UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO ALVES ROCHA

SYNTHESIS OF FIRST-ORDER SENTENCES USING EHRENFEUCHT–FRAÏSSÉ GAMES AND BOOLEAN SATISFIABILITY

FORTALEZA

2019

THIAGO ALVES ROCHA

SYNTHESIS OF FIRST-ORDER SENTENCES USING EHRENFEUCHT–FRAÏSSÉ GAMES AND BOOLEAN SATISFIABILITY

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Lógica e Inteligência Artificial

Orientadora: Profa. Dra. Ana Teresa de Castro Martins

Coorientador: Prof. Dr. Francicleber Martins Ferreira

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R577s Rocha, Thiago Alves.
Synthesis of First-Order Sentences using Ehrenfeucht–Fraïssé Games and Boolean Satisfiability / Thiago Alves Rocha. – 2019.
136 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2019.
Orientação: Profa. Dra. Ana Teresa de Castro Martins.
Coorientação: Prof. Dr. Francicleber Martins Ferreira.

1. Síntese de Fórmulas. 2. Inferência Gramatical. 3. Jogo Ehrenfeucht–Fraïssé. I. Título.
CDD 005

THIAGO ALVES ROCHA

SYNTHESIS OF FIRST-ORDER SENTENCES USING EHRENFEUCHT–FRAÏSSÉ GAMES AND BOOLEAN SATISFIABILITY

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Lógica e Inteligência Artificial

Aprovada em: 30 de Agosto de 2019

BANCA EXAMINADORA

Profa. Dra. Ana Teresa de Castro Martins (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Francicleber Martins Ferreira (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Fernando Lima Alcântara
Universidade Federal do Ceará (UFC)

Prof. Dr. Ruy José Guerra Barretto de Queiroz
Universidade Federal de Pernambuco (UFPE)

Prof. Dr. Edward Herman Haeusler
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

RESUMO

Neste trabalho, nós investigamos o problema da síntese de sentenças de primeira-ordem a partir de amostras de estruturas relacionais classificadas. Em outras palavras, nós consideramos o seguinte problema: para uma classe de estruturas relacionais fixa, dada uma amostra de estruturas classificadas, encontrar uma sentença de primeira-ordem de *quantifier rank* mínimo que é consistente com a amostra. Nós contemplamos as seguintes classes de estruturas: estruturas monádicas, estruturas de equivalência, uniões disjuntas de ordens lineares e *strings* representadas por estruturas finitas com uma relação de successor. Nós usamos resultados do jogo Ehrenfeucht–Fraïssé nessas classes de estruturas com a finalidade de projetar um algoritmo para encontrar tal sentença. Para essas classes de estruturas, o problema de determinar se um dos jogadores tem uma estratégia vencedora no jogo Ehrenfeucht–Fraïssé é resolvido em tempo polinomial. Nós também introduzimos sentenças de distinguibilidade que são sentenças que distinguem duas estruturas dadas. Nós definimos as sentenças de distinguibilidade usando condições necessárias e suficientes para uma estratégia vencedora no jogo Ehrenfeucht–Fraïssé. Nosso algoritmo retorna uma combinação Booleana de tais sentenças. Nós também mostramos que qualquer sentença de primeira-ordem é equivalente à uma combinação Booleana de sentenças de distinguibilidade. Finalmente, nós também mostramos que o tempo de execução do nosso algoritmo é polinomial no tamanho da amostra de entrada. Como, em geral, sentenças de primeira-ordem são dificeis de ler, nós definimos uma formal normal livre de quantificadores (QNF – do inglês: quantifier-free normal form) para as classes de estruturas que estamos considerando. Sentenças QNF são definidas com respeito a um vocabulário mais rico tal que sentenças atômicas são abreviações de sentenças arbitrárias de primeira-ordem sobre o vocabulário padrão. Dessa forma, sentenças QNF consistem de combinações Booleanas de tais sentenças atômicas sobre esse vocabulário não-padrão. Além disso, nós definimos uma forma normal disjuntiva (DNF – do inglês: disjunctive normal form) para sentenças QNF. Portanto, dada uma amostra de *strings* classificadas e o número de cláusulas disjuntivas, nós investigamos o problema de encontrar uma sentença QNF na DNF que é consistente com a amostra. Nós provamos que esse problema é NP-completo e mostramos uma solução baseada em uma codificação para o problema da satisfatibilidade Booleana (SAT). Soluções para o problema de encontrar uma sentença QNF na DNF tal que apenas o número de cláusulas é limitado podem ter um número grande de literais por cláusula. Portanto, nós consideramos uma variação desse problema no qual o número máximo de literais por cláusula também é fornecido como entrada. Isso é essencial pois sentenças com

poucas cláusulas e poucos literais por cláusula são mais compactas e mais fáceis de interpretar. Novamente, nós mostramos que esse problema é NP-completo e nossa abordagem para resolvê-lo é baseada na redução para o SAT. Nós também apresentamos extensões desses problemas que são robustas com respeito a amostras ruidosas. Nesse caso, uma sentença pode não ser consistente com a amostra da entrada. Nós cobrimos duas abordagens para lidar com amostras com ruídos. Na primeira abordagem, nós consideramos o problema no qual o objetivo é encontrar uma sentença que classifica corretamente o maior número de *strings*. Nós resolvemos essa versão generalizada através de uma codificação no problema da satisfatibilidade máxima (MaxSAT). Na nossa segunda abordagem, o objetivo é encontrar uma sentença que não classifica corretamente no máximo uma quantidade dada de *strings*. Nós mostramos que esse problema relativo a um número limitado de erros também é NP-completo. Além disso, nós apresentamos uma solução baseada no SAT para resolver esse problema. Dentre as classes que estamos considerando, as *strings* são mais interessantes pois elas podem ser usadas para modelar dados textuais, padrões de tonicidade em línguas humanas, sequências biológicas e sequências de dados simbólicos em geral. Como a lógica de primeira-ordem sobre *strings* define exatamente a classe das linguagens *locally threshold testable* (LTT), nossos resultados podem ser úteis em inferência gramatical quando o propósito é encontrar uma descrição formal de uma linguagem LTT a partir de uma amostra de *strings*. Na área de inferência gramatical, um dos principais problemas estudados é a tarefa de obter um modelo de linguagem consistente com uma amostra de *strings* classificadas.

Palavras-chave: Síntese de Fórmulas. Inferência Gramatical. Jogo Ehrenfeucht–Fraïssé.

ABSTRACT

In this work, we investigate the problem of synthesis of first-order sentences from samples of classified relational structures. In other words, we investigate the following problem: for a fixed class of relational structures, given a sample of classified structures, find a first-order sentence of minimum quantifier rank that is consistent with the sample. We consider the following classes of structures: monadic structures, equivalence structures, disjoint unions of linear orders, and strings represented by finite structures with a successor relation. We use results of the Ehrenfeucht–Fraïssé game on these classes of structures in order to design an algorithm to find such a sentence. For these classes of structures, the problem of determining whether the Duplicator has a winning strategy in an Ehrenfeucht–Fraïssé game is solved in polynomial time. We also introduce the distinguishability sentences, which are sentences that distinguish between two given structures. We define the distinguishability sentences based on necessary and sufficient conditions for a winning strategy in Ehrenfeucht–Fraïssé games. Our algorithm returns a Boolean combination of such sentences. We also show that any first-order sentence is equivalent to a Boolean combination of distinguishability sentences. Finally, we also show that our algorithm's running time is polynomial in the size of the input. Since general first-order sentences are hard to read, we define a quantifier-free normal form (QNF) over the classes of structures we are considering. QNF sentences are defined over a richer vocabulary such that atomic formulas are an abbreviation of general first-order sentences over a standard vocabulary. Then, QNF sentences consist of Boolean combinations of such atomic sentences over this non-standard vocabulary. Moreover, we define a DNF version for QNF sentences. Then, given a sample of strings and the number of disjunctive clauses, we investigate the problem of finding a DNF formula that is consistent with the sample. We show that this problem is NP-complete and we solve it by a translation into Boolean satisfiability (SAT). We also present an extension of this problem that is robust concerning noisy samples. We solve this generalized version by a codification into the maximum satisfiability problem. Solutions to the problem of finding a QNF sentence in DNF such that the number of clauses is bounded may have a large number of literals per clause. Therefore, we consider a variation of this problem in which the maximum number of literals per clause is also given as input. This is essential since sentences with few clauses and few literals per clause are more compact and easier to interpret. Again, we show that this problem is NP-complete, and our approach for solving it is based on a reduction to the SAT. We also present extensions of these problems that are robust concerning noisy samples. In this case, a

sentence may not be consistent with the input sample. We cover two approaches to deal with noisy samples. In the first approach, we consider a problem in which the goal is to find a sentence that classifies the maximum number of strings correctly. We solve this generalized version by a codification into the maximum satisfiability problem (MaxSAT). In our second approach, the goal is to find a sentence such that it does not correctly classify at most a given number of strings. We show that this problem concerning a limited number of errors is also NP-complete. Moreover, we give a SAT-based solution to this problem. Among the classes we are considering, strings are more appealing since they may be used to model text data, stress patterns in human languages, biological sequences, and sequences of symbolic data in general. As first-order logic over strings defines exactly the class of locally threshold testable (LTT) languages, our results can be useful in grammatical inference when the goal is to find a model of an LTT language from a sample of strings. In the field of grammatical inference, one of the main problems studied is the task of finding a language model consistent with a given sample of strings.

Keywords: Formula Synthesis. Grammatical Inference. Ehrenfeucht–Fraïssé Game.

CONTENTS

1 INTRODUCTION

Finite model theory (EBBINGHAUS; FLUM, 1995; LIBKIN, 2004; GRÄDEL *et al.*, 2005) studies the expressive power of logics on finite structures, and its motivation comes from several areas of computer science, such as database theory (ABITEBOUL *et al.*, 1995), computational complexity (IMMERMAN, 1999), and formal languages (ROZENBERG; SALOMAA, 1997). The expressive power of a logic is measured by its ability to define properties.

Ehrenfeucht–Fraïssé games (EF games, for short) (EHRENFEUCHT, 1961) is a fundamental technique of finite model theory to prove the inexpressibility of certain properties in first-order logic (FO). For instance, consider the problem of checking whether a graph has even cardinality. One may use EF games to show that first-order logic is unable to define this property. The EF game is played on two structures by two players, the Spoiler and the Duplicator. If the Spoiler has a winning strategy for r rounds of the game, it means that the structures can be distinguished by a first-order sentence φ whose quantifier rank is at most r, i.e., φ holds in exactly one of these structures.

