

On Finding a First-Order Sentence Consistent with a Sample of Strings

Thiago Alves Rocha

Department of Computing
Federal Institute of Ceará, Brazil
thiago.alves@ifce.edu.br

Ana Teresa Martins*

Department of Computing
Federal University of Ceará, Brazil
ana@dc.ufc.br

Francicleber Martins Ferreira

Department of Computing
Federal University of Ceará, Brazil
fran@lia.ufc.br

We investigate the following problem: given a sample of classified strings, find a first-order sentence of minimal quantifier rank that is consistent with the sample. We represent strings as successor string structures, that is, finite structures with unary predicates to denote symbols in an alphabet, and a successor relation. We use results of the Ehrenfeucht–Fraïssé game over successor string structures in order to design an algorithm to find such sentence. We use conditions characterizing the winning strategies for the Spoiler on successor strings structures in order to define formulas which distinguish two strings. Our algorithm returns a boolean combination of such formulas.

1 Introduction

In this paper, we explore the problem of finding a first-order formula that describes a given sample of classified strings. This problem is meaningful because strings may be used to model sequences of symbolic data such as biological sequences. For instance, in Table 1, we present a sample of classified strings.

Table 1:

String	Class
<i>stviil</i>	positive
<i>ktvive</i>	negative
<i>stviie</i>	positive
<i>stpiie</i>	negative

The sample in Table 1 represents biological sequences which have been associated with a group of diseases called amyloidosis [22]. The first-order sentence below represents that *stv* occurs in a string, and it describes the sample. Variables range over positions in strings, $P_a(i)$ is true if the symbol a occurs in position i , and S represents the successor relation over positions.

$$\exists x_1 \exists x_2 \exists x_3 (P_s(x_1) \wedge S(x_1, x_2) \wedge P_t(x_2) \wedge S(x_2, x_3) \wedge P_v(x_3)).$$

An algorithm to deal with the problem of finding a formula of minimal quantifier rank consistent with a given sample of structures over an arbitrary vocabulary is introduced in [13]. As this algorithm works for arbitrary finite relational structures, it runs in exponential time. This algorithm is applied in a

*This author was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under the grant number 424188/2016-3.

general system for learning formulas defining board game rules. These results are also used in finding reductions and polynomial-time programs [11, 12]. The work in [17] investigates a variation of the problem introduced in [13] when the class of structures is fixed. This study in [17] considers monadic structures, equivalence structures, and disjoint unions of linear orders.

In this paper, we study the problem introduced in [13] when the sample consists of strings represented by finite structures with a successor relation and a finite number of pairwise disjoint unary predicates. We call such a structure a successor string structure. A sample $S = (P, N)$ consists of two disjoint, finite sets P, N of successor string structures. Given a sample S , the problem is to find a first-order sentence φ_S of minimal quantifier rank that is consistent with S , i.e., it holds in all structures in P and does not hold in any structure in N . The size of the sample is the sum of the lengths of all strings in the sample. We intend to solve this problem in polynomial time in the size of S .

Ehrenfeucht–Fraïssé games (EF games) [5] is a fundamental technique in finite model theory [3, 6] in proving the inexpressibility of certain properties in first-order logic. For instance, first-order logic cannot express that a finite structure has even cardinality. The Ehrenfeucht–Fraïssé game is played on two structures by two players, the Spoiler and the Duplicator. If the Spoiler has a winning strategy for k rounds of such a game, it means that the structures can be distinguished by a first-order sentence φ whose quantifier rank is at most k , i.e., φ holds in exactly one of these structures.

Besides providing a tool to measure the expressive power of a logic, Ehrenfeucht–Fraïssé games allow one to investigate the similarity between structures. In [14], explicit conditions are provided for the characterization of winning strategies for the Duplicator on successor string structures. Using these conditions, the minimum number of rounds such that the Spoiler has a winning strategy in a game between two such structures can be computed in polynomial time in the size of the structures. This allows one to define a notion of similarity between successor string structures using Ehrenfeucht–Fraïssé games.

An essential part of the algorithm in [13] is the computation of r -Hintikka formulas from structures. An r -Hintikka formula is a formula obtained from a structure \mathcal{A} and a positive integer r that describes the properties of \mathcal{A} on the Ehrenfeucht–Fraïssé game with r rounds [3]. An r -Hintikka formula $\varphi_{\mathcal{A}}^r$ has size exponential in the size of \mathcal{A} and holds exactly on all structures \mathcal{B} such that the Duplicator has a winning strategy for the Ehrenfeucht–Fraïssé game with r rounds on \mathcal{A} and \mathcal{B} . Besides, Hintikka formulas are representative because any first-order formula is equivalent to a disjunction of Hintikka formulas.

We use results of the Ehrenfeucht–Fraïssé game over successor string structures [14] in order to design an algorithm to find a sentence which is consistent with the sample in polynomial time. Also, as the size of a Hintikka formula is exponential in the size of a given structure, our algorithm does not use Hintikka formulas. In our case, we define what we call distinguishability formulas. They are defined for two successor string structures u, v and a natural number r based on conditions characterizing the winning strategies for the Spoiler on successor strings structures [14]. In this way, we show that distinguishability formulas hold on u , do not hold on v , and they have quantifier rank at most r . This result is also crucial for the definition of our algorithm and to guarantee its correctness. Our algorithm returns a disjunction of conjunctions of distinguishability formulas. We also show that any first-order formula over successor string structures is equivalent to a boolean combination of distinguishability formulas. This result suggests that our approach has the potential to find any first-order sentence.

Our framework is close to grammatical inference. Research in this area investigates the problem of finding a language model of minimal size consistent with a given sample of strings [9]. A language model can be a deterministic finite automaton (DFA) or a context-free grammar, for instance. Grammatical inference has applications in many areas because strings may be used to model text data, traces of program executions, biological sequences, and sequences of symbolic data in general.

A recent model-theoretic approach to grammatical inference is introduced in [20]. In this approach, it is also used successor string structures to represent strings and first-order sentences as a representation of formal languages. Then, our approach may also be seen as a model-theoretic framework to grammatical inference. The first main difference is that we work with full first-order logic, while the approach in [20] uses a fragment called CNPL. Formulas of CNPL have the form $\bigwedge_{w_1 \in W_1} \phi_{w_1} \wedge \bigwedge_{w_2 \in W_2} \phi_{w_2}$ such that each ϕ_u is a first-order sentence which defines exactly all strings w such that u is a substring of w . Also, CNPL is less expressive than first-order logic. Second, given k , the goal of the framework in [20] is to find a CNPL formula such that $\max\{|w| \mid w \in W_1 \cup W_2\} \leq k$ and $|w|$ is the length of w . Our goal is to find a first-order sentence of minimal quantifier rank.

It is well known that a language is definable in first-order logic over successor string structures if and only if it is a locally threshold testable (LTT) language [21]. A language is LTT if membership of a string can be tested by inspecting its prefixes, suffixes, and infixes up to some length, and counting infixes up to some threshold. The class of LTT languages is a subregular class, i.e., a subclass of the regular languages [19]. A grammatical inference algorithm that returns a DFA may return an automaton which recognizes a language not in LTT. Therefore, our results can be useful when one desires to find a model of an LTT language from a sample of strings. We believe that this is the first work on finding a language model of LTT languages from positive and negative strings.

A recent logical framework to find a formula given a sample, also with a model-theoretic approach, can be found in [8, 7]. In this framework, a sample consists of classified elements from only one structure. The problem is to find a hypothesis consistent with the classified elements where this hypothesis is a formula from some logic. Recall that, in our framework, samples consist of many classified structures. Another logical framework for a similar problem is Inductive Logic Programming (ILP) [15, 2]. ILP uses logic programming as a uniform representation for the sample and hypotheses. As far as we know, our work has no direct relationship with these frameworks.

This paper is organized as follows. In Section 2, we give the necessary definitions of formal language theory and finite model theory used in this paper. Also in Section 2, we have an EF game characterization on strings, and, in Section 3, we translate it into first-order sentences. In Section 3, we also introduce the concept of distinguishability formulas providing some useful properties. In Section 4, we introduce our algorithm, give an example of how the algorithm works, and show its correctness. Furthermore, in this section, we briefly discuss how to find a formula with the minimum number of conjunctions. We conclude in Section 5.

2 Formal Languages and EF Games on Strings

We consider strings over an alphabet Σ . The set of all such finite strings is denoted by Σ^* , and the empty string by ε . If w is a string, then $|w|$ is the length of w . Let uv denote the concatenation of strings u and v . For all $u, v, w, x \in \Sigma^*$, if $w = uxv$, then x is a substring of w . Moreover, if $u = \varepsilon$ (resp. $v = \varepsilon$) we say that x is a prefix (resp. suffix) of w . We denote the prefix (resp. suffix) of length k of w by $pref_k(w)$ (resp. $suff_k(w)$). Let i and j be positions in a string. The distance between i and j , denoted by $d(i, j)$, is $|i - j|$. A formal language is a subset of Σ^* . A language is locally threshold testable (LTT) if it is a boolean combination of languages of the form $\{w \mid u \text{ is prefix of } w\}$, for some $u \in \Sigma^*$, $\{w \mid u \text{ is suffix of } w\}$, for some $u \in \Sigma^*$, and $\{w \mid w \text{ has } u \text{ as infix at least } d \text{ times}\}$, for some $u \in \Sigma^*$ and $d \in \mathbb{N}$ [23]. Therefore, membership of a string can be tested by inspecting its prefixes, suffixes and infixes up to some length, and counting infixes up to some threshold. We assume some familiarity with formal languages. See [10] for details.

