88)90167-6.

[5] Engelfriet, J., Rozenberg, G., 1980. Fixed point languages, equality languages, and representation of recursively enumerable languages. J. ACM 27, 499–518. URL: https://doi.org/10.1145/322203.322211.

[6] Foryś, W., 1986. Fixed point languages of rational transductions. Semigroup Forum 34, 177–183.

[7] Ginsburg, S., 1958. On the length of the smallest uniform experiment which distinguishes the terminal states of a machine. Journal of the ACM 5, 266–280.

[8] Ginsburg, S., 1966. The Mathematical Theory of Context-free Languages. McGraw-Hill, Inc.

[9] Goldreich, O., 2008. Computational complexity - a conceptual perspective. Cambridge University Press. doi:10.1017/CBO9780511804106.

[10] Golomb, S.W., 1970. A class of probability distributions on the integers. Journal of Number Theory 2, 189–192.

[11] Golomb, S.W., 1992. Probability, information theory, and prime number theory. Discrete Mathematics 106/107, 219–229.

[12] Gurari, E.M., Ibarra, O.H., 1983. A note on finitely-valued and finitely ambiguous transducers. Math. Syst. Theory 16, 61–66. URL: https://doi.org/10.1007/BF01744569.

[13] Han, Y., Salomaa, A., Salomaa, K., 2017. Ambiguity, nondeterminism and state complexity of finite automata. Acta Cybern. 23, 141–157. doi:10.14232/actacyb.23.1.2017.9.

[14] Harju, T., Karhumäki, J., 1997. Morphisms, in: [24]. pp. 439–510.

[15] Hopcroft, J.E., Ullman, J.D., 1979. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley.

[16] Hromkovic, J., 1987. Reversal-bounded nondeterministic multicounter machines and complementation. Theor. Comput. Sci. 51, 325–330. URL: https://doi.org/10.1016/0304-3975(87)90040-5.

[17] Ibarra, O.H., 1978. Reversal-bounded multicounter machines and their decision problems. J. ACM 25, 116–133. URL: https://doi.org/10.1145/322047.322058.

[18] Immerman, N., 1988. Nondeterministic space is closed under complementation. SIAM J. Comput. 17, 935–938. URL: https://doi.org/10.1137/0217058.

[19] Konstantinidis, S., Mastnak, M., Moreira, N., Reis, R., 2023. Approximate NFA universality and related problems motivated by information theory. Theor. Comput. Sci. 972, 114076. URL: https://doi.org/10.1016/j.tcs.2023.114076.

[20] Konstantinidis, S., Santean, N., Yu, S., 2010. On implementing recognizable transductions. Int. J. Comput. Math. 87, 260–277. URL: https://doi.org/10.1080/00207160801968754. journal version of "Recognizable Transductions, Saturated Transducers and Edit Languages," Technical Report 2005-02, Department of Mathematics and Computing Science, Saint Mary's University, May 2005.

[21] Leung, H., 1998. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM J. Comput. 27, 1073–1082. doi:10.1137/S0097539793252092.

[22] Mateescu, A., Salomaa, A., 1997. Formal languages: an introduction and a synopsis, in: [24]. pp. 1–39.

[23] Roche, E., Schabes, Y., 1996. Introduction to Finite-State Devices in Natural Language Processing. Report TR-96-13. Mitsubishi Electric Research Laboratories.

[24] Rozenberg, G., Salomaa, A. (Eds.), 1997. Handbook of Formal Languages, Vol. I. Springer-Verlag, Berlin.

[25] Sakarovitch, J., 2009. Elements of Automata Theory. Cambridge University Press, Berlin.

[26] Sakarovitch, J., de Souza, R., 2008. On the decomposition of k-valued rational relations, in: Albers, S., Weil, P. (Eds.), STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany. pp. 621–632. URL: https://doi.org/10.4230/LIPIcs.STACS.2008.1324.

[27] Salomaa, A., 1973. Formal languages. Academic Press, New York.

[28] Salomaa, A., 1978. Equality sets for homomorphisms of free monoids. Acta Cybern. 4, 127–139. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3172.

[29] Santean, N., Yu, S., 2006. On weakly ambiguous finite transducers, in: Ibarra, O.H., Dang, Z. (Eds.), Developments in Language Theory, 10th International Conference, DLT 2006, Santa Barbara, CA, USA, June 26-29, 2006, Proceedings, Springer. pp. 156–167. URL: https://doi.org/10.1007/11779148_15.

[30] Savitch, W.J., 1970. Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci. 4, 177–192. URL: https://doi.org/10.1016/S0022-0000(70)80006-X.

[31] Stockmeyer, L., Meyer, A., 1973. Word problems requiring exponential time (preliminary report), in: Proceedings of the 5th annual ACM symposium on Theory of computing, ACM. pp. 1–9.

[32] Szelepcsényi, R., 1987. The method of forcing for nondeterministic automata. Bull. EATCS 33, 96–99.

[33] Weber, A., 1990. On the valuedness of finite transducers. Acta Informatica 27, 749–780. URL: https://doi.org/10.1007/BF00264285.

[34] Weber, A., 1993. Decomposing finite-valued transducers and deciding their equivalence. SIAM J. Comput. 22, 175–202. URL: https://doi.org/10.1137/0222014.

[35] Wood, D., 1987. Theory of Computation. Harper & Row, New York.

[36] Yu, S., 1997. Regular languages, in: [24]. pp. 41–110.