Besides providing a tool to measure the expressive power of a logic, EF games allow one to investigate the similarity between structures (MONTANARI *et al.*, 2005). In a game played on structures $\mathscr{A}$ and $\mathscr{B}$, the similarity between $\mathscr{A}$ and $\mathscr{B}$ is the minimum number of rounds such that the Spoiler has a winning strategy. Most results in the literature (FAGIN *et al.*, 1995; SCHWENTICK, 1996; ARORA; FAGIN, 1997; KEISLER; LOTFALLAH, 2004) only give sufficient conditions for a winning strategy of the Duplicator. However, the minimum number of rounds cannot be computed only from sufficient conditions. Then, in order to explore this notion of similarity, necessary and sufficient conditions for a winning strategy of the Spoiler must be determined.

Explicit conditions characterizing winning strategies for the players on some standard classes of finite structures are provided in (KHOUSSAINOV; LIU, 2009). Examples of such classes are monadic structures (MS) and equivalence structures (ES). Besides, there are well known necessary and sufficient conditions characterizing the winning strategies on linear orders (LO) (LIBKIN, 2004). Using these results, the similarity can be computed in polynomial time in the size of the structures.

In (MONTANARI *et al.*, 2005), multiplicity and scattering of substrings are used to prove a characterization of winning strategies in EF games on strings represented by finite

structures with a successor relation and a finite number of pairwise disjoint unary predicates. Multiplicity and scattering mean the number and distribution of occurrences of substrings, respectively. 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 result allows one to define a notion of similarity between strings based on EF games.

An algorithm to deal with the problem of finding a formula of minimum quantifier rank that distinguishes two sets of structures over an arbitrary vocabulary is presented in (KAISER, 2012). An important part of this algorithm is the use of Hintikka formulas. Given a structure $\mathscr{A}$ and a natural number r, an r-Hintikka formula $\varphi^r_{\mathscr{A}}$ (HINTIKKA, 1953) is a formula that describes the properties of $\mathscr{A}$ on EF games with r rounds (EBBINGHAUS; FLUM, 1995). An r-Hintikka formula $\varphi^r_{\mathscr{A}}$ holds exactly on all structures $\mathscr{B}$ such that the Duplicator has a winning strategy for the EF game with r rounds on $\mathscr{A}$ and $\mathscr{B}$. This suggests that the similarity between $\mathscr{A}$ and $\mathscr{B}$ is greater than r. Also, $\varphi^r_{\mathscr{A}}$ has size exponential in r. Besides, any first-order formula is equivalent to a disjunction of Hintikka formulas.

Although this algorithm in (KAISER, 2012) works for arbitrary finite relational structures, it runs in exponential time. A general system for learning formulas defining board game rules uses this algorithm. For instance, one can use this framework to automatically obtain the winning conditions of board games, such as Tic-Tac-Toe, Breakthrough, and Connect Four. The formulas are obtained from a set of structures representing winning positions and a set of structures representing non-winning positions. These results in (KAISER, 2012) are also used in finding reductions automatically (JORDAN; KAISER, 2013b; JORDAN; KAISER, 2013a) and learning programs (JORDAN; KAISER, 2013c; JORDAN; KAISER, 2016).

1.1 The Synthesis Problem

In this work, we study a variation of the problem introduced in (KAISER, 2012) where the class of structures is fixed. For a fixed class of structures $\mathscr{C}$, a sample $S = (P,N)$ consists of two finite sets $P,N \subseteq \mathscr{C}$ such that for each $\mathscr{A} \in P$, $\mathscr{B} \in N$, $\mathscr{A}$ and $\mathscr{B}$ are not isomorphic. For a fixed class of structures $\mathscr{C}$, given a sample S of structures in $\mathscr{C}$, the task is to find a first-order sentence φ_S of minimum quantifier rank that is consistent with S. In other words, the sentence 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 sizes of all structures in the sample. We call this problem as the

synthesis of minimum quantifier rank sentence problem.

We define an algorithm for the synthesis problem on the following classes of structures: MS, ES, disjoint unions of linear orders (DULO), and strings. We consider MS, ES, and strings because necessary and sufficient conditions for a winning strategy of the players in an EF game on these classes are provided in the literature (KHOUSSAINOV; LIU, 2009; MONTANARI *et al.*, 2005).

Disjoint unions of linear orders are compelling because we may model a state of the elementary blocks world by using them (COOK; LIU, 2003). In the elementary blocks world, states consist of a set of cubic blocks, with the same size and color, sitting on a table. A robot can pick up a block and moves it to another position, either onto the table or on the top of some other block (GUPTA; NAU, 1992). The blocks world is one of the most famous planning domains in artificial intelligence. Automated planning is the process of automatically constructing a sequence of actions that achieve a goal given some initial state (GHALLAB *et al.*, 2004). For disjoint unions of linear orders, we give conditions for winning strategies.

We use these characterization results on EF games in order to design an algorithm to find a sentence of minimum quantifier rank which is consistent with the sample. Also, for MS, ES, DULO, and strings, the notion of similarity based on EF games can be computed in polynomial time in the size of the sample. Using these results, we show that our algorithm runs in polynomial time in the size of the sample. Therefore, this result improves the one in (KAISER, 2012) for MS, ES, DULO, and strings.

Differently from (KAISER, 2012), our algorithm does not use Hintikka formulas since the size of a Hintikka formula is exponential in the quantifier rank. In our case, we define what we call the distinguishability sentences. They are defined based on conditions characterizing the winning strategies for the Spoiler. In this way, given two structures $\mathscr{A}, \mathscr{B}$ and a natural number r, we show that the distinguishability sentences hold in $\mathscr{A}$, do not hold in $\mathscr{B}$, and they have quantifier rank at most r. This result is essential for the definition of our algorithm and to guarantee its correctness. We define distinguishability sentences for each particular class of structures we are considering. We also define distinguishability sentences in a way such that they have polynomial size. This result is essential to ensure that our algorithm runs in polynomial time in the size of the sample. We also show that any first-order sentence is equivalent to a boolean combination of distinguishability sentences.

For an example of the synthesis problem, let T be the sample represented in Table 1.

The first-order sentence φ_T below states that the prefix of a string is *stv*, and it is consistent with the sample in Table 1. Variables range over positions in strings, $P_a(i)$ is true if the symbol a of the alphabet Σ occurs in position i, and S represents the successor relation over positions.

$$\varphi_T := \exists x_1 \exists x_2 \exists x_3 (P_s(x_1) \wedge P_t(x_2) \wedge P_v(x_3) \wedge S(x_1,x_2) \wedge S(x_2,x_3) \wedge \forall x_4 \neg S(x_4,x_1))$$

Table 1 – A sample of strings.

String	Class
stviil	Positive
ktvive	Negative
stviie	Positive
stpiie	Negative

Source: Own elaboration

The formula φ_T above illustrates one disadvantage of our first approach for the synthesis problem. For instance, for strings, first-order formulas over the vocabulary $\tau = \{S, (P_a)_{a\in\Sigma}\}$ are hard to read. Furthermore, the problem of checking whether a first-order formula holds in a structure is PSPACE-complete even when the structure is fixed (STOCKMEYER, 1974; VARDI, 1982; GRÄDEL *et al.*, 2005). Therefore, it is also PSPACE-complete for the classes of structures we are considering in this work. Then, we define a quantifier-free normal form (QNF) for MS, ES, DULO, and strings.

Formulas in quantifier-free normal form are easier to read than general first-order sentences. For example, pref(stv) is an atomic sentence in this quantifier-free normal form, and it also represents that the prefix of a string is *stv*. This normal form allows one to explicitly use predicates for substring properties and constants for strings. Then, one advantage of defining formulas in this normal form is that they are more succinct than first-order formulas over the standard vocabulary. This is the case because sentences in quantifier-free normal form are defined over a richer vocabulary such that atomic sentences are an abbreviation of first-order sentences over the standard vocabulary.

We also show that every first-order sentence can be converted into an equivalent formula in QNF and vice versa. We show this by using the results of EF games (MONTANARI *et al.*, 2005; KHOUSSAINOV; LIU, 2009) in the classes of structures we are considering in this work. We also use these results to show that one can check in polynomial time whether a formula in QNF holds in a structure. This is a second advantage of using formulas in quantifier-free normal form.

We also define a DNF version of formulas in quantifier-free normal form. Then, we introduce the problem of synthesis of formulas in quantifier-free disjunctive normal form (QDNF): given a sample of strings and the number of conjunctive clauses, the goal consists in finding a formula φ in QDNF with bounded number of clauses that holds in all positive structures and does not hold in any negative structure. In this problem, it is also given a set of atomic QNF formulas. Then, all atoms occurring in φ must be among the ones in the set. Then, we show how to consider a suitable set of atomic formulas in QNF. We also show that the synthesis problem for QDNF formulas is NP-complete.

We propose an approach to solve the synthesis problem for QDNF formulas by using a Boolean satisfiability (SAT) encoding. Then, we use a modern SAT solver to search for a solution. Therefore, we obtain an algorithm that returns a formula which is consistent with the sample. We call this method QDNFSAT.

The main disadvantage of the above method is that it may return a QNF sentence with a large number of literals per clause. Therefore, we also contemplate a problem in which the number of literals per clause is taken into consideration as well. Sentences with few clauses and with few literals per clause are more natural to be understood by humans than sentences with few clauses and hundreds of literals per clause. Then, if a QDNF sentence has at most l literals per clause, we say that it is a l-QDNF sentence. We define a codification of this variation of the synthesis problem for l-QDNF sentences into the SAT problem. Then, we call our method for this variation as l-QDNFSAT. We also show that this version concerning the number of literals per clause is NP-complete as well.

In real-world applications, it is unreasonable to expect that a sample is noise-free. For example, errors are introduced by measurement tools, such as sensors. Hence, it is a common task to deal with samples in which some elements are wrongly classified. In this sense, we also consider the synthesis problem when samples of structures have noise, i.e., some structures are wrongly classified. Therefore, this flexible version is a generalization over the synthesis from noiseless samples.