We view a string $w = a_1 \dots a_n$ as a logical structure \mathcal{A}_w over the vocabulary $\tau = \{S, (P_a)_{a \in \Sigma}, \min, \max\}$ with domain $A = \{1, \dots, n\}$, that is, the elements of A are positions of w . The predicate S is the successor relation and each P_a is a unary predicate for positions labeled with a . The constants \min and \max are interpreted as the positions 1 and n , respectively. We call these structures successor string structures. We assume some familiarity with first-order logic (FO), and we use this logic over successor string structures. For details on first-order logic see [3, 4]. The size of a first-order formula φ is the number of symbols occurring in φ . By the quantifier rank of a formula, we mean the depth of nesting of its quantifiers as in the following.

Definition 1 (Quantifier Rank). *Let φ be a first-order formula. The quantifier rank of φ , written $qr(\varphi)$, is defined as*

$$qr(\varphi) := \begin{cases} 0, & \text{if } \varphi \text{ is atomic} \\ \max(qr(\varphi_1), qr(\varphi_2)), & \text{if } \varphi = \varphi_1 \square \varphi_2 \text{ such that } \square \in \{\wedge, \vee, \leftarrow\} \\ qr(\psi), & \text{if } \varphi = \neg \psi \\ qr(\psi) + 1, & \text{if } \varphi = Qx\psi \text{ such that } Q \in \{\exists, \forall\} \end{cases}$$

Given a first-order sentence φ over successor string structures, the formal language defined by φ is simply $L(\varphi) := \{w \in \Sigma^* \mid \mathcal{A}_w \models \varphi\}$. In general, we do not distinguish between successor string structures and strings. As an example, if $\varphi = \exists x P_a(x)$, then $L(\varphi) = \Sigma^* a \Sigma^*$. LTT languages can be defined in terms of first-order logic. A language is definable by a sentence of FO over successor string structures if and only if it is LTT [21].

Now, we can formally define the problem we are interested in. A sample $S = (P, N)$ is a finite number of classified strings consisting of two disjoint, finite sets $P, N \subseteq \Sigma^*$ of strings over an alphabet Σ . Intuitively, P contains positively classified strings, and N contains negatively classified strings. The size of a sample S is the sum of the lengths of all strings it includes. We use $|S|$ to denote the size of the sample S . A sentence φ is consistent with a sample S if $P \subseteq L(\varphi)$ and $N \cap L(\varphi) = \emptyset$. Therefore, a sentence is consistent with a sample if it holds in all strings in P and does not hold in any string in N . Given a sample S , the problem consists of finding a first-order sentence φ of minimum quantifier rank such that φ is consistent with S .

It is well known that every finite structure can be characterized in first-order logic up to isomorphism, i.e., for every finite structure \mathcal{A} , there is a first-order sentence $\varphi_{\mathcal{A}}$ such that for all structures \mathcal{B} we have $\mathcal{B} \models \varphi_{\mathcal{A}}$ iff \mathcal{A} and \mathcal{B} are isomorphic. Since samples are finite sets of finite structures, one can easily build in polynomial-time a first-order sentence consistent with a given sample. For example, let $P = \{bbabbb, baba\}$ and $N = \{bbbb\}$. The sentence $\varphi_{bbabbb} \vee \varphi_{baba}$ is consistent with the sample. Unfortunately, the quantifier rank of $\varphi_{\mathcal{A}}$ is the number of elements in the domain of \mathcal{A} plus one. Then, $\varphi = \exists x P_a(x)$ is also consistent with the sample and $qr(\varphi) < qr(\varphi_{bbabbb} \vee \varphi_{baba})$. Therefore, $\varphi_{bbabbb} \vee \varphi_{baba}$ is not a solution to the problem.

Now, we focus on Ehrenfeucht–Fraïssé games and its importance in order to solve the problem we are considering. Let r be an integer such that $r \geq 0$, u and v two successor string structures. The Ehrenfeucht–Fraïssé game $\mathcal{G}_r(u, v)$ is played by two players called the Spoiler and the Duplicator. Each play of the game has r rounds and, in each round, the Spoiler plays first and picks an element from the domain A of u , or from the domain B of v . Then, the Duplicator responds by picking an element from the domain of the other structure. Let $a_i \in A$ and $b_i \in B$ be the two elements picked by the Spoiler and the Duplicator in the i th round. The Duplicator wins the play if the mapping $(a_1, b_1), \dots, (a_r, b_r)$ is an isomorphism between the substructures induced by a_1, \dots, a_r and b_1, \dots, b_r , respectively. Otherwise, Spoiler wins this play. We say that a player has a winning strategy in $\mathcal{G}_r(u, v)$ if it is possible for her

to win each play whatever choices are made by the opponent. In this work, we always assume that u is different from v . Note that if $r \geq |u| + |v|$, then the Spoiler has a winning strategy. Therefore, we assume that r is bounded by $|u| + |v|$. Now, we define formulas describing the properties of a structure in EF games.

Definition 2 (Hintikka Formulas). *Let w be a structure, $\bar{a} = a_1 \dots a_s \in A^s$, and $\bar{x} = x_1, \dots, x_s$ a tuple of variables,*

$$\begin{aligned} \varphi_{w,\bar{a}}^0(\bar{x}) &:= \bigwedge \{ \varphi(\bar{x}) \mid \varphi \text{ is atomic or negated atomic and } w \models \varphi[\bar{a}] \}, \\ \text{and for } r > 0, \varphi_{w,\bar{a}}^r(\bar{x}) &:= \bigwedge_{a \in A} \exists x_{s+1} \varphi_{w,\bar{a}a}^{r-1}(\bar{x}, x_{s+1}) \wedge \forall x_{s+1} \left(\bigvee_{a \in A} \varphi_{w,\bar{a}a}^{r-1}(\bar{x}, x_{s+1}) \right). \end{aligned}$$

A Hintikka formula $\varphi_{w,\bar{a}}^r$ describes the isomorphism type of the substructure generated by \bar{a} in w . We write φ_w^r whenever $s = 0$. Given a string w and a positive integer r , the size of the r -Hintikka formula φ_w^r is $O(2^r \times |w|^r)$. Therefore, since r is bounded by $|w|$, the size of φ_w^r is exponential in the size of w . The following theorems are important to prove our main results. They are presented in [3] (Theorem 2.2.8 and Theorem 2.2.11).

Theorem 1 (Ehrenfeucht's Theorem). *Given u and v , and $r \geq 0$, the following are equivalent:*

- *the Duplicator has a winning strategy in $\mathcal{G}_r(u, v)$.*
- *If φ is a sentence of quantifier rank at most r , then $u \models \varphi$ iff $v \models \varphi$.*
- *$v \models \varphi_u^r$.*

Theorem 2. *Let φ be a sentence of quantifier rank at most r . Then, there exists structures u_1, \dots, u_k such that*

$$\models \varphi \leftrightarrow (\varphi_{u_1}^r \vee \dots \vee \varphi_{u_k}^r).$$

We use Theorem 2 in order to show that any first-order formula over successor string structures is equivalent to a boolean combination of distinguishability formulas. EF games are essential in our framework because if the Spoiler has a winning strategy in a game on strings u and v with r rounds, then there exists a first-order sentence φ of quantifier rank at most r that holds in u and does not hold in v . Also, in this case, the sentence φ_u^r is an example of such a sentence. Unfortunately, over arbitrary vocabularies, the problem of determining whether the Spoiler has a winning strategy is *PSPACE*-complete [16].

However, it is possible to do better in the particular case of EF games on successor string structures. For details see [14]. First, we need the following definitions. Let $A \subseteq \mathbb{N}$. A partition of A is a collection of subsets X of A such that each element of A is included in exactly one subset. An l -segmentation of A is a partition of A with the minimum number of subsets such that for all i, j in the same subset, $d(i, j) \leq l$ and if i, j are in the same subset X and $i \leq h \leq j$, then $h \in X$. Each subset X in the partition is called a segment.

In the following, we consider substrings α over Σ such that $|\alpha| = 2^{q_\alpha} - 1$ for some $q_\alpha > 0$. Let $w = w_1 \dots w_n$ be a string such that $w_i \in \Sigma$, for $i \in \{1, \dots, n\}$. An occurrence of α is centered on a position i in a string w if $w_{i-(2^{q_\alpha}-1)} \dots w_i \dots w_{i+2^{q_\alpha}-1} = \alpha$. An occurrence of α centered on a position i in w is free if $|\min^w - i| > 2^{q_\alpha-1}$ and $|\max^w - i| > 2^{q_\alpha-1}$. The set of free occurrences of α in w is $\Gamma(\alpha, w) = \{i \mid |\min^w - i| > 2^{q_\alpha-1}, |\max^w - i| > 2^{q_\alpha-1}, w_{i-(2^{q_\alpha}-1)} \dots w_i \dots w_{i+2^{q_\alpha}-1} = \alpha\}$. The free multiplicity of α in w , denoted by $\gamma(\alpha, w)$, is the number of free occurrences of α in w , i.e., $|\Gamma(\alpha, w)|$. The free scattering of α in w , denoted by $\sigma(\alpha, w)$, is the number of segments in a 2^{q_α} -segmentation of $\Gamma(\alpha, w)$.