One should note that, for noisy samples, a structure may be classified as positive and negative. For example, for the sample in Table 2, the string *stpiie* is classified as positive and negative.

Regarding noisy samples, a sentence may not correctly classify a structure in the sample. Therefore, we use a cost associated with a sentence concerning a sample. The cost is the

Table 2 – A noisy sample of strings.

String	Class
stviil	Positive
ktvive	Negative
stviie	Positive
stpiie	Negative
stpiie	Positive

Source: Own elaboration

number of structures in the sample classified incorrectly by the sentence. In this noisy case, we also show how to consider an adequate set of atomic sentences in QDNF. In other words, we define a set of atomic sentences in QDNF such that there exists a QDNF sentence in which its atoms are among the ones in this set. Besides, its cost on the input sample is minimum.

We study two approaches to solve the problem with respect to noisy samples. In the first case, the goal is to find a QDNF formula that classifies the maximum number of structures correctly. We introduce an approach to solve this first case by a translation to the maximum satisfiability (MaxSAT) problem. We call this approach QDNFMaxSAT. In our second approach for handling noisy samples, a limit K to the number of examples not covered is also given as input. Then, a solution does not correctly classify at most K structures in the input sample. Therefore, our method is able to determine whether there exists such a sentence. Again, our approach for solving this second case is based on a reduction to the SAT problem. Since we consider the number of literals per clause in this case, we call this approach l-QDNFSAT-Noise.

1.2 Application in Grammatical Inference

For the class of strings, the problem we are considering is close to those in grammatical inference which investigates the task of finding a language model consistent with a given sample of strings (HIGUERA, 2010; HEINZ; SEMPERE, 2016; WIECZOREK, 2016). A language model can be deterministic finite automata (DFA) or context-free grammars, for example. Since strings may be used to model text data, traces of program executions, stress patterns in human languages, biological sequences, and sequences of symbolic data in general, grammatical inference can be applied in linguistics (STROTHER-GARCIA *et al.*, 2017; VU *et al.*, 2018; CHANDLEE *et al.*, 2019), robotic planning (RAWAL *et al.*, 2011; VU *et al.*, 2018), and classification of biological sequences (WIECZOREK; UNOLD, 2014; WIECZO-

REK; UNOLD, 2016; WIECZOREK; UNOLD, 2017; WIECZOREK *et al.*, 2019). For example, strings in Table 1 represent short segments of proteins. In our framework, first-order sentences can be seen as language models. Therefore, first-order sentences can also be used in the same applications as grammatical inference.

One of the most explored problems in grammatical inference is the DFA synthesis (ONCINA; GARCíA, 1992; HIGUERA, 2005; GARCÍA *et al.*, 2012). The goal of this problem is to find a DFA with the minimum number of states that is consistent with a given sample of classified strings. This problem is NP-hard (GOLD, 1978). However, several effective methods to solve this problem were developed (LUCAS; REYNOLDS, 2003; BUGALHO; OLIVEIRA, 2005; HEULE; VERWER, 2010; ULYANTSEV *et al.*, 2015; ZAKIRZYANOV *et al.*, 2019). The methods in (LUCAS; REYNOLDS, 2003; BUGALHO; OLIVEIRA, 2005) cannot guarantee that the DFA obtained is one with the minimum number of states. The method DFASAT introduced in (HEULE; VERWER, 2010) guarantees to find a minimum-size DFA. These approaches are based on a series of translations to SAT.

In (ULYANTSEV *et al.*, 2015), the authors also consider the problem of learning DFA from noisy data. This means that strings in the sample may be wrongly classified. Then, in this case, given a natural number K, the goal is to find a DFA with a given number of states such that it does not correctly classify at most K strings. Our second approach that returns QDNF sentences from noisy samples directly follows from the one in (ULYANTSEV *et al.*, 2015).

It is well known that a formal language is definable by a sentence of first-order logic over strings if and only if it is locally threshold testable (LTT) (THOMAS, 1982). The class of LTT languages is a subregular class of languages, i.e., it is included in the class of regular languages (ROGERS *et al.*, 2013). The problem of learning models of subregular languages is investigated in several works in grammatical inference (GARCIA; RUIZ, 2004; ROGERS *et al.*, 2013; HEINZ; ROGERS, 2013; AVCU *et al.*, 2017; STROTHER-GARCIA *et al.*, 2017).

A wide range of stress patterns in human languages may be defined in terms of subregular languages (HEINZ, 2009; ROGERS *et al.*, 2013; STROTHER-GARCIA *et al.*, 2017). In phonology, stress is a relative emphasis given to a certain syllable in a word. For example, the universal constraint that every word has exactly one syllable that receives primary stress can be defined as an LTT language (LAMBERT; ROGERS, 2019). Furthermore, the stress patterns of the human language Cambodian can also be defined in first-order logic over strings (LAMBERT; ROGERS, 2019). Table 3 represents a sample of stress patterns in the human

language Cambodian (LAMBERT; ROGERS, 2019). Symbols L and H denote light and heavy syllable weight, respectively. Acute accent denotes primary stress, and grave accent denotes secondary stress.

Table 3 – Sample of stress patterns in Cambodian.

String	Class
$\grave{H}\grave{L}\grave{H}\acute{L}$	Positive
$\grave{H}LH\acute{L}$	Negative
$\grave{H}L\grave{H}L\acute{H}$	Positive
$\grave{H}L\grave{L}\grave{H}\acute{H}$	Negative
$\grave{H}L\grave{H}L$	Negative

Source: Own elaboration

A grammatical inference algorithm that returns 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 (possibly noisy) sample of strings. For example, in order to find a first-order sentence which represents the stress patterns in a human language. As far as we know, this is the first work on finding a language model for LTT languages from positive and negative strings.

1.3 Related Work

In connection with this problem of data classification, grammatical inference is related to machine learning, an important branch of artificial intelligence. Machine learning includes methods such as deep learning, support vector machines (SVM), and Naive Bayes. These methods can also be used in the classification of strings (STANISLAWSKI *et al.*, 2013). However, grammatical inference provides a formal framework that allows us to state the complexity of patterns in terms of formal languages. For example, in phonological research, patterns in human languages may be defined in terms of regular languages (HEINZ, 2006; HEINZ, 2007; ROGERS *et al.*, 2013; STROTHER-GARCIA *et al.*, 2017). Therefore, DFASAT guarantees to find a model which recognizes a regular language while methods such as deep learning, SVM, and naive Bayes do not have this guarantee.

Furthermore, the main drawback of methods such as deep learning and SVM is that human-readable knowledge cannot be extracted directly from the models obtained by these approaches. For example, deep learning is based on artificial neural networks which are hard

to understand due to their black-box nature. From the point of view of interpretability, QDNF sentences are more appealing since they give a reason why a string belongs to a particular class. Interpretability is essential in fields where human experts need to be able to infer new knowledge from the model provided by the classification method.

A new logical framework to find a formula given a sample, also with a model-theoretic approach, can be found in (GROHE; RITZERT, 2017; GROHE *et al.*, 2017; GRIENENBERGER; RITZERT, 2019). In this framework, the input is only one structure, and its elements are classified as positive or negative. The problem is to find a hypothesis consistent with the classified elements where this hypothesis is a first-order formula in (GROHE; RITZERT, 2017) and a monadic second-order formula in (GROHE *et al.*, 2017; GRIENENBERGER; RITZERT, 2019). The input structures considered in (GRIENENBERGER; RITZERT, 2019) are trees. Moreover, only strings are considered as the input structure in (GROHE *et al.*, 2017). Observe that this problem is different from our problem here, as it deals with only one structure, whereas a sample consists of (possibly) many strings classified as positive or negative in our approach. Furthermore, the algorithm in (GROHE; RITZERT, 2017) assumes that the quantifier rank is fixed while we obtain a sentence of minimum quantifier rank.

A recent logical approach to grammatical inference defined in (STROTHER-GARCIA *et al.*, 2017) uses a propositional language. In this propositional language, atomic sentences are substrings which are taken to be true in a string if and only if they occur in that string as infix, prefix or suffix. Formulas in (STROTHER-GARCIA *et al.*, 2017) are defined by conjunctions of negative and positive literals (CNPL). In our approach, we also contemplate multiplicity and scattering of substrings, disjunctions, and length of strings. Furthermore, our approach uses a language as expressive as full first-order logic over strings. Formulas in CNPL can also be seen as first-order sentences. However, CNPL is less expressive than first-order logic over strings. Therefore, our approach is built on a language more expressive than the one in (STROTHER-GARCIA *et al.*, 2017).

Another logical framework for a similar problem is Inductive Logic Programming (ILP) (MUGGLETON, 1991; MUGGLETON; RAEDT, 1994; RAEDT, 2008). ILP uses logic programming as a uniform representation for the sample and hypotheses. Then, the fundamental difference to our framework is that, in ILP, the sample consists of formulas, while the sample consists of classified strings in our framework. Due to this difference between the framework of ILP and our approach, as far as we know, our work has no direct relationship with ILP. Therefore,

techniques used in ILP cannot easily be applied in our approach.

1.4 Outline

This work is organized as follows. In Chapter 2, we give the general notions of logics, finite model theory, Ehrenfeucht–Fraïssé games, formal languages, and automata. Then, in Chapter 3, we present EF games for a fixed class of structures. We consider monadic structures, equivalence structures, and strings. In Chapter 4, we show the related work on Boolean function synthesis, DFA synthesis, grammatical inference with finite model theory, and the algorithm in (KAISER, 2012).

In the rest of this work, we present our contributions. We give necessary and sufficient conditions characterizing the winning strategies for both players on disjoint union of linear orders in Chapter 5. In Chapter 6, for each class of structures we are considering, we introduce the distinguishability sentences and provide some useful properties. Also in Chapter 6, we propose our algorithm for the synthesis of minimum quantifier rank formulas.

In Chapter 7, we define the quantifier-free normal form for the classes of structures we are considering in this work. We also define the synthesis of QDNF formulas in Chapter 7. Furthermore, we also analyze the computational complexity of this problem and show how to solve it in this chapter. In Chapter 8, we consider the extension of the synthesis problem to handle noisy samples. Finally, we conclude and give some directions for future research in Chapter 9.

2 LOGICS, EF GAMES AND AUTOMATA

In this chapter, we start with a brief introduction to propositional (HUTH; RYAN, 2004; BIERE *et al.*, 2009) and first-order logics (EBBINGHAUS *et al.*, 1994). We also recall the fundamental notions of finite model theory and Ehrenfeucht–Fraïssé game for first-order logic. See (EBBINGHAUS; FLUM, 1995) for more details. In addition, we also give the basic definitions of formal languages and deterministic finite automata (HOPCROFT *et al.*, 2006). Lastly, we also consider the connection between formal languages and finite model theory. See (LIBKIN, 2004) for more details.

2.1 Propositional Logic

Propositional logic concentrates on propositions formed by connecting indivisible units by logical connectives. The propositional language is defined by using a set of propositional variables Ψ whose elements are usually denoted by p, q, r, and so on. The propositional variables are the indivisible units. The language of propositional logic consists of formulas defined as a Boolean combination of variables in Ψ as follows.

Definition 2.1.1 (Propositional Formulas)**.** *Let Ψ be a set of propositional variables. Propositional formulas are those obtained by using the following rules finitely many times:*

- *If $p \in \Psi$, then p is a formula.*
- *If ψ is a formula, then $(\neg\psi)$ is a formula.*
- *If ψ_1 and ψ_2 are formulas, then $(\psi_1 \circ \psi_2)$ is a formula, such that $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.*

We define $A(\psi)$ as the set of propositional variables in a formula ψ. The semantics of propositional logic is defined with respect to valuations, which assign a truth value to each of the propositional variables of the language. Formally, a valuation is a mapping $\mathcal{V} : \Psi \rightarrow \{T, F\}$. Let $\mathcal{V}$ be a valuation and ψ a propositional formula. If ψ is true in $\mathcal{V}$, we also say that $\mathcal{V}$ satisfies ψ, and we write $\mathcal{V} \models \psi$. If ψ is false in $\mathcal{V}$, we write $\mathcal{V} \not\models \psi$.

Definition 2.1.2. *Let $\mathcal{V}$ be a valuation. We define when $\mathcal{V}$ satisfies a propositional formula ψ inductively in the standard way as follows.*

- *$\mathcal{V} \models p$ iff $\mathcal{V}(p) = T$;*
- *$\mathcal{V} \models (\neg\psi)$ iff $\mathcal{V} \not\models \psi$;*
- *$\mathcal{V} \models (\psi_1 \wedge \psi_2)$ iff $\mathcal{V} \models \psi_1$ and $\mathcal{V} \models \psi_2$;*

- $\mathcal{V} \models (\psi_1 \vee \psi_2)$ *iff* $\mathcal{V} \models \psi_1$ *or* $\mathcal{V} \models \psi_2$.

A literal L is either a variable $p \in \Psi$ or its negation $\neg p$. A conjunctive clause C is a conjunction of literals $L_1 \wedge ... \wedge L_r$. A formula is in disjunctive normal form (DNF) if it is a disjunction of conjunctive clauses $C_1 \vee ... \vee C_s$. It is well known that any propositional formula can be expressed in DNF. For details, see (HUTH; RYAN, 2004).

In the SAT problem, the goal is to determine whether a given propositional formula is satisfiable, i.e., if there exists a valuation such that the propositional formula is true in this valuation. In this case, we say that the formula is satisfiable. It is well-known that SAT is NP-complete (COOK, 1971). However, highly optimized modern SAT solvers are able to solve many real-world instances with millions of variables efficiently (BIERE *et al.*, 2009). Current SAT methods assume that the propositional formula is in conjunctive normal form (CNF). A CNF propositional formula is a conjunction of clauses $\bigwedge_i C_i$ such that each clause C_i is a disjunction of literals $\bigvee_j L_j$. It is also well known that any propositional formula can be converted into an equivalent propositional formula that is in CNF. For details, see (HUTH; RYAN, 2004).

An optimization version of SAT is the MaxSAT problem which is NP-hard, and it consists in finding a valuation that maximizes the number of satisfied clauses. In this work, we consider the version of MaxSAT in which some clauses are declared to be soft clauses. Then, the input of MaxSAT is a set of soft clauses and a set of hard clauses $(\Psi^{soft}, \Psi^{hard})$, respectively. Soft clauses may be falsified, and the goal is finding a valuation that satisfies the maximum number of soft clauses. Moreover, the valuation must satisfy all the hard clauses. In real-life scenarios, solutions satisfying all the hard constraints and violating a minimum number of soft constraints are considered acceptable solutions. MaxSAT is an alternative to handle problems with hard and soft constraints in a natural and compact way (BIERE *et al.*, 2009).

Example 2.1.1. *Let* $\Psi^{hard} = \{(\neg p_1 \vee p_2), (\neg p_1 \vee \neg p_2)\}$ *and* $\Psi^{soft} = \{(p_1 \vee p_2), (p_1 \vee \neg p_2)\}$. *Clearly,* $(p_1 \vee p_2) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2)$ *is unsatisfiable. However, let v be a valuation such that* $v \not\models p_1$ *and* $v \models p_2$. *Then, v satisfies all hard clauses, and it maximizes the number of satisfied soft clauses.*

2.2 First-Order Logic

In order to define a first-order language, we first define the notion of alphabet.

Definition 2.2.1 (Alphabet)**.** *An alphabet of a first-order language consists of the following symbols:*

- *An enumerable set of variables VAR:* $x, y, z, x_1, y_1, z_1, x_2, ...;$
- *Logical symbols:* $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists;$
- *The equality symbol:* $=;$
- *Punctuation symbols:* $(,);$
- *An enumerable set* τ *of constant symbols, n-ary relation symbols and m-ary function symbols such that* $n, m \in \mathbb{N}$.

The only symbols that change over alphabets are the symbols in τ. The symbols in τ determine the first-order language. We call τ the vocabulary. Then, we represent a vocabulary as

$$\tau = \{R_1^{n_1}, R_2^{n_2}, ..., f_1^{m_1}, f_2^{m_2}, ..., c_1, c_2, ...\},$$

such that each n_i and m_i, for $i \in \mathbb{N}$, is the arity of R_i and f_i, respectively. When it is clear from context, we omit the arity of relation symbols and function symbols.

In the following, we define terms. Terms represent the elements in a domain. Then, terms are combinations of variables, constants and function symbols in a particular way.

Definition 2.2.2 (Terms)**.** *Let* τ *be a vocabulary.* τ*-terms are precisely the elements which can be obtained by finitely applications of the following rules:*

- *Every variable is a* τ*-term.*
- *Every constant symbol in* τ *is a* τ*-term.*
- *If* t_1, ..., t_k *are* τ*-terms and* $f \in \tau$ *is a k-ary function symbol, then* $f(t_1, ..., t_k)$ *is a* τ*-term.*

When the vocabulary τ is arbitrary or clear from context, we call τ-term just as terms. Now, we are able to define formulas using the definition of terms.

Definition 2.2.3 (Formulas)**.** *Let* τ *be a vocabulary.* τ*-formulas are obtained by finitely many applications of the following rules:*

- *If* t_1 *and* t_2 *are* τ*-terms, then* $t_1 = t_2$ *is a* τ*-formula.*
- *If* t_1, ..., t_k *are* τ*-terms and* $R^k \in \tau$ *is a relational symbol, then* $R(t_1, ..., t_k)$ *is a* τ*-formula.*
- *If* φ *is a* τ*-formula, then* $(\neg\varphi)$ *is a* τ*-formula.*
- *If* φ *and* ψ *are* τ*-formulas, then* $(\varphi \circ \psi)$ *is a* τ*-formula, such that* $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- *If* φ *is a* τ*-formula and* $x \in VAR$ *is a variable, then* $(\forall x\varphi)$ *and* $(\exists x\varphi)$ *are* τ*-formulas.*

As in the case of terms, we omit the vocabulary when it is arbitrary or clear from context. In this case, we simply use formulas instead of τ-formulas.

For the benefit of readability, we often omit parenthesis if they are clear from context. Furthermore, we use the natural infix notation for common function symbols or relation symbols, such as $\leq$. For instance, we write $x \leq y$ instead of $\leq (x,y)$.

The depth of nesting of the quantifiers in a formula is an essential concept in this work. This notion is related to rounds in the Ehrenfeucht–Fraïssé game. By the quantifier rank of a formula, we mean the depth of nesting of its quantifiers as in the following definition.

Definition 2.2.4 (Quantifier Rank)**.** *The quantifier rank of a first-order formula φ, written $qr(\varphi)$, is defined as*

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

A variable x can either occur free in a formula or bound to a quantifier. If the free variables in a formula φ are of interest, we write $\varphi(x_1 ..., x_n)$. A formula without free variables is called a sentence.

In order to determine whether a formula is true or false, we need to define a domain and a interpretation for the relation symbols, function symbols and constant symbols.

Definition 2.2.5 (Structures)**.** *Let τ be a vocabulary. A τ-structure $\mathscr{A}$ is a tuple*

$$\mathscr{A} = \langle A, R_1^{\mathscr{A}}, ..., f_1^{\mathscr{A}}, ..., c_1^{\mathscr{A}}, ... \rangle,$$

such that A is a nonempty set, the domain of $\mathscr{A}$, each $R_i^{\mathscr{A}}$ is an n_i-ary relation on A such that $R_i^{n_i}$ is a relation symbol in τ, each $f_i^{\mathscr{A}}$ is an m_i-ary function on A such that $f_i^{m_i}$ is a function symbol in τ and each $c_i^{\mathscr{A}}$ is an element of A such that $c_i \in \tau$.

Observe that a τ-structure gives an interpretation for constant symbols, function symbols and relation symbols. The interpretation of variables is given by an assignment.

Definition 2.2.6 (Assignment)**.** *Let $\mathscr{A}$ be a τ-structure. An assignment in $\mathscr{A}$ is a mapping of the set of variables into the domain A, i.e., a function $\beta : VAR \to A$.*

Now, we define the notion of interpretation in what follows.

Definition 2.2.7 (Interpretation)**.** *An τ-interpretation is a pair $(\mathscr{A},\beta)$ such that $\mathscr{A}$ is a τ-structure and β is an assignment in $\mathscr{A}$.*

We shall call simply structures and interpretations when the vocabulary τ is arbitrary or clear from context. The following definitions are important in order to define the truth-value of a formula under an interpretation.