Example 1. *Let $w = ababababbababaaba$ and $\alpha = aba$. Note that $q_\alpha = 2$. The occurrence of α centered on position 2 in w is not free because $|\min^w - 2| \leq 2$. However, the occurrence of α centered on position 4 in w is free. The set of free occurrences of α in w is $\Gamma(\alpha, w) = \{4, 6, 11, 13\}$. Therefore, $\gamma(\alpha, w) = 4$. A 2^{q_α} -segmentation of $\Gamma(\alpha, w)$ is $\{\{4, 6\}, \{11, 13\}\}$. Then, $\sigma(\alpha, w) = 2$.*

Now, we have a result of EF games on successor string structures.

Theorem 3. [14] *Let r be a natural number, u and v be strings. The Duplicator has a winning strategy in $\mathcal{G}_r(u, v)$ if and only if the following conditions hold:*

1. $d(\min^u, \max^u) = d(\min^v, \max^v)$ or $(d(\min^u, \max^u) > 2^r \text{ and } d(\min^v, \max^v) > 2^r)$;
2. $\text{pref}_{2^r}(u) = \text{pref}_{2^r}(v)$ and $\text{suff}_{2^r}(u) = \text{suff}_{2^r}(v)$;
3. $\sigma(\alpha, u) + q_\alpha > r$ and $\sigma(\alpha, v) + q_\alpha > r$ for all α such that $|\alpha| = 2^{q_\alpha} - 1$ and $(\sigma(\alpha, u) \neq \sigma(\alpha, v) \text{ or } \gamma(\alpha, u) \neq \gamma(\alpha, v))$.

Besides the importance of EF games on strings to our framework, we also use the above result to define the distinguishability formulas. These formulas are defined based on the conditions characterizing a winning strategy for the Spoiler on successor string structures. In [14], this result is also used to define a notion of similarity between successor string structures using Ehrenfeucht–Fraïssé games. The EF-similarity between strings u and v , written $EFsim(u, v)$, is the minimum number of rounds r such that the Spoiler has a winning strategy in the game $\mathcal{G}_r(u, v)$. Then, the EF-similarity between two strings can be computed in polynomial time in the size of the strings in the following way.

$$\begin{aligned} EFsim(u, v) &:= \min\{\text{simLength}(u, v), \text{simPref}(u, v), \text{simSuff}(u, v), \text{simSub}(u, v)\}, \text{ such that} \\ \text{simLength}(u, v) &= \lceil \log_2(\min(|u|, |v|) - 1) \rceil, \\ \text{simPref}(u, v) &= \min\{\lceil \log_2(k) \rceil \mid \text{pref}_k(u) \neq \text{pref}_k(v)\}, \\ \text{simSuff}(u, v) &= \min\{\lceil \log_2(k) \rceil \mid \text{suff}_k(u) \neq \text{suff}_k(v)\}, \\ \text{simSub}(u, v) &= \min\{q_\alpha + \min(\sigma(\alpha, u), \sigma(\alpha, v)) \mid \gamma(\alpha, u) \neq \gamma(\alpha, v) \text{ or } \sigma(\alpha, u) \neq \sigma(\alpha, v)\}. \end{aligned}$$

Given two strings u and v , $EFsim(u, v)$ can be computed in $O((|u| + |v|)^2 \log(|u| + |v|))$, that is, it can be computed in polynomial time [14]. Our algorithm's first step is to compute the sufficient quantifier rank to distinguish between any two strings $u \in P$ and $v \in N$. Then, the fact that $EFsim(u, v)$ can be computed in polynomial time is important to show that our algorithm runs in polynomial time as well.

It is easy to build a first-order sentence consisting of a disjunction of Hintikka formulas of minimal quantifier rank that is consistent with a given sample. For example, let $P = \{u_1\}$, $N = \{v_1, v_2\}$, $r = \max\{EFsim(u_1, v_1), EFsim(u_1, v_2)\}$, and $S = (P, N)$. The sentence $\varphi_{u_1}^r$ is a first-order sentence of minimal quantifier rank that is consistent with S . Unfortunately, the size of $\varphi_{u_1}^r$ is exponential in the size of S . Therefore, $\varphi_{u_1}^r$ can not be built in polynomial time in the size of the sample. This motivates the introduction of distinguishability formulas in Section 3.

3 Distinguishability Formulas

In this section, we define distinguishability formulas for strings u, v and a natural number r . Distinguishability formulas are formulas that hold on u , do not hold on v and they have quantifier rank at most r . The first step is to show that the conditions of Theorem 3 can be expressed by first-order formulas. These formulas are defined recursively in order to reduce the quantifier rank. The recursive definitions can all be simplified to direct definitions with higher quantifier ranks but, in this case, we can not guarantee that the quantifier rank is adequate. These formulas are also important to help the explanation, and they improve readability of sentences returned by our algorithm.

First, we introduce $\varphi_{\leq n}^{d(t_1, t_2)}$. It describes that the distance between terms t_1 and t_2 is at most n . This can be used to represent condition 1 of Theorem 3.

$$\varphi_{\leq n}^{d(t_1, t_2)} := \begin{cases} t_1 = t_2 \vee S(t_1, t_2), & \text{if } n = 1 \\ \exists y(\varphi_{\leq \lfloor \frac{n}{2} \rfloor}^{d(t_1, y)} \wedge \varphi_{\leq \lceil \frac{n}{2} \rceil}^{d(y, t_2)}), & \text{otherwise.} \end{cases}$$

We also set $\varphi_{>n}^{d(t_1,t_2)} := \neg\varphi_{\leq n}^{d(t_1,t_2)}$, $\varphi_{\geq n}^{d(t_1,t_2)} := \varphi_{>n-1}^{d(t_1,t_2)}$, $\varphi_{<n}^{d(t_1,t_2)} := \neg\varphi_{\geq n}^{d(t_1,t_2)}$, and $\varphi_{=n}^{d(t_1,t_2)} := \varphi_{\leq n}^{d(t_1,t_2)} \wedge \varphi_{\geq n}^{d(t_1,t_2)}$. Clearly, $qr(\varphi_{>n}^{d(t_1,t_2)}) = \lceil \log_2(n) \rceil$ and $w \models \varphi_{>n}^{d(\min, \max)}$ iff $|w| \triangleright n + 1$ for $\triangleright \in \{<, >, \leq, \geq, =\}$. Besides, the size of $\varphi_{>n}^{d(t_1,t_2)}$ is $O(n)$. For example, for $r = 3$ and strings u and v such that $|u| = 9$ and $|v| = 12$, we have that $u \models \varphi_{\leq 8}^{d(\min, \max)}$, $v \not\models \varphi_{\leq 8}^{d(\min, \max)}$, and $qr(\varphi_{\leq 8}^{d(\min, \max)}) = 3$. Then, $d(\min^u, \max^u) \neq d(\min^v, \max^v)$ and $d(\min^u, \max^u) \leq 2^r$. Therefore, the Spoiler has a winning strategy for $\mathcal{G}_3(u, v)$.

Now, we turn to the cases in which substrings are important. These cases are conditions 2 and 3 from Theorem 3. Formulas $\varphi_{t_1 a_1 \dots a_k t_2}$ hold in a string w when the string between t_1 and t_2 is $a_1 \dots a_k$. Formulas $\varphi_{t a_1 \dots a_k}$ and $\varphi_{a_1 \dots a_k t}$ express that a string $a_1 \dots a_k$ occurs immediately on the right and immediately on the left of a term t , respectively.

$$\varphi_{t_1 a_1 \dots a_k t_2} := \begin{cases} \exists z (P_{a_1}(z) \wedge S(t_1, z) \wedge S(z, t_2)), & \text{if } k = 1 \\ \exists z (P_{a_1}(z) \wedge S(t_1, z) \wedge \varphi_{z a_2 t_2}), & \text{if } k = 2 \\ \exists z (P_{a_{\lfloor \frac{k}{2} \rfloor}}(z) \wedge \varphi_{t_1 a_1 \dots a_{\lfloor \frac{k}{2} \rfloor - 1} z} \wedge \varphi_{z a_{\lfloor \frac{k}{2} \rfloor + 1} \dots a_k t_2}), & \text{otherwise.} \end{cases}$$

$$\varphi_{t a_1 \dots a_k} := \begin{cases} \exists y (S(t, y) \wedge P_{a_1}(y)), & \text{if } k = 1 \\ \exists y (P_{a_1}(y) \wedge S(t, y) \wedge \varphi_{y a_2}), & \text{if } k = 2 \\ \exists y (P_{a_{\lfloor \frac{k}{2} \rfloor}}(y) \wedge \varphi_{t a_1 \dots a_{\lfloor \frac{k}{2} \rfloor - 1} y} \wedge \varphi_{y a_{\lfloor \frac{k}{2} \rfloor + 1} \dots a_k}), & \text{otherwise.} \end{cases}$$

$$\varphi_{a_1 \dots a_k t} := \begin{cases} \exists y (S(y, t) \wedge P_{a_1}(y)), & \text{if } k = 1 \\ \exists y (P_{a_1}(y) \wedge \varphi_{y a_2 t}), & \text{if } k = 2 \\ \exists y (P_{a_{\lfloor \frac{k}{2} \rfloor}}(y) \wedge \varphi_{a_1 \dots a_{\lfloor \frac{k}{2} \rfloor - 1} y} \wedge \varphi_{y a_{\lfloor \frac{k}{2} \rfloor + 1} \dots a_k t}), & \text{otherwise.} \end{cases}$$

With respect to the quantifier rank, we have $qr(\varphi_{t_1 a_1 \dots a_k t_2}) = qr(\varphi_{t a_1 \dots a_k}) = qr(\varphi_{a_1 \dots a_k t}) = \lceil \log_2(k + 1) \rceil$. Furthermore, the size of these formulas is $O(k)$. Now, we define sentences to handle the prefix and suffix of strings. These sentences express that the prefix of length k is $a_1 \dots a_k$ and the suffix of length k is $a_1 \dots a_k$, respectively.

$$\varphi_{\text{pref}_k = a_1 \dots a_k} := \begin{cases} P_{a_1}(\min), & \text{if } k = 1 \\ P_{a_1}(\min) \wedge \varphi_{\min a_2 \dots a_k}, & \text{if } k = 2 \text{ or } k = 3 \\ P_{a_1}(\min) \wedge \exists x (P_{a_{\lfloor \frac{k+1}{2} \rfloor}}(x) \wedge \varphi_{\min a_2 \dots a_{\lfloor \frac{k+1}{2} \rfloor - 1} x} \wedge \varphi_{x a_{\lfloor \frac{k+1}{2} \rfloor + 1} \dots a_k}), & \text{otherwise.} \end{cases}$$

$$\varphi_{\text{suff}_k = a_1 \dots a_k} := \begin{cases} P_{a_k}(\max), & \text{if } k = 1 \\ P_{a_k}(\max) \wedge \varphi_{a_1 \dots a_{k-1} \max}, & \text{if } k = 2 \text{ or } k = 3 \\ P_{a_k}(\max) \wedge \exists x (P_{a_{\lfloor \frac{k}{2} \rfloor}}(x) \wedge \varphi_{a_1 \dots a_{\lfloor \frac{k}{2} \rfloor - 1} x} \wedge \varphi_{x a_{\lfloor \frac{k}{2} \rfloor + 1} \dots a_{k-1} \max}), & \text{otherwise.} \end{cases}$$

We also set abbreviations $\varphi_{\text{pref}_k \neq a_1 \dots a_k} := \neg \varphi_{\text{pref}_k = a_1 \dots a_k}$ and $\varphi_{\text{suff}_k \neq a_1 \dots a_k} := \neg \varphi_{\text{suff}_k = a_1 \dots a_k}$. Therefore, $qr(\varphi_{\text{pref}_k \blacktriangleright a_1 \dots a_k}) = \lceil \log_2(k) \rceil$ and $w \models \varphi_{\text{pref}_k \blacktriangleright a_1 \dots a_k}$ iff $\text{pref}_k(w) \blacktriangleright a_1 \dots a_k$, where $\blacktriangleright \in \{=, \neq\}$. Analogously for $\varphi_{\text{suff}_k \blacktriangleright a_1 \dots a_k}$. Also, the size of $\varphi_{\text{pref}_k = a_1 \dots a_k}$ and $\varphi_{\text{suff}_k = a_1 \dots a_k}$ is $O(k)$. We use these formulas to express condition 2 of Theorem 3. To see why, Let $r = 2$, $u = \text{bbbaabbbb}$ and $v = \text{bbbbabbbb}$. Thus, $u \models \varphi_{\text{pref}_4 \neq \text{bbbb}}$ and $v \not\models \varphi_{\text{pref}_4 \neq \text{bbbb}}$. Then, $\text{pref}_2^r(u) \neq \text{pref}_2^r(v)$ and, from condition 2 of Theorem 3, it follows that the Spoiler has a winning strategy in $\mathcal{G}_2(u, v)$.

Now, we need sentences regarding free multiplicity and free scattering. Let $\alpha = a_1 \dots a_k$ be a string such that each $a_i \in \Sigma$, and $k = 2^{q\alpha} - 1$ for $q\alpha > 0$ as in condition 3 from Theorem 3. Now, we set

the formula $\varphi_{a_1\dots a_k}(x)$ describing that a string $a_1\dots a_k$ occurs centered on position x . Then, we give an example of a formula $\varphi_\alpha(x)$.

$$\varphi_{a_1\dots a_k}(x) := \begin{cases} P_{a_1}(x), & \text{if } k = 1 \\ P_{a_{\lceil \frac{k}{2} \rceil}}(x) \wedge \varphi_{a_1\dots a_{\lceil \frac{k}{2} \rceil - 1}}(x) \wedge \varphi_{a_{\lceil \frac{k}{2} \rceil + 1}\dots a_k}, & \text{if } k \geq 3. \end{cases}$$

Example 2. Let $\alpha = abc$. Then,

$$\varphi_\alpha(x) = P_b(x) \wedge \exists y_1 (S(y_1, x) \wedge P_a(y_1)) \wedge \exists y_1 (P_c(y_1) \wedge S(x, y_1)).$$

Note that $qr(\varphi_\alpha(x)) = q_\alpha - 1$ and the size of $\varphi_\alpha(x)$ is $O(k)$. Now, we can use formulas $\varphi_\alpha(x)$ to define $\varphi_{\gamma(\alpha) \geq n}$ expressing that α has at least n free occurrences. Then, we need to use n pairwise different variables and each variable must be in a proper distance from min and max .

$$\varphi_{\gamma(\alpha) \geq n} := \exists x_1 \exists x_2 \dots \exists x_n (\bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \wedge \bigwedge_{i=1}^n \varphi_\alpha(x_i) \wedge \bigwedge_{i=1}^n (\varphi_{>2^{q_\alpha-1}}^{d(min, x_i)} \wedge \varphi_{>2^{q_\alpha-1}}^{d(x_i, max)})).$$

Now, we need to deal with formulas $\varphi_{\sigma(\alpha) \geq n}$ expressing that the scattering of α is at least n . First, in the following, we set an auxiliary formula in order to make the presentation simpler. The formula below indicates that α occurs centered on a position on the left of x and at least $2^{q_\alpha-1}$ distant from x . This formula is important in ensuring a proper distance from other occurrences of α , that is, greater than 2^{q_α} . Furthermore, the distance between α and min or max must be greater than $2^{q_\alpha-1}$ in order to α occur free.

$$\varphi_\alpha^{d(x) \geq 2^{q_\alpha-1}} := \exists y (\varphi_{\geq 2^{q_\alpha-1}}^{d(y, x)} \wedge \varphi_\alpha(y) \wedge \varphi_{>2^{q_\alpha-1}}^{d(min, y)} \wedge \varphi_{>2^{q_\alpha-1}}^{d(y, max)}).$$

With respect to the quantifier rank, we have $qr(\varphi_\alpha^{d(x) \geq 2^{q_\alpha-1}}) = q_\alpha$. Now, we can define the sentence $\varphi_{\sigma(\alpha) \geq n}$. After that, we give an example of $\varphi_{\gamma(\alpha) \geq n}$ and $\varphi_{\sigma(\alpha) \geq n}$.

$$\varphi_{\sigma(\alpha) \geq n} := \exists x_1 (\varphi_\alpha^{d(x_1) \geq 2^{q_\alpha-1}} \wedge \exists x_2 (\varphi_{>2^{q_\alpha}}^{d(x_1, x_2)} \wedge \varphi_\alpha^{d(x_2) \geq 2^{q_\alpha-1}} \wedge \dots \wedge \exists x_{n-1} (\varphi_{>2^{q_\alpha}}^{d(x_{n-2}, x_{n-1})} \wedge \varphi_\alpha^{d(x_{n-1}) \geq 2^{q_\alpha-1}} \wedge \exists x_n (\varphi_{>2^{q_\alpha-1}}^{d(x_{n-1}, x_n)} \wedge \varphi_\alpha(x_n)))))).$$

Example 3. Let $\alpha = abc$ and $n = 2$. Thus,

$$\varphi_{\gamma(\alpha) \geq n} = \exists x_1 \exists x_2 (x_1 \neq x_2 \wedge \varphi_{abc}(x_1) \wedge \varphi_{abc}(x_2) \wedge \varphi_{>2^{q_\alpha-1}}^{d(min, x_1)} \wedge \varphi_{>2^{q_\alpha-1}}^{d(x_1, max)} \wedge \varphi_{>2^{q_\alpha-1}}^{d(min, x_2)} \wedge \varphi_{>2^{q_\alpha-1}}^{d(x_2, max)}).$$

$$\varphi_{\sigma(\alpha) \geq n} = \exists x_1 (\varphi_\alpha^{d(x_1) \geq 2^{q_\alpha-1}} \wedge \exists x_2 (\varphi_{>2^{q_\alpha-1}}^{d(x_1, x_2)} \wedge \varphi_\alpha(x_2))).$$

We also define the following abbreviations $\varphi_{\gamma(\alpha) < n} := \neg \varphi_{\gamma(\alpha) \geq n}$ and $\varphi_{\gamma(\alpha) = n} := \varphi_{\gamma(\alpha) \geq n} \wedge \varphi_{\gamma(\alpha) < n+1}$. Then, $qr(\varphi_{\gamma(\alpha) \leq n}) = q_\alpha + n - 1$ and $w \models \varphi_{\gamma(\alpha) \leq n}$ iff $\gamma(\alpha, w) \leq n$ for $\leq \in \{\geq, <, =\}$. It is analogous to $\varphi_{\sigma(\alpha) \leq n}$. Besides, the size of $\varphi_{\gamma(\alpha) \leq n}$ and $\varphi_{\sigma(\alpha) \leq n}$ is $O((n + |\alpha|)^2)$.