Definition 2.2.8. *Let β be an assignment in a structure $\mathscr{A}$, $a \in A$ and x be a variable. Then, $\beta\frac{a}{x}$ is an assignment that maps x to a and it agrees with β on all variables distinct from x:*

$$\beta\frac{a}{x}(y) := \begin{cases} \beta(y), & \text{if } y \neq x \\ a, & \text{otherwise} \end{cases}$$

If $\mathscr{I} = (\mathscr{A},\beta)$, then $\mathscr{I}\frac{a}{x} := (\mathscr{A},\beta\frac{a}{x})$. Now, we define the interpretation of terms.

Definition 2.2.9 (Interpretation of Terms)**.** *Let τ be a vocabulary and $\mathscr{I} = (\mathscr{A},\beta)$ be an interpretation. The interpretation of terms is defined in what follows:*

- *If $x \in VAR$, then $\mathscr{I}(x) = \beta(x)$;*
- *If c is a constant symbol in τ, then $\mathscr{I}(c) = c^{\mathscr{A}}$;*
- *If f is an n-ary function symbol in τ and t_1, ..., t_n are terms, then $\mathscr{I}(f(t_1,...,t_n)) = f^{\mathscr{A}}(\mathscr{I}(t_1),...,\mathscr{I}(t_n))$.*

Now, we define the satisfaction relation that determines when a formula is true under a given interpretation. We say that an interpretation $\mathscr{I}$ satisfies a formula φ or φ holds in $\mathscr{I}$, and we write $\mathscr{I} \models \varphi$.

Definition 2.2.10 (Satisfaction Relation)**.** *Let $\mathscr{I} = (\mathscr{A},\beta)$ be an interpretation. We define the satisfaction relation inductively on formulas:*

- $\mathscr{I} \models t_1 = t_2$ *iff* $\mathscr{I}(t_1) = \mathscr{I}(t_2)$;
- $\mathscr{I} \models R(t_1,...,t_k)$ *iff* $\mathscr{I}(t_1),...,\mathscr{I}(t_k) \in R^{\mathscr{A}}$;
- $\mathscr{I} \models (\neg\varphi)$ *iff* $\mathscr{I} \not\models \varphi$;
- $\mathscr{I} \models (\varphi \wedge \psi)$ *iff* $\mathscr{I} \models \varphi$ *and* $\mathscr{I} \models \psi$;
- $\mathscr{I} \models (\varphi \vee \psi)$ *iff* $\mathscr{I} \models \varphi$ *or* $\mathscr{I} \models \psi$;
- $\mathscr{I} \models (\varphi \to \psi)$ *iff if* $\mathscr{I} \models \varphi$ *then* $\mathscr{I} \models \psi$);
- $\mathscr{I} \models (\varphi \leftrightarrow \psi)$ *iff (*$\mathscr{I} \models \varphi$ *if and only if* $\mathscr{I} \models \psi$;
- $\mathscr{I} \models (\exists x\varphi)$ *iff there is an* $a \in A$ *such that* $\mathscr{I}\frac{a}{x} \models \varphi$;
- $\mathscr{I} \models (\forall x\varphi)$ *iff for all* $a \in A$, $\mathscr{I}\frac{a}{x} \models \varphi$.

Observe that when a formula φ is a sentence, the assignment is unimportant. In this case, we write $\mathscr{A} \models \varphi$ without mentioning the assignment.

2.3 Ehrenfeucht–Fraïssé Games

Now, we focus on Ehrenfeucht–Fraïssé games (EHRENFEUCHT, 1961) for first-order logic and its importance in this work. For details, see (EBBINGHAUS; FLUM, 1995; GRÄDEL *et al.*, 2005; LIBKIN, 2004). The following definitions and results follow the description in (EBBINGHAUS; FLUM, 1995). First, we need the following definitions. For Ehrenfeucht–Fraïssé games, we consider only finite vocabularies without function symbols and constant symbols and finite structures, i.e., structures whose domain is finite.

Definition 2.3.1 (Isomorphism)**.** *Let τ be vocabulary. Let $\mathscr{A} = \langle A, R_1^{\mathscr{A}}, ..., R_m^{\mathscr{A}} \rangle$ and $\mathscr{B} = \langle B, R_1^{\mathscr{B}}, ..., R_m^{\mathscr{B}} \rangle$ be τ-structures. $\mathscr{A}$ and $\mathscr{B}$ are isomorphic, writen $\mathscr{A} \cong \mathscr{B}$, if there is an isomorphism, i.e., a bijection $h : A \to B$ such that for every n-ary $R_i \in \tau$ and $(a_1, ..., a_n) \in A^n$, $(a_1, ..., a_n) \in R_i^{\mathscr{A}}$ if and only if $(h(a_1), ..., h(a_n)) \in R_i^{\mathscr{B}}$.*

Definition 2.3.2 (Substructure)**.** *Let $\mathscr{A} = \langle A, R_1^{\mathscr{A}}, ..., R_m^{\mathscr{A}} \rangle$ and $\mathscr{B} = \langle B, R_1^{\mathscr{B}}, ..., R_m^{\mathscr{B}} \rangle$ be τ-structures. $\mathscr{A}$ is a substructure of $\mathscr{B}$ if*

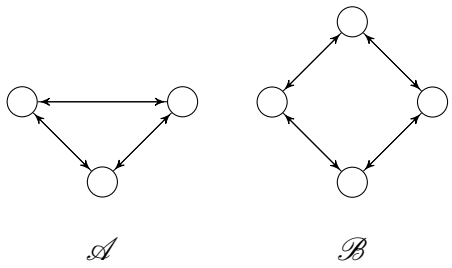
- *$A \subseteq B$.*
- *For every n-ary $R_i \in \tau$, $R_i^{\mathscr{A}} = R_i^{\mathscr{B}} \cap A^n$.*

Definition 2.3.3 (Induced Substructure)**.** *Let τ be a vocabulary. Let $\mathscr{A} = \langle A, R_1^{\mathscr{A}}, ..., R_m^{\mathscr{A}} \rangle$ be a τ-structure and $\{a_1, ..., a_k\} \subseteq A$. The substructure of $\mathscr{A}$ induced by $a_1, ..., a_k$ is the structure $\mathscr{A} \restriction \{a_1, ..., a_k\}$ such that its domain is $D = \{a_1, ..., a_k\}$ and for every n-ary $R_i \in \tau$, $R_i^{\mathscr{A} \restriction \{a_1, ..., a_k\}} = R_i^{\mathscr{A}} \cap D^n$.*

Definition 2.3.4 (Ehrenfeucht–Fraïssé Game)**.** *Let r be an integer such that $r \geq 0$, τ be a vocabulary, $\mathscr{A}$ and $\mathscr{B}$ be two τ-structures. The Ehrenfeucht–Fraïssé game (EF game, for short) $\mathscr{G}_r(\mathscr{A}, \mathscr{B})$ 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 $\mathscr{A}$, or from the domain B of $\mathscr{B}$. 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 ith round. The Duplicator wins the play if the mapping $(a_1, b_1), ..., (a_r, b_r)$ is an isomorphism between the substructures induced by $a_1, ..., a_r$ and $b_1, ..., b_r$, respectively. Otherwise, Spoiler wins this play.*

We say that a player has a winning strategy in $\mathcal{G}_r(\mathscr{A},\mathscr{B})$ if it is possible for him/her to win each play whatever choices are made by the opponent.

Figure 1 – EF game



Source: Own elaboration

Example 2.3.1. *Let $\mathscr{A}$ and $\mathscr{B}$ be the structures in Figure 1. The Spoiler has a winning strategy in $\mathcal{G}_2(\mathscr{A},\mathscr{B})$ by picking two elements in B with no edge between them.*

In this work, we always assume that $\mathscr{A}$ and $\mathscr{B}$ are not isomorphic. Then, note that if $r \geq min\{|A|,|B|\}$, then it follows directly that the Spoiler has a winning strategy. Therefore, we can assume that r is bounded by $min\{|A|,|B|\}$.

Now, for a structure $\mathscr{A}$ and a natural number r, we define formulas describing the properties of $\mathscr{A}$ in any EF game with r rounds.

Definition 2.3.5 (Hintikka Formulas)**.** *Let $\mathscr{A}$ be a structure, $\overline{a} = a_1...a_s \in A^s$, and $\overline{x} = x_1,...,x_s$ a tuple of variables,*

$$\varphi^0_{\mathscr{A},\overline{a}}(\overline{x}) := \bigwedge\{\varphi(\overline{x}) \mid \varphi \text{ is atomic or negated atomic and } w \models \varphi[\overline{a}]\},$$

and, for $r > 0$,

$$\varphi^r_{\mathscr{A},\overline{a}}(\overline{x}) := \bigwedge_{a\in A} \exists x_{s+1}\varphi^{r-1}_{\mathscr{A},\overline{a}a}(\overline{x},x_{s+1}) \wedge \forall x_{s+1}(\bigvee_{a\in A} \varphi^{r-1}_{\mathscr{A},\overline{a}a}(\overline{x},x_{s+1})).$$

We write $\varphi^r_{\mathscr{A}}$ whenever $s = 0$.

Example 2.3.2. *Let $\mathscr{A} = \langle A, R^{\mathscr{A}}, S^{\mathscr{A}}\rangle$ such that $A = \{1,2\}$, $R^{\mathscr{A}} = \{(1,1),(1,2)\}$ and $S^{\mathscr{A}} = \{2\}$. It follows that $\varphi^1_{\mathscr{A}} = \exists v_1\varphi^0_{\mathscr{A},1}(v1) \wedge \exists v_1\varphi^0_{\mathscr{A},2}(v1) \wedge \forall v_1(\varphi^0_{\mathscr{A},1}(v1) \vee \varphi^0_{\mathscr{A},2}(v1))$. Then,*

$$\varphi^1_{\mathscr{A}} = \begin{array}{l} \exists v_1(R(v_1,v_1) \wedge \neg S(v_1)) \wedge \exists v_1(\neg R(v_1,v_1) \wedge S(v_1)) \wedge \\ \forall v_1((R(v_1,v_1) \wedge \neg S(v_1)) \vee (\neg R(v_1,v_1) \wedge S(v_1))). \end{array}$$

Given a structure $\mathscr{A}$ and a natural number r, the size of $\varphi^r_{\mathscr{A}}$ is $O(2^r|A|^r)$. Therefore, since r is bounded by $|A|$, the size of $\varphi^r_{\mathscr{A}}$ is exponential in the size of $\mathscr{A}$. The following theorems are important to prove our main results. They are presented in (EBBINGHAUS; FLUM, 1995) (Theorem 2.2.8 and Theorem 2.2.11).