Now, we can define the distinguishability formulas. Distinguishability formulas are defined from a pair of strings u, v and a quantifier rank r . These formulas have quantifier rank at most r , and they hold in u and do not hold in v . In what follows, α is a substring of u or v .

Definition 3 (Distinguishability Formulas). Let u, v be strings over some alphabet Σ and r be a natural number. The set of distinguishability formulas from u, v and r is

$$\Phi_{u,v}^r := \Phi_{u,v}^{r, length} \cup \Phi_{u,v}^{r, pref} \cup \Phi_{u,v}^{r, suff} \cup \Phi_{u,v}^{r, sub}.$$

where

$$\begin{aligned} \Phi_{u,v}^{r,length} &:= \left\{ \varphi_{\leq n}^{d(min,max)} \mid |u| < |v|, |u| - 1 \leq n \leq \min(2^r, |v| - 2) \right\} \cup \\ &\quad \left\{ \varphi_{\geq n}^{d(min,max)} \mid |u| > |v|, |v| \leq n \leq \min(2^r + 1, |u| - 1) \right\} \\ \Phi_{u,v}^{r,pref} &:= \left\{ \varphi_{pref_k=pref_k(u)} \mid pref_k(u) \neq pref_k(v), k \leq \min(2^r, |u|, |v|) \right\} \cup \\ &\quad \left\{ \varphi_{pref_k \neq pref_k(v)} \mid pref_k(u) \neq pref_k(v), k \leq \min(2^r, |u|, |v|) \right\} \\ \Phi_{u,v}^{r,suff} &:= \left\{ \varphi_{suff_k=suff_k(u)} \mid suff_k(u) \neq suff_k(v), k \leq \min(2^r, |u|, |v|) \right\} \cup \\ &\quad \left\{ \varphi_{suff_k \neq suff_k(v)} \mid suff_k(u) \neq suff_k(v), k \leq \min(2^r, |u|, |v|) \right\} \\ \Phi_{u,v}^{r,sub} &:= \left\{ \varphi_{\sigma(\alpha) \geq n} \mid \sigma(\alpha, u) > \sigma(\alpha, v), \sigma(\alpha, v) < n \leq \min(r - q_\alpha + 1, \sigma(\alpha, u)) \right\} \cup \\ &\quad \left\{ \varphi_{\sigma(\alpha) < n} \mid \sigma(\alpha, u) < \sigma(\alpha, v), \sigma(\alpha, u) < n \leq \min(r - q_\alpha + 1, \sigma(\alpha, v)) \right\} \cup \\ &\quad \left\{ \varphi_{\gamma(\alpha) \geq n} \mid \gamma(\alpha, u) > \gamma(\alpha, v), \gamma(\alpha, v) < n \leq \min(r - q_\alpha + 1, \gamma(\alpha, u)) \right\} \cup \\ &\quad \left\{ \varphi_{\gamma(\alpha) < n} \mid \gamma(\alpha, u) < \gamma(\alpha, v), \gamma(\alpha, u) < n \leq \min(r - q_\alpha + 1, \gamma(\alpha, v)) \right\}. \end{aligned}$$

Observe that, given u, v , and r , the size of a formula $\varphi \in \Phi_{u,v}^r$ is $O((|u| + |v|)^2)$, and the number of elements $|\Phi_{u,v}^r|$ in $\Phi_{u,v}^r$ is $O((|u| + |v|)^3)$. This is crucial in order to guarantee that our algorithm runs in polynomial time in the size of the sample. Also, by the definition of distinguishability formulas and Theorem 1, it follows that the Spoiler has a winning strategy in $\mathcal{G}_r(u, v)$ if and only if there exists $\varphi \in \Phi_{u,v}^r$. In what follows, we give examples of distinguishability formulas.

Example 4. Let $u = aaacbbb$, $v = aaabbbb$, and $r = 2$. Therefore, $\varphi_{pref_4=aaac}$, $\varphi_{pref_4 \neq aaab} \in \Phi_{u,v}^r$. Furthermore, $\varphi_{pref_3=aaa}$, $\varphi_{pref_3 \neq aaa} \notin \Phi_{u,v}^r$ because $pref_3(u) = pref_3(v)$. Also, $\varphi_{\gamma(c) \geq 1} \in \Phi_{u,v}^r$ because $\gamma(c, u) > \gamma(c, v)$ and $\gamma(c, v) < 1 \leq \min(2, 1)$. Besides, $\varphi_{\gamma(bbb) < 1} \in \Phi_{u,v}^r$ because $\gamma(bbb, u) < \gamma(bbb, v)$ and $\gamma(bbb, u) < 1 \leq \min(1, 1)$. With respect to the length, $\Phi_{u,v}^{r,length} = \emptyset$ because $n > \min(2^r, 6)$.

Example 5. Now, let $u = bbaaaaaabb$, $v = bbaaaaaabb$, and $r = 4$. Then, $\varphi_{\sigma(aaa) \geq 2} \in \Phi_{u,v}^r$ as $\sigma(aaa, u) = 2$, $\sigma(aaa, v) = 1$, and $\sigma(aaa, v) < n \leq \min(3, 2)$. Besides, $\varphi_{\geq 10}^{d(min,max)} \in \Phi_{u,v}^r$ because $|u| > |v|$ and $9 \leq 10 \leq \min(16, 11)$.

Now, we show results ensuring adequate properties of distinguishability formulas.

Lemma 1. Let u, v be strings and r be a natural number. Let $\varphi \in \Phi_{u,v}^r$. Then, $u \models \varphi$ and $v \not\models \varphi$.

Proof. First, suppose $\varphi = \varphi_{\leq n}^{d(min,max)}$. Then, $|u| < |v|$ and $|u| - 1 \leq n \leq \min(2^r, |v| - 2)$. As $|u| - 1 \leq n$, then $d(\min^u, \max^u) \leq n$. Therefore, $u \models \varphi$. Clearly, $n \leq |v| - 2$ because $n \leq \min(2^r, |v| - 2)$. Then, $n \leq d(\min^v, \max^v) - 1$. Therefore, $v \not\models \varphi$. The case in which $\varphi = \varphi_{\geq n}^{d(min,max)}$ is similar.

Now, let $\varphi = \varphi_{pref_k=pref_k(u)}$. Then, $pref_k(u) \neq pref_k(v)$. It also holds that $k \leq \min(2^r, |u|, |v|)$. As $k \leq |u|$, $k \leq |v|$, and $pref_k(u) \neq pref_k(v)$, then $u \models \varphi$ and $v \not\models \varphi$. The cases in which $\varphi = \varphi_{pref_k \neq pref_k(v)}$, $\varphi = \varphi_{suff_k=suff_k(v)}$, and $\varphi = \varphi_{suff_k \neq suff_k(v)}$ are similar.

Next, let $\varphi = \varphi_{\gamma(\alpha) \geq n}$. Thus, $\gamma(\alpha, v) < \gamma(\alpha, u)$ and $\gamma(\alpha, v) < n \leq \min(r - q_\alpha + 1, \gamma(\alpha, u))$. Therefore, $\gamma(\alpha, v) < n \leq \gamma(\alpha, u)$. Then, $u \models \varphi$ and $v \not\models \varphi$. The cases in which $\varphi = \varphi_{\gamma(\alpha) < n}$, $\varphi = \varphi_{\sigma(\alpha) \geq n}$, and $\varphi = \varphi_{\sigma(\alpha) < n}$ are analogous. \square

Lemma 2. Let u, v be strings and r be a natural number. Let $\varphi \in \Phi_{u,v}^r$. Then, $EFsim(u, v) \leq qr(\varphi) \leq r$.

Proof. From Lemma 1, it follows that $EFsim(u, v) \leq qr(\varphi)$. Now, we need to show that $qr(\varphi) \leq r$.

If $\varphi = \varphi_{\leq n}^{d(min,max)}$ where $\leq \in \{\leq, \geq\}$, then $n \leq 2^r$. Hence, $qr(\varphi) = \lceil \log_2(n) \rceil$. It follows that $qr(\varphi) \leq \lceil \log_2(2^r) \rceil = r$.

Let $\varphi \in \{\varphi_{pref_k \blacktriangleright w}, \varphi_{suff_k \blacktriangleright w}\}$ where $\blacktriangleright \in \{=, \neq\}$. Then, $k \leq 2^r$. Therefore, $qr(\varphi) = \lceil \log_2(k) \rceil \leq r$.

Finally, if $\varphi \in \{\varphi_{\gamma(\alpha) \triangleright n}, \varphi_{\sigma(\alpha) \triangleright n}\}$ where $\triangleright \in \{<, \geq\}$, then $n \leq r - q_\alpha + 1$. Thus, $qr(\varphi) = q_\alpha + n - 1 \leq r$. \square

Distinguishability formulas consist of a boolean combination of sentences $\varphi_{\leq n}^{d(\min, \max)}$, $\varphi_{pref_k=a_1 \dots a_k}$, $\varphi_{suff_k=a_1 \dots a_k}$, $\varphi_{\gamma(\alpha) \geq n}$, and $\varphi_{\sigma(\alpha) \geq n}$. Then, the following result ensures an important property of distinguishability formulas.

Proposition 1. *Given a string w and ψ a boolean combination of distinguishability formulas, one can check if $w \models \psi$ in polynomial time.*

Proof. We need to show that given a string w and a distinguishability formula φ , we can check if $w \models \varphi$ in polynomial time. From that, it follows directly that it also holds for any boolean combination of such formulas.

First, let $\varphi = \varphi_{\leq n}^{d(\min, \max)}$. It is possible to check in linear time in the size of w whether $|w| \leq n + 1$.

Next, let $\varphi = \varphi_{pref_k=a_1 \dots a_k}$. Clearly, it is also possible to check in linear time in the size of w whether $pref_k(w) = a_1 \dots a_k$. The case in which $\varphi = \varphi_{suff_k=a_1 \dots a_k}$ is analogous.

Now, let $\varphi = \varphi_{\gamma(\alpha) \geq n}$. At each position of w , it is necessary to check if α occurs free and centered in that position. Then, it takes polynomial time in the size of w .

Finally, let $\varphi = \varphi_{\sigma(\alpha) \geq n}$. First, it is necessary to compute a $2^{q\alpha}$ -segmentation of the set of free occurrences of α . This takes polynomial time in the size of w . Then, it suffices to compare the number of partitions with n . Thus, it takes polynomial time to check whether $w \models \varphi_{\sigma(\alpha) \geq n}$.

Therefore, if ψ is a boolean combination of distinguishability formulas, then it takes polynomial to check whether $w \models \psi$. \square

Distinguishability formulas are also representative for the set of first-order sentences over successor string structures. For example, let $\varphi = \exists x(P_a(x) \wedge \forall y(x \neq y \rightarrow \neg P_a(y)))$. Also let $u = bbabb$, $v_1 = bbbbb$, $v_2 = bbabbabb$, and $r = 2$. Therefore, $\varphi_{\gamma(a) \geq 1} \in \Phi_{u, v_1}^r$ and $\varphi_{\gamma(a) < 2} \in \Phi_{u, v_2}^r$. Thus, φ is equivalent to $\varphi_{\gamma(a) \geq 1} \wedge \varphi_{\gamma(a) < 2}$. Now we will show that this holds for any first-order sentence over strings in our setting. First, we define formulas equivalent to Hintikka formulas.

$$\begin{aligned} \varphi_w^{r, length} &:= \begin{cases} \varphi_{=d(\min^w, \max^w)}^{d(\min, \max)}, & \text{if } |w| \leq 2^r + 1 \\ \varphi_{>2^r}^{d(\min, \max)}, & \text{otherwise.} \end{cases} \\ \varphi_w^{r, pref} &:= \varphi_{pref_{2^r} = pref_{2^r}(w)} \\ \varphi_w^{r, suff} &:= \varphi_{suff_{2^r} = suff_{2^r}(w)} \\ \varphi_w^{r, \alpha} &:= \begin{cases} \varphi_{\sigma(\alpha) = \sigma(\alpha, w)} \wedge \varphi_{\gamma(\alpha) = \gamma(\alpha, w)}, & \text{if } q\alpha + \sigma(\alpha, w) \leq r \\ \varphi_{\sigma(\alpha) \geq r - q\alpha + 1}, & \text{otherwise.} \end{cases} \\ \varphi_w^{r, sub} &:= \bigwedge \{ \varphi_w^{r, \alpha} \mid |\alpha| = 2^q - 1, q > 0 \}. \end{aligned}$$

Lemma 3. $\models \varphi_w^r \leftrightarrow (\varphi_w^{r, length} \wedge \varphi_w^{r, pref} \wedge \varphi_w^{r, suff} \wedge \varphi_w^{r, sub})$.

Proof. Let $u \models \varphi_w^r$. Then, the Duplicator has a winning strategy in $\mathcal{G}_r(w, u)$ and the conditions of Theorem 3 hold. Then, $d(\min^w, \max^w) = d(\min^u, \max^u)$ or $d(\min^w, \max^w) > 2^r$ and $d(\min^u, \max^u) > 2^r$. We have two cases depending on the size of w :

1. $|w| \leq 2^r + 1$. Then, $d(\min^w, \max^w) = d(\min^u, \max^u)$ and it follows that $u \models \varphi_w^{r, length}$.
2. $|w| > 2^r + 1$. Then, $d(\min^w, \max^w) > 2^r$ and $d(\min^u, \max^u) > 2^r$. Therefore, $u \models \varphi_w^{r, length}$.

From Theorem 3, it also holds that $pref_{2^r}(w) = pref_{2^r}(u)$ and $suff_{2^r}(w) = suff_{2^r}(u)$. Then, $u \models \varphi_w^{r, pref} \wedge \varphi_w^{r, suff}$.

From condition 3 of Theorem 3, it holds that $\sigma(\alpha, w) + q_\alpha > r$ and $\sigma(\alpha, u) + q_\alpha > r$ for all α such that $|\alpha| = 2^{q_\alpha} - 1$ and $\sigma(\alpha, w) \neq \sigma(\alpha, u)$ or $\gamma(\alpha, w) \neq \gamma(\alpha, u)$. Let α such that $|\alpha| = 2^q - 1$. We have two cases:

1. $q + \sigma(\alpha, w) \leq r$. Then, $\sigma(\alpha, w) = \sigma(\alpha, u)$ or $\gamma(\alpha, w) = \gamma(\alpha, u)$. Therefore, $u \models \varphi_w^{r, \alpha}$.
2. $q + \sigma(\alpha, w) > r$. If $\sigma(\alpha, w) + q \leq r$, then $\sigma(\alpha, w) \neq \sigma(\alpha, u)$. Therefore, $\sigma(\alpha, u) + q > r$. Hence, $u \models \varphi_w^{r, \alpha}$. Finally, $u \models \varphi_w^{r, sub}$.

Conversely, let $u \models \varphi_w^{r, length} \wedge \varphi_w^{r, pref} \wedge \varphi_w^{r, suff} \wedge \varphi_w^{r, sub}$. We have that $pref_{2^r}(w) = pref_{2^r}(u)$ and $suff_{2^r}(w) = suff_{2^r}(u)$ because $u \models \varphi_w^{r, pref} \wedge \varphi_w^{r, suff}$. We also have that $u \models \varphi_w^{r, length}$. We have two cases:

1. $|w| \leq 2^r + 1$. Then, $d(\min^w, \max^w) = d(\min^u, \max^u)$.
2. $|w| > 2^r + 1$. Then, $d(\min^w, \max^w) > 2^r$ and $d(\min^u, \max^u) > 2^r$.

It also holds that $u \models \varphi_w^{r, sub}$. Let α such that $|\alpha| = 2^q - 1$ and $\sigma(\alpha, w) \neq \sigma(\alpha, u)$ or $\gamma(\alpha, w) \neq \gamma(\alpha, u)$. Then, $u \models \varphi_w^{r, \alpha}$. We have that $\sigma(\alpha, w) + q > r$, otherwise, $\sigma(\alpha, w) = \sigma(\alpha, u)$ and $\gamma(\alpha, w) = \gamma(\alpha, u)$. Therefore, $\sigma(\alpha, u) + q > r$. Then, for all α such that $|\alpha| = 2^q - 1$ and $\sigma(\alpha, w) \neq \sigma(\alpha, u)$ or $\gamma(\alpha, w) \neq \gamma(\alpha, u)$, we have that $\sigma(\alpha, w) + q > r$ and $\sigma(\alpha, u) + q > r$. All the conditions of Theorem 3 hold. Then, the Duplicator has a winning strategy in $\mathcal{G}_r(w, u)$. It follows that $u \models \varphi_w^r$. \square

Now, we need the following lemmas.

Lemma 4. *Let r be a natural number and w a string. There is a set of strings V_{length} such that $\varphi_w^{r, length}$ is equivalent to a boolean combination of sentences in $\bigcup_{v \in V_{length}} \Phi_{w, v}^r$.*

Proof. If $\varphi_w^{r, length} = \varphi_{>2^r}^{d(\min, \max)}$, then let $V_{length} = \{v\}$ such that $|v| = 2^r + 1$. Then, $d(\min^v, \max^v) = 2^r$. Observe that $\models \varphi_w^{r, length} \leftrightarrow \varphi_{\geq 2^r+1}^{d(\min, \max)}$ and $\varphi_{\geq 2^r+1}^{d(\min, \max)} \in \Phi_{w, v}^r$ because $|w| > |v|$ and $|v| \leq 2^r + 1 \leq \min(2^r + 1, |u| - 1)$. If $\varphi_w^{r, length} = \varphi_{=d(\min^w, \max^w)}^{d(\min, \max)}$, then let $V_{length} = \{v_1, v_2\}$ such that $|v_1| = |w| - 1$ and $|v_2| = |w| + 1$. Thus, $|v_1| < |w|$ and $|v_1| \leq d(\min^w, \max^w) \leq |w| - 1$. It follows that $\varphi_{\geq d(\min^w, \max^w)}^{d(\min, \max)} \in \Phi_{w, v_1}^r$. We also have that $|v_2| > |w|$ and $|w| - 1 \leq d(\min^w, \max^w) \leq |v_2| - 2$. Therefore, $\varphi_{\leq d(\min^w, \max^w)}^{d(\min, \max)} \in \Phi_{w, v_2}^r$. Obviously, $\varphi_w^{r, length}$ is equivalent to $\varphi_{\geq d(\min^w, \max^w)}^{d(\min, \max)} \wedge \varphi_{\leq d(\min^w, \max^w)}^{d(\min, \max)}$. \square

Lemma 5. *Let r be a natural number and w a string. There is a set of strings $V_{pref, suff}$ such that $\varphi_w^{r, pref} \wedge \varphi_w^{r, suff}$ is equivalent to a boolean combination of sentences in $\bigcup_{v \in V_{pref, suff}} \Phi_{w, v}^r$.*