Theorem 2.3.1 (Ehrenfeucht's Theorem)**.** *(EHRENFEUCHT, 1961) Given structures $\mathscr{A}$ and $\mathscr{B}$, and $r \geq 0$, the following are equivalent:*

- *the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$.*
- *if φ is a sentence of quantifier rank at most r, then $\mathscr{A} \models \varphi$ iff $\mathscr{B} \models \varphi$.*
- *$\mathscr{B} \models \varphi^r_{\mathscr{A}}$.*

Example 2.3.3. *Let $\varphi^1_{\mathscr{A}}$ be the Hintikka formula from Example 2.3.2 and $\mathscr{B} = \langle B, R^{\mathscr{B}}, S^{\mathscr{B}}\rangle$ be a structure such that $B = \{1,2\}$, $R^{\mathscr{B}} = \{(1,1),(1,2),(2,2)\}$ and $S^{\mathscr{B}} = \{2\}$. Then, $\mathscr{B} \not\models \varphi^1_{\mathscr{A}}$ and the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$. Now, let $\mathscr{C} = \langle C, R^{\mathscr{C}}, S^{\mathscr{C}}\rangle$ be a structure such that $C = \{1,2,3\}$, $R^{\mathscr{C}} = \{(1,1),(1,2),(3,2),(3,3)\}$ and $S^{\mathscr{C}} = \{2\}$. Therefore, $\mathscr{C} \models \varphi^1_{\mathscr{A}}$ and the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{C})$.*

The following result ensures that every first-order sentence is equivalent to a disjunction of Hintikka formulas.

Theorem 2.3.2. *(EBBINGHAUS; FLUM, 1995) Let φ be a sentence of quantifier rank at most r. Then, there exist structures $\mathscr{A}_1$, ..., $\mathscr{A}_s$ such that*

$$\models \varphi \leftrightarrow (\varphi^r_{\mathscr{A}_1} \vee \ldots \vee \varphi^r_{\mathscr{A}_s}).$$

EF games provide information about the similarity between structures. If two structures $\mathscr{A}$ and $\mathscr{B}$ are not isomorphic, then there is an r such that the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$. This notion of similarity is introduced in (MONTANARI *et al.*, 2005).

Definition 2.3.6 ($MinRound(\mathscr{A},\mathscr{B})$)**.** *Let $\mathscr{A}$ and $\mathscr{B}$ be two structures. The minimum number of rounds r such that Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$ is denoted by $MinRound(\mathscr{A},\mathscr{B})$.*

Example 2.3.4. *Let $\mathscr{A} = \langle A, R^{\mathscr{A}}\rangle$ and $\mathscr{B} = \langle B, R^{\mathscr{B}}\rangle$ be structures such that $A = \{1,2,3\}$, $R^{\mathscr{A}} = \{1,2\}$, $B = \{1,2,3,4\}$, $R^{\mathscr{B}} = \{3,4\}$. Then, $MinRound(\mathscr{A},\mathscr{B}) = 2$ because the Spoiler chooses $1 \notin R^{\mathscr{B}}$ and then chooses $2 \notin R^{\mathscr{B}}$. Observe that the Spoiler does not have a winning strategy in less than two rounds.*

If $MinRound(\mathscr{A},\mathscr{B})$ is large, then the Spoiler needs more rounds in order to ensure a winning strategy. It follows that $\mathscr{A}$ and $\mathscr{B}$ are very similar.

EF games are important in our framework because if the Spoiler has a winning strategy in a game on $\mathscr{A}$ and $\mathscr{B}$ with r rounds, then there exists a first-order sentence φ of quantifier rank at most r that holds in $\mathscr{A}$ and does not hold in $\mathscr{B}$. Also, in this case, the sentence $\varphi^r_{\mathscr{A}}$ is an example of such a sentence. However, over arbitrary vocabularies, the problem of determining whether the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$ is *PSPACE*-complete (PEZZOLI, 1998). Fortunately, it is possible to do better for EF games on strings (MONTANARI *et al.*, 2005), monadic structures, linear orders, and equivalence structures (KHOUSSAINOV; LIU, 2009).

2.4 Formal Languages and Finite Automata

We begin this section by giving the basic notations of strings and formal languages. Then, we give the notion of deterministic finite automata.

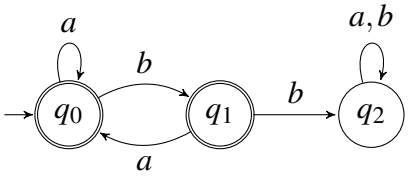
In formal language theory, an alphabet Σ is a nonempty, finite set which its elements are called symbols. A string $w = a_1a_2...a_n$ over an alphabet Σ is a finite sequence of symbols $a_i \in \Sigma$, for $i \in \{1,2,...,n\}$. The set of all such finite strings is denoted by Σ^*. A formal language is a subset of $L \subseteq \Sigma^*$. The empty sequence is called the empty string and it is denoted by ε. The length of w, denoted by $|w|$, is the number of symbols in w. The set $\{1,...,n\}$ is the set of positions of w. We assume some familiarity with formal languages. See (HOPCROFT *et al.*, 2006) for details.

For two strings $u = a_1...a_n$ and $v = b_1...b_m$, the concatenation of u and v is the string $uv = a_1...a_nb_1...b_m$. 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. $suff_k(w)$).

A deterministic finite automaton (DFA) is a finite-state machine which accepts or rejects strings. The following Figure 2 depicts the state diagram of a DFA A_1.

DFA A_1 has three states, labeled q_0, q_1 and q_2. The start state, q_0, is indicated by an arrow pointing at it from nowhere. The final states, q_0 and q_1, are the ones with a double circle. The arrows going from one state to another are called transitions. A_1 is deterministic because for a given state and symbol, its transition goes to exactly one state. Now, we formally define deterministic finite automata.

Figure 2 – DFA A_1



Source: Own elaboration

Definition 2.4.1. *(HOPCROFT et al., 2006) A deterministic finite automaton is a 5-tuple* $A = (Q, \Sigma, \delta, q_0, F)$ *such that*

- Q *is a finite set which its elements are called states,*
- Σ *is an alphabet,*
- $\delta : Q \times \Sigma \to Q$ *is the transition function,*
- $q_0 \in Q$ *is the start state, and*
- $F \subseteq Q$ *is the set of final states.*

Example 2.4.1. *We describe* A_1 *formally by* $A_1 = (Q, \Sigma, \delta, q_0, F)$ *such that*

- $Q = \{q_0, q_1, q_2\}$,
- $\Sigma = \{a, b\}$,
- δ *is described as*

	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_2

- q_0 *is the start state, and*
- $F = \{q_0, q_1\}$.

A computation of a DFA A on some string $w = a_1...a_n \in \Sigma^*$ is a sequence $q_1, ..., q_n$ such that $q_i \in Q$ for $i \in \{1, ..., n\}$, q_1 is the start state, $\delta(q_i, a_i) = q_{i+1}$ for $i \in \{1, ..., n-1\}$. We write $q_1 \to^w q_n$ to represent the computation. For example, the computation of A_1 on *aabab* is the sequence $q_0, q_0, q_0, q_1, q_0, q_1$. Then, we write $q_0 \to^{aabab} q_1$.

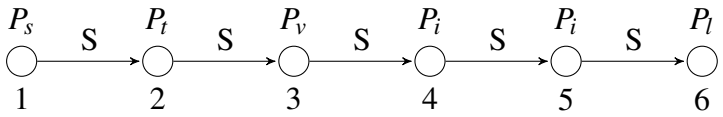
A string w is accepted by A if $q_1 \to q_n$ is the computation of A on w and $q_n \in F$. In this case, we say that A accepts w. If a DFA A does not accept a string w, we say that A rejects w. For example, A_1 accepts *aabab*, and it rejects *aababba*.

The language recognized by A is the set $L(A) = \{w \in \Sigma^* \mid q_1 \rightarrow^w q, q \in F\}$. We also say that A recognizes a language L' if $L(A) = L'$. For example, A_1 recognizes the set of all strings over $\{a,b\}^*$ that do not have consecutive b's. A language L' is called regular if there exists a DFA A such that $L(A) = L'$. Then, the language $L = \{w \in \{a,b\}^* \mid w \text{ has no consecutive } b's\}$ is regular.

2.5 Formal Languages and Finite Model Theory

In finite model theory, we view a string $w = a_1...a_n$ over Σ as a logical structure $\mathscr{A}_w$ over the vocabulary $\tau = \{S, (P_a)_{a\in\Sigma}\}$ with domain $A = \{1,...,n\}$, 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. Figure 3 represents the string *stvill* as a logical structure.

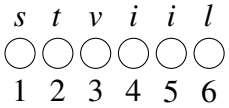
Figure 3 – String *stviil* as a logical structure $\mathscr{A}_{stviil}$



Source: Own elaboration

In order to make the presentation simpler, we use the following notation. Then, we remove the successor relation and we use the actual symbol in Σ as the unary predicate.

Figure 4 – Simpler presentation for $\mathscr{A}_{stviil}$



Source: Own elaboration

Given a first-order sentence φ over strings, the formal language defined by φ is simply $L(\varphi) := \{w \in \Sigma^* \mid \mathscr{A}_w \models \varphi\}$. As an example, if $\varphi = \exists x P_a(x)$, then $L(\varphi) = \Sigma^* a \Sigma^*$. Then, we can understand some classes of languages using concepts from finite model theory and first-order logic. In general, we do not distinguish between such structures and strings. Therefore, we use the notation $w \models \varphi$ instead of $\mathscr{A}_w \models \varphi$.

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 $\{w \mid w$ has u as infix at least d times $\}$, for some $u \in \Sigma^*$ and $d \in \mathbb{N}$ (ZEITOUN *et al.*, 2014). 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.

LTT languages can be defined in terms of first-order logic. A language is definable by a sentence of first-order logic (FO) over strings if and only if it is LTT (THOMAS, 1982). Fragments of first-order logic define other classes of languages such as locally testable (LT) and strictly local (SL).