Proof. Let v_1 such that $|v_1| = |w|$ and $v_1 \neq w$. If $|w| \leq 2^r$, then it follows that $\models \varphi_{pref_{2^r}=pref_{2^r}(w)} \leftrightarrow \varphi_{pref_{|w|=w}}$. Hence, $\varphi_{pref_{|w|=w}} \in \Phi_{w, v_1}^r$. If $|w| > 2^r$, then $\varphi_{pref_{2^r}=pref_{2^r}(w)} \in \Phi_{w, v_1}^r$. Obviously, the case for $\varphi_{suff_{2^r}=suff_{2^r}(w)}$ is analogous. Let v_2 such that $\varphi_{suff_{2^r}=suff_{2^r}(w)} \in \Phi_{w, v_2}^r$. Therefore, $V_{pref, suff} = \{v_1, v_2\}$. \square

Lemma 6. *Let r be a natural number and w, α strings. There is a set of strings V_α such that $\varphi_w^{r, \alpha}$ is equivalent to a boolean combination of sentences in $\bigcup_{v \in V_\alpha} \Phi_{w, v}^r$.*

Proof. If $\varphi_w^{r, \alpha} = \varphi_{\sigma(\alpha) \geq r - q_\alpha + 1}$, then $\sigma(\alpha, w) \geq r - q_\alpha + 1$. Let $V_\alpha = \{v\}$ such that $\sigma(\alpha, w) > \sigma(\alpha, v)$ and $\sigma(\alpha, v) < r - q_\alpha + 1$. Thus, $\varphi_{\sigma(\alpha) \geq r - q_\alpha + 1} \in \Phi_{w, v}^r$. If $\varphi_w^{r, \alpha} = \varphi_{\sigma(\alpha) = \sigma(\alpha, w)} \wedge \varphi_{\gamma(\alpha) = \gamma(\alpha, w)}$, then $\sigma(\alpha, w) \leq r - q_\alpha$. For $\varphi_{\sigma(\alpha) = \sigma(\alpha, w)}$, let v_1 such that $\sigma(\alpha, w) > \sigma(\alpha, v_1)$. Then, $\varphi_{\sigma(\alpha) \geq \sigma(\alpha, w)} \in \Phi_{w, v_1}^r$. Let v_2 such that $\sigma(\alpha, v_2) > \sigma(\alpha, w)$. Note that $\sigma(\alpha, w) < \sigma(\alpha, w) + 1 \leq \min(r - q_\alpha + 1, \sigma(\alpha, v_2))$. Then, $\varphi_{\sigma(\alpha) < \sigma(\alpha, w) + 1} \in \Phi_{w, v_1}^r$. Clearly, the case for $\varphi_{\gamma(\alpha) = \gamma(\alpha, w)}$ is analogous. Let v_3 and v_4 such that $\varphi_{\gamma(\alpha) \geq \gamma(\alpha, w)} \in \Phi_{u, v_3}^r$ and $\varphi_{\gamma(\alpha) < \gamma(\alpha, w) + 1} \in \Phi_{u, v_4}^r$. Therefore $V_\alpha = \{v_1, v_2, v_3, v_4\}$. \square

Lemma 7. *Let r be a natural number and w a string. There is a set of strings V such that φ_w^r is a boolean combination of sentences in $\bigcup_{v \in V} \Phi_{w,v}^r$.*

Proof. By Lemma 3, $\models \varphi_w^r \leftrightarrow (\varphi_w^{r,length} \wedge \varphi_w^{r,pref} \wedge \varphi_w^{r,suff} \wedge \varphi_w^{r,sub})$. Let $V = \bigcup_{\{\alpha \mid |\alpha|=2q-1, q>0\}} V_\alpha \cup V_{length} \cup V_{pref,suff} \cup$ as in Lemma 4, Lemma 5, and Lema 6. From these lemmas, it follows that, φ_w^r is equivalent to a boolean combination of sentences in $\bigcup_{v \in V} \Phi_{w,v}^r$. \square

The next result is related to Theorem 2. It suggests that our approach is likely to find any first-order sentence given a suitable sample of strings. Recall that, by Theorem 2, any first-order sentence is equivalent to a disjunction of Hintikka formulas. Thus, we have the following result.

Theorem 4. *Let φ be a first-order sentence over successor string structures. Then, φ is equivalent to a boolean combination of distinguishability formulas.*

Proof. Let r such that $qr(\varphi) = r$. From Theorem 2 it follows that $\models \varphi \leftrightarrow \varphi_{u_1}^r \vee \dots \vee \varphi_{u_s}^r$. Let $U = \{u_1, \dots, u_s\}$. In accord to Lemma 7, let V_i be such that $\varphi_{u_i}^r$ is equivalent to a boolean combination of sentences in $\bigcup_{v \in V_i} \Phi_{u_i,v}^r$. Therefore, the sentence φ is equivalent to a boolean combination of sentences in $\bigcup_{i=1}^s (\bigcup_{v \in V_i} \Phi_{u_i,v}^r)$. \square

4 The Algorithm and Its Analysis

In this section, we define an algorithm for finding a first-order sentence φ_S from a sample of strings S . Subformulas of φ_S are distinguishability formulas from sets of the form $\Phi_{u,v}^r$ such that $u \in P$ and $v \in N$. We also give an example of how the algorithm works. We guarantee that our algorithm runs in polynomial time in the size of the input sample S . The size of the sample S is the sum of the lengths of all strings it includes. We use $|S|$ to denote the size of the sample S . We also show that φ_S returned by our algorithm is consistent with S . Furthermore, we also prove that φ_S is a sentence of minimal quantifier rank consistent with S . The pseudocode of our algorithm is in Algorithm 1.

Algorithm 1

Input: Sample of strings $S = (P, N)$
 $r \leftarrow \max\{EFsim(u, v) \mid u \in P, v \in N\}$
 $\varphi_S \leftarrow \bigvee_{u \in P} \bigwedge_{v \in N} \mathbf{choose} \varphi \in \Phi_{u,v}^r$
return φ_S

First, the algorithm finds the minimum value r such that there exists a sentence of quantifier rank r that is consistent with the input sample S . After that, the algorithm constructs φ_S . It goes through all strings in $P \cup N$, and, for $u \in P, v \in N$, it chooses a formula $\varphi \in \Phi_{u,v}^r$. For each $u \in P$, Algorithm 1 builds a conjunction of sentences in $\bigcup_{v \in N} \Phi_{u,v}^r$. Finally, it returns a disjunction of such conjunctions. In the following, we show an example of how this algorithm works on a simple instance.

Example 6. *Let S be the sample in Table 1. Note that $\max\{EFsim(u, v) \mid u \in P, v \in N\} = 1$ as witnessed by $\sigma(p, stviie) + 1 \leq 1$ and $\sigma(p, stpiie) + 1 > 1$. Clearly, $\varphi_{pref_1=s} \in \Phi_{stviil,ktvive}^1$, $\varphi_{suff_1 \neq e} \in \Phi_{stviil,stpiie}^1$, $\varphi_{pref_1=s} \in \Phi_{stviie,ktvive}^1$, and $\varphi_{\sigma(p)<1} \in \Phi_{stviie,stpiie}^1$. Therefore, Algorithm 1 returns φ_S below. Observe that φ_S is consistent with S and $qr(\varphi_S) = 1$.*

$$\varphi_S = (\varphi_{pref_1=s} \wedge \varphi_{suff_1 \neq e}) \vee (\varphi_{pref_1=s} \wedge \varphi_{\sigma(p)<1}).$$

In the following, we prove the correctness and the time complexity of our algorithm. First, we show that it returns a sentence that is consistent with the sample. After that, we show that it returns a sentence of minimal quantifier rank. Then, we prove that the running time of our learning algorithm is polynomial in the size of the given sample.

Theorem 5. *Let S be a sample and φ_S returned by Algorithm 1. Then, φ_S is consistent with S .*

Proof. Let $u \in P$. Then, $\varphi_u = \varphi_1 \wedge \dots \wedge \varphi_k$ such that, for all i , $\varphi_i \in \Phi_{u,v}^r$, for some $v \in N$. In this way, $u \models \varphi_i$, for all i and then $u \models \varphi_S$. Now, let $v \in N$ and assume that $v \models \varphi_S$, i.e., $v \models \varphi_u$, for some $u \in P$. Therefore, φ_u has a conjunct $\varphi \in \Phi_{u,v}^r$. This is an absurd because, from Lemma 1, it follows that $v \not\models \varphi$. \square

Theorem 6. *The sentence φ_S returned by Algorithm 1 is a first-order sentence of minimal quantifier rank that is consistent with S .*

Proof. Suppose a first-order sentence ψ consistent with S such that $qr(\psi) < qr(\varphi_S) = \max\{EFsim(u, v) \mid u \in P, v \in N\}$. Let $u' \in P$ and $v' \in N$ such that $EFsim(u', v') = \max\{EFsim(u, v) \mid u \in P, v \in N\}$. Then, u' and v' are satisfied by the same first-order sentences of quantifier rank q such that $q < EFsim(u', v')$. Then, $u' \models \psi$ iff $v' \models \psi$. Therefore, ψ is not consistent with S . This is a contradiction. \square