Let $w = a_1 ... a_n$ be a string. We define first-order sentences

$$\phi_w := \exists x_1, ... \exists x_n (\bigwedge_{1 \leq i \leq n-1} S(x_i, x_{i+1}) \wedge \bigwedge_{1 \leq i \leq n} P_{a_i}(x_i)).$$

$$\phi_w^{pref} := \exists x_1, ... \exists x_n (\bigwedge_{1 \leq i \leq n-1} S(x_i, x_{i+1}) \wedge \bigwedge_{1 \leq i \leq n} P_{a_i}(x_i) \wedge \forall z \neg S(z, x_1)).$$

$$\phi_w^{suff} := \exists x_1, ... \exists x_n (\bigwedge_{1 \leq i \leq n-1} S(x_i, x_{i+1}) \wedge \bigwedge_{1 \leq i \leq n} P_{a_i}(x_i) \wedge \forall z \neg S(x_n, z)).$$

Lemma 2.5.1. *Let w and v be strings. Then,*

- *v is a substring of w iff $w \models \phi_v$;*
- *v is a prefix of w iff $w \models \phi_v^{pref}$;*
- *v is a suffix of w iff $w \models \phi_v^{suff}$.*

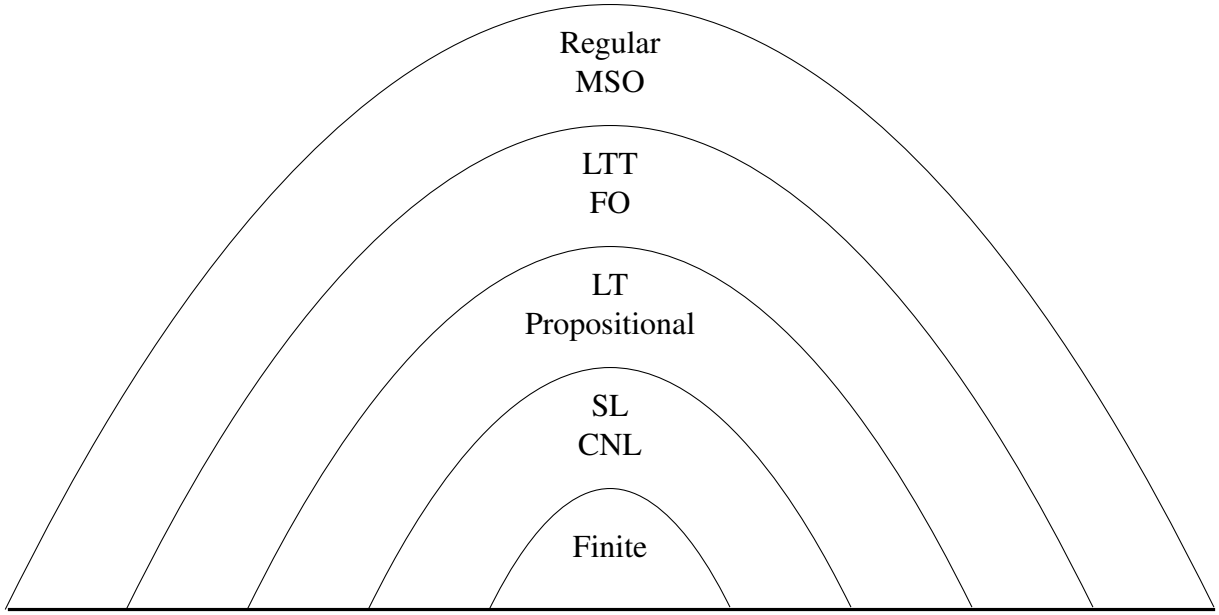
For example, for $\phi_{ab} = \exists x_1 \exists x_2 S(x_1, x_2) \wedge P_a(x_1) \wedge P_b(x_2)$, it follows that $abb \models \phi_{ab}$.

Now, we give some results on formal languages and finite model theory for LT and SL. A first-order sentence ϕ is called propositional if it is a Boolean combination of formulas of the forms ϕ_w, ϕ_w^{pref} and ϕ_w^{suff}. Then, the class of LT languages is exactly the set of languages defined by propositional first-order sentences (ROZENBERG; SALOMAA, 1997).

In order to obtain the class of SL languages, we need to restrict the set of propositional first-order sentences further. We call literal formulas of the forms ϕ_w, ϕ_w^{pref}, and ϕ_w^{suff} or their negations $\neg\phi_w$, $\neg\phi_w^{pref}$, and $\neg\phi_w^{suff}$. A conjunction of negative literals (CNL) is a propositional first-order sentence of the form $\phi = \bigwedge_{i=1}^{n} \phi_i$ such that each ϕ_i is a negative literal. A language is SL if and only if it is defined by a CNL sentence (ROGERS *et al.*, 2013).

We finish this section with Figure 5 that shows a hierarchy of classes of languages defined by logics. The class of regular languages is exactly the class of languages definable in monadic second-order logic (MSO) (BÜCHI, 1960). Monadic second-order logic is an extension of first-order logic in which quantification over subsets of the domain is allowed.

Figure 5 – A Hierarchy of classes of languages defined by logics over strings with the successor relation.



Source: Own elaboration

In the next chapter, we review the results on EF games for monadic structures, equivalence structures, and strings.

3 EF GAMES FOR A FIXED CLASS OF STRUCTURES

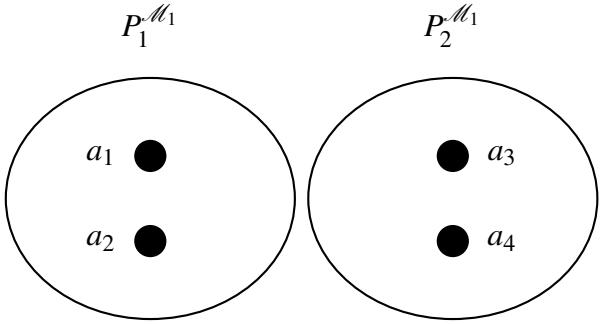
In Chapter 2, we showed the basic definitions and results on EF games over an arbitrary vocabulary. However, over arbitrary vocabularies, the problem of determining whether the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$ is *PSPACE*-complete (PEZZOLI, 1998). In this chapter, we show results on EF games over a particular class of structures. We focus on monadic structures (MS), equivalence structures (ES), and strings. We consider these classes because one can check in polynomial time in the size of the given structures $\mathscr{A}$ and $\mathscr{B}$ whether the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$. All three cases can be found in the literature. For details of this result on MS and ES, see (KHOUSSAINOV; LIU, 2009). For the case of strings, see (MONTANARI *et al.*, 2005).

3.1 Monadic Structures

A monadic structure is a structure $\mathscr{M} = \langle M, P_1^{\mathscr{M}}, ..., P_k^{\mathscr{M}} \rangle$ such that each P_i is monadic and $P_1^{\mathscr{M}}, ..., P_k^{\mathscr{M}}$ are pairwise disjoint. We set $P_{k+1} = \neg(P_1 \cup ... \cup P_k)$.

Example 3.1.1. *Let $M_1 = \{a_1, a_2, a_3, a_4\}$ and $P_1^{\mathscr{M}_1} = \{a_1, a_2\}$. $\mathscr{M}_1 = \langle M_1, P_1^{\mathscr{M}_1} \rangle$ is a monadic structure. Then, $P_2^{\mathscr{M}_1} = \{a_3, a_3\}$. Figure 6 represents $\mathscr{M}_1$.*

Figure 6 – Monadic structure $\mathscr{M}_1$



Source: Own elaboration

Theorem 3.1.1 (EF Games on MS)**.** *(KHOUSSAINOV; LIU, 2009) Let $\mathscr{M}_1$ and $\mathscr{M}_2$ be monadic structures. The Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{M}_1,\mathscr{M}_2)$ iff there exists $i \in \{1,...,k+1\}$ such that ($|P_i^{\mathscr{M}_1}| < r$ or $|P_i^{\mathscr{M}_2}| < r$) and $|P_i^{\mathscr{M}_1}| \neq |P_i^{\mathscr{M}_2}|$.*

Proof. ($\Rightarrow$) Suppose that for all $i \in \{1,...,k+1\}$, $|P_i^{\mathscr{M}_1}| \geq r$ and $|P_i^{\mathscr{M}_2}| \geq r$ or $|P_i^{\mathscr{M}_1}| = |P_i^{\mathscr{M}_2}|$. If the Spoiler chooses an element $a \in P_i^{\mathscr{M}_1}$, then the Duplicator chooses $b \in P_i^{\mathscr{M}_2}$, for $i \in \{1,...,k+1\}$. Analogously, if the Spoiler chooses an element $b \in P_i^{\mathscr{M}_2}$. This strategy is clearly winning for the Duplicator.
($\Leftarrow$) Suppose that there exists $i \in \{1,...,s+1\}$ such that $|P_i^{\mathscr{M}_1}| < r$ or $|P_i^{\mathscr{M}_2}| < r$ and $|P_i^{\mathscr{M}_1}| \neq |P_i^{\mathscr{M}_2}|$. Assume that $|P_i^{\mathscr{M}_1}| < |P_i^{\mathscr{M}_2}|$. The winning strategy for the Spoiler is to choose all elements from $P_i^{\mathscr{M}_2}$. □

Given the result in Theorem 3.1.1, now we show how to compute $MinRound(\mathscr{M}_1, \mathscr{M}_2)$. The Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{M}_1, \mathscr{M}_2)$ if and only if there exists $i \in \{1,...,k+1\}$ such that ($|P_i^{\mathscr{M}_1}| < r$ or $|P_i^{\mathscr{M}_2}| < r$) and $|P_i^{\mathscr{M}_1}| \neq |P_i^{\mathscr{M}_2}|$. Recall we always assume that the structures are not isomorphic. Then, the minimum number of rounds such that the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{M}_1, \mathscr{M}_2)$ is

$$MinRound(\mathscr{M}_1, \mathscr{M}_2) = min\{min(|P_i^{\mathscr{M}_1}|, |P_i^{\mathscr{M}_2}|) \mid |P_i^{\mathscr{M}_1}| \neq |P_i^{\mathscr{M}_2}|, 1 \leq i \leq k+1\} + 1.$$

Since $\mathscr{M}_1$ and $\mathscr{M}_2$ are not isomorphic, such a number $MinRound(\mathscr{M}_1, \mathscr{M}_2)$ always exists. Moreover, it is clear that one can compute $MinRound(\mathscr{M}_1, \mathscr{M}_2)$ in polynomial time in the size of $\mathscr{M}_1$ and $\mathscr{M}_2$. Specifically, in time $O(|M_1| + |M_2|)$.