Theorem 7. *Given a sample S , Algorithm 1 returns φ_S in time $O(|S|^7)$.*

Proof. First, the algorithm computes $\max\{EFsim(u, v) \mid u \in P, v \in N\}$ in order to use a suitable quantifier rank. This takes time $O(|S|^4 \times \log(|S|))$ because, for a given $u \in P, v \in N$, $|u| + |v| < |S|$, to compute $EFsim(u, v)$ takes time $O(|S|^2 \times \log(|S|))$, and this procedure is executed $|S|^2$ times. Then, our algorithm loops over strings in the sample and, in each loop, it chooses a formula $\varphi \in \Phi_{u,v}^r$. As the size of each $\varphi \in \Phi_{u,v}^r$ is $O(|S|^2)$ and $|\Phi_{u,v}^r|$ is $O(|S|^3)$, one iteration of the loop runs in time $O(|S|^5)$. This loop is executed $O(|S|^2)$ times, then, this loop takes time $O(|S|^7)$. The first step runs in time $O(|S|^4 \log(|S|))$ and the rest takes time $O(|S|^7)$. Therefore the overall complexity of Algorithm 1 is $O(|S|^7)$. \square

Therefore, our algorithm is an improvement over the work in [13], for this particular problem on successor string structures. However, observe that the sentence returned by Algorithm 1 is a disjunction of $|P|$ conjunctions of distinguishability formulas. The algorithm in [13] also returns formulas which are long and hard to read. Then, they greedily remove subformulas that are not necessary. Now, we also consider the number of conjunctions in our approach. Let $\Phi_S := \bigcup_{u \in P, v \in N} \Phi_{u,v}^{\max\{EFsim(u,v) \mid u \in P, v \in N\}}$. We say that a first-order formula is in m -DDF (Disjunctive Distinguishability Form) over Φ_S if it is a disjunction of m conjunctions of distinguishability formulas in Φ_S . Therefore, we can also define a problem where the goal is to find a formula φ_S in m -DDF such that m is minimum. This formula φ_S improves interpretability. Given a sample S , the problem consists of finding a first-order sentence φ_S in m -DDF over Φ_S such that φ_S is consistent with S , and m is minimum.

An algorithm to return a first-order sentence given a sample of strings and a set of first-order sentences is presented in [18]. Formally, given a sample of strings S and a set of first-order sentences Φ , the goal is to find a first-order sentence φ such that φ is consistent with S , φ is a disjunction of conjunctions of sentences in Φ , and the number of conjunctions is minimum. This problem is NP-complete. It is easy to polynomially reduce our problem of distinguishability formulas to this problem in [18]. Then, our problem of distinguishability formulas is in NP, and it is still a better approach than the one in [13],

for the particular case of successor string structures. In this new setting, the formula for the sample of Table 1 is below. This formula is smaller than the formula of Example 6.

$$\varphi_S = \varphi_{\text{pref}_1=s} \wedge \varphi_{\gamma(v) \geq 1}.$$

5 Conclusions and Future Work

Motivated by the framework defined in [13] and results of the Ehrenfeucht–Fraïssé Game over successor string structures in [14], we introduced an algorithm that returns a first-order sentence of minimal quantifier rank that is consistent with a given sample of strings. Our algorithm runs in time $O(|S|^7)$ where $|S|$ is the size of the input sample. Then, our algorithm runs in polynomial time in the size of S . The algorithm in [13] runs in exponential time as it works for arbitrary structures.

We designed our algorithm using distinguishability formulas which are defined based on the conditions characterizing a winning strategy for the Spoiler on successor string structures. Our algorithm returns a disjunction of conjunctions of distinguishability formulas. The size of a distinguishability formula is polynomial in the length of two given strings. The algorithm in [13] uses Hintikka formulas which have size exponential in the size of a given string. Therefore, our proposed algorithm is an improvement over the one in [13], for successor string structures. We also show that any first-order sentence is equivalent to a boolean combination of distinguishability formulas. Then, our approach has the potential to find any first-order sentence given the right sample. We also showed how to find a formula with the minimum number of conjunctions. A small formula is preferable for explaining a sample of strings.

Our results are also relevant to grammatical inference, where the goal is to find a language model of minimal size that describes a given sample of strings. In our framework, the language model is the first-order logic over successor string structures, and we use the quantifier rank as a natural measure of first-order sentences. As strings may be used to model sequences of symbolic data, our results can be applied in the analysis of biological sequences [22]. A recent model-theoretic approach to grammatical inference [20] uses a fragment of first-order logic which is less expressive than full first-order logic in our approach.

As future work, we intend to explore the problem of finding a first-order sentence over strings with the linear order relation. First-order logic over strings with the linear order defines the class of star-free languages [21], and it is more expressive than first-order logic over successor string structures. Finally, we plan to extend our approach to monadic second-order logic. Regular languages are exactly the languages definable in monadic second-order logic [1]. An algorithm which returns monadic second-order sentences can be used in the problem of finding a finite automaton consistent with a given sample of strings.

References

- [1] Richard J. Büchi (1960): *Weak Second-Order Arithmetic and Finite Automata*. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 6(1-6), pp. 66–92, doi:10.1002/malq.1960060105.
- [2] Luc De Raedt (2008): *Logical and relational learning*. Springer Science & Business Media, doi:10.1007/978-3-540-68856-3.
- [3] Heinz-Dieter Ebbinghaus & Jörg Flum (1995): *Finite model theory*. Perspectives in Mathematical Logic, Springer, doi:10.1007/978-3-662-03182-7.
- [4] Heinz-Dieter Ebbinghaus, Jörg Flum & Wolfgang Thomas (1994): *Mathematical Logic (2. ed.)*. Undergraduate texts in mathematics, Springer, doi:10.1007/978-1-4757-2355-7.

- [5] Andrzej Ehrenfeucht (1961): *An application of games to the completeness problem for formalized theories*. *Fundamenta Mathematicae* 49(2), pp. 129–141, doi:10.4064/fm-49-2-129-141.
- [6] Erich Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, Moshe Y. Vardi, Y. Venema & Scott Weinstein (2005): *Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, doi:10.1007/3-540-68804-8.
- [7] Martin Grohe, Christof Löding & Martin Ritzert (2017): *Learning MSO-definable hypotheses on strings*. In: *International Conference on Algorithmic Learning Theory*, pp. 434–451.
- [8] Martin Grohe & Martin Ritzert (2017): *Learning first-order definable concepts over structures of small degree*. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2017)*, IEEE, pp. 1–12, doi:10.1109/LICS.2017.8005081.
- [9] Colin De la Higuera (2010): *Grammatical inference: learning automata and grammars*. Cambridge University Press, doi:10.1017/CBO9781139194655.
- [10] John E. Hopcroft, Rajeev Motwani & Jeffrey D. Ullman (2006): *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [11] Charles Jordan & Łukasz Kaiser (2013): *Experiments with reduction finding*. In: *International Conference on Theory and Applications of Satisfiability Testing, Lecture Notes in Computer Science 7962*, Springer, pp. 192–207, doi:10.1007/978-3-642-39071-5_15.
- [12] Charles Jordan & Łukasz Kaiser (2013): *Learning programs as logical queries*. In: *Workshop on Learning Theory and Complexity, LTC*.
- [13] Łukasz Kaiser (2012): *Learning Games from Videos Guided by Descriptive Complexity*. In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI'12*, AAAI Press, pp. 963–969.
- [14] Angelo Montanari, Alberto Policriti & Nicola Vitacolonna (2005): *An algorithmic account of Ehrenfeucht games on labeled successor structures*. In: *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, Springer, pp. 139–153, doi:10.1007/11591191_11.
- [15] Stephen Muggleton (1991): *Inductive logic programming*. *New generation computing* 8(4), pp. 295–318, doi:10.1007/BF03037089.
- [16] Elena Pezzoli (1998): *Computational complexity of Ehrenfeucht-Fraïssé games on finite structures*. In: *International Workshop on Computer Science Logic*, Springer, pp. 159–170, doi:10.1007/10703163_11.
- [17] Thiago Alves Rocha, Ana Teresa Martins & Francicleber Martins Ferreira (2018): *On Distinguishing Sets of Structures by First-Order Sentences of Minimal Quantifier Rank*. In: *13th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2018)*, p. to appear.
- [18] Thiago Alves Rocha, Ana Teresa Martins & Francicleber Martins Ferreira (2018): *Synthesis of a DNF Formula from a Sample of Strings*. In: *7th Brazilian Conference on Intelligent Systems (BRACIS 2018)*, IEEE, p. to appear.
- [19] James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert & Sean Wibel (2013): *Cognitive and sub-regular complexity*. In: *Formal grammar*, Springer, pp. 90–108, doi:10.1007/978-3-642-39998-5_6.
- [20] Kristina Strother-Garcia, Jerrey Heinz & Hyun Jin Hwangbo (2017): *Using model theory for grammatical inference: a case study from phonology*. In: *International Conference on Grammatical Inference*, pp. 66–78.
- [21] Wolfgang Thomas (1982): *Classifying regular events in symbolic logic*. *Journal of Computer and System Sciences* 25(3), pp. 360–376, doi:10.1016/0022-0000(82)90016-2.
- [22] Wojciech Wieczorek & Olgierd Unold (2014): *Induction of directed acyclic word graph in a bioinformatics task*. In: *International Conference on Grammatical Inference*, pp. 207–217.
- [23] Marc Zeitoun, Lorijn van Rooijen & Thomas Place (2014): *On Separation by Locally Testable and Locally Threshold Testable Languages*. *Logical Methods in Computer Science* 10, doi:10.2168/LMCS-10(3:24)2014.