Example 3.1.2. *Let $\mathscr{M}_1$ and $\mathscr{M}_2$ be the monadic structures in Figure 6 and Figure 7, respectively. Clearly, the Duplicator has a winning strategy in $\mathscr{G}_2(\mathscr{M}_1, \mathscr{M}_2)$. By the proof of Theorem 3.1.1, the Spoiler has a winning strategy in $\mathscr{G}_3(\mathscr{M}_1, \mathscr{M}_2)$. Therefore, $MinRound(\mathscr{M}_1, \mathscr{M}_2) = 3$.*

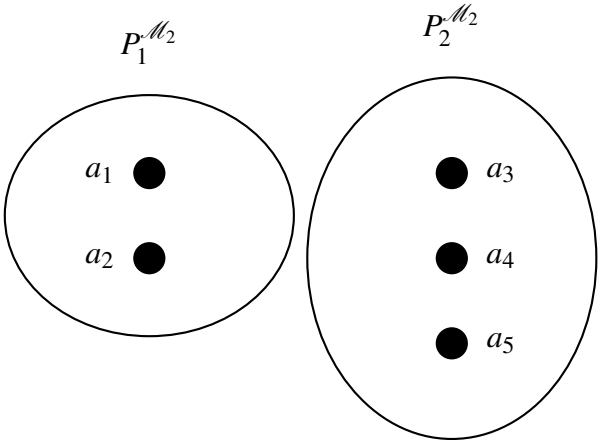
3.2 Equivalence Structures

An equivalence structure is a structure of the form $\mathscr{E} = \langle A, E^{\mathscr{E}} \rangle$ such that $E^{\mathscr{E}}$ is an equivalence relation on A. Let $q_t^{\mathscr{E}}$ be the number of equivalence classes in $\mathscr{E}$ of size t. Let $q_{\geq t}^{\mathscr{E}}$ be the number of equivalence classes in $\mathscr{E}$ of size at least t.

Example 3.2.1. *Let $\mathscr{E}_1$ be the equivalence structure in Figure 8. Then, $q_1^{\mathscr{E}_1} = 1$, $q_2^{\mathscr{E}_1} = 3$, and $q_3^{\mathscr{E}_1} = 0$. Furthermore, $q_{\geq 1}^{\mathscr{E}_1} = 4$, $q_{\geq 2}^{\mathscr{E}_1} = 3$, $q_{\geq 3}^{\mathscr{E}_1} = 0$.*

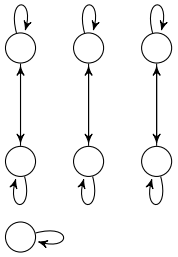
Definition 3.2.1 (Small Disparity)**.** *Let r be a natural number, $\mathscr{E}_1$ and $\mathscr{E}_2$ be equivalence structures. We say that $\mathscr{G}_r(\mathscr{E}_1, \mathscr{E}_2)$ has a small disparity if there exists a $t < r$ such that $q_t^{\mathscr{E}_1} \neq q_t^{\mathscr{E}_2}$ and $r \geq min\{q_t^{\mathscr{E}_1}, q_t^{\mathscr{E}_2}\} + t + 1$*

Figure 7 – Monadic structure $\mathscr{M}_2$



Source: Own elaboration

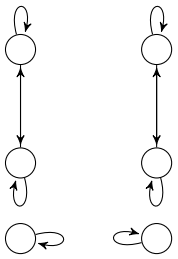
Figure 8 – Equivalence structure $\mathscr{E}_1$



Source: Own elaboration

Example 3.2.2. *Let $\mathscr{E}_1$ and $\mathscr{E}_2$ be the equivalence structure in Figure 8 and Figure 9, respectively. Then, $\mathscr{G}_3(\mathscr{E}_1,\mathscr{E}_2)$ has a small disparity because there exists $t = 1$ such that $t < 3$, $q_t^{\mathscr{E}_1} \neq q_t^{\mathscr{E}_2}$, and $3 \geq min\{2,1\} + t + 1$.*

Figure 9 – Equivalence structure $\mathscr{E}_2$



Source: Own elaboration

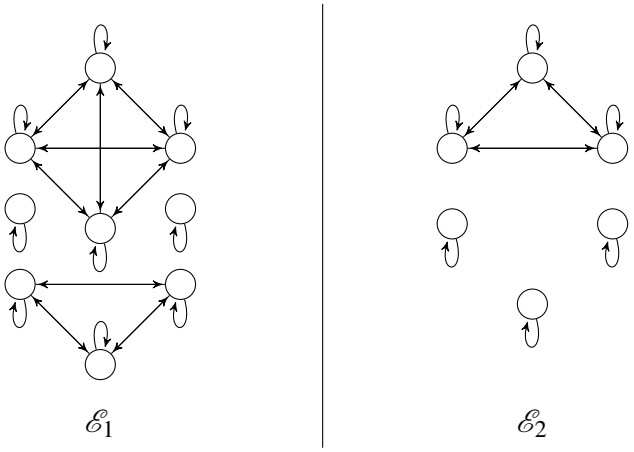
Observe that the Spoiler has a winning strategy in $\mathscr{G}_3(\mathscr{E}_1,\mathscr{E}_2)$. First, the Spoiler selects elements

b_1, b_2 from distinct equivalence classes of size 1 in $\mathscr{E}_2$. In the first round, the Duplicator must choose an element a_1 also in an equivalence class of size 1. Clearly, in the second round, the Duplicator must choose an element a_2 in an equivalence class of size different from 1. Then, the Spoiler wins by choosing an element a_3 such that there is an edge between a_3 and a_2.

Definition 3.2.2 (Large Disparity)**.** *Let r be a natural number, $\mathscr{E}_1$ and $\mathscr{E}_2$ be equivalence structures. We say that $\mathscr{G}_r(\mathscr{E}_1, \mathscr{E}_2)$ has a large disparity if there exists a $t \leq r$ such that $q^{\mathscr{E}_1}_{\geq t} \neq q^{\mathscr{E}_2}_{\geq t}$ and $r \geq min\{q^{\mathscr{E}_1}_{\geq t}, q^{\mathscr{E}_2}_{\geq t}\} + t$.*

Example 3.2.3. *Let $\mathscr{E}_1$ and $\mathscr{E}_2$ be the equivalence structures in Figure 10. Then, $\mathscr{G}_3(\mathscr{E}_1, \mathscr{E}_2)$ has a large disparity as there exists $t = 2$ such that $q^{\mathscr{E}_1}_{\geq 2} = 2$, $q^{\mathscr{E}_2}_{\geq 2} = 1$, $q^{\mathscr{E}_1}_{\geq 2} \neq q^{\mathscr{E}_2}_{\geq 2}$, and $r \geq min\{2,1\} + 2$. It is important to note that the Spoiler has a winning strategy in $\mathscr{G}_3(\mathscr{E}_1, \mathscr{E}_2)$. He/She chooses elements a_1, a_2 from distinct equivalence classes of size at least 2 in $\mathscr{E}_1$. Clearly, the Duplicator must choose an element b_1 from an equivalence class of size at least 2 and an element b_2 from an equivalence class of size 1 in $\mathscr{E}_2$, respectively. The Spoiler selects an element a_3 in $\mathscr{E}_1$ from the equivalence class of a_2. Thus, the Spoiler wins the game.*

Figure 10 – Large Disparity



Source: Own elaboration

Lemma 3.2.1. *(KHOUSSAINOV; LIU, 2009) Let r be a natural number, $\mathscr{E}_1$ and $\mathscr{E}_2$ be two equivalence structures. If $\mathscr{G}_r(\mathscr{E}_1, \mathscr{E}_2)$ has a small or large disparity, then the Spoiler has a winning strategy.*

Proof. First, assume that $q^{\mathscr{E}_1}_t > q^{\mathscr{E}_2}_t$ and $r \geq q^{\mathscr{E}_2}_t + t + 1$. The Spoiler's winning strategy is to select distinct elements a_1, ..., $a_{q^{\mathscr{E}_2}_t}$ from distinct equivalence classes of size t in $\mathscr{E}_1$. The

Duplicator must select b_1, ..., $b_{q_t^{\mathscr{E}_2}}$ from distinct equivalence classes of size t in $\mathscr{E}_2$. Otherwise, the Duplicator loses the game. Then, the Spoiler chooses distinct elements $a_{q_t^{\mathscr{E}_2}+1}$, ..., $a_{q_t^{\mathscr{E}_2}+t}$ from an equivalence class of size t not chosen before in $\mathscr{E}_1$. Therefore, the Duplicator must choose $b_{q_t^{\mathscr{E}_2}+1}$, ..., $b_{q_t^{\mathscr{E}_2}+t}$ from an equivalence class B of size greater than t. Then, the Spoiler wins the game by choosing $b_{q_t^{\mathscr{E}_2}+t+1}$ in B.

For the second part, assume that $q_{\geq t}^{\mathscr{E}_1} > q_{\geq t}^{\mathscr{E}_2}$ and $r \geq q_{\geq t}^{\mathscr{E}_2} + t$. The Spoiler's winning strategy is the following. The Spoiler chooses distinct elements a_1, ..., $a_{q_{\geq t}^{\mathscr{E}_2}}$ from distinct equivalence classes of size greater or equal to t in $\mathscr{E}_1$. If the Duplicator chooses a b_i from an equivalence class of size smaller than t in $\mathscr{E}_2$, then the Spoiler wins by selecting t elements in the equivalence class of a_i. Therefore, the Duplicator chooses b_1, ..., $b_{q_{\geq t}^{\mathscr{E}_2}}$ from distinct equivalence classes of size greater or equal to t in $\mathscr{E}_2$. Then, the Spoiler chooses $t-1$ distinct elements $a_{q_{\geq t}^{\mathscr{E}_2}+1}$, ..., $a_{q_{\geq t}^{\mathscr{E}_2}+t-1}$ from an equivalence class B of size greater or equal to t not chosen before in $\mathscr{E}_1$. Clearly, the Duplicator must choose $b_{q_{\geq t}^{\mathscr{E}_2}+1}$, ..., $b_{q_{\geq t}^{\mathscr{E}_2}+t-1}$ from an equivalence class of size smaller than t. Therefore, the Spoiler wins by selecting $a_{q_t^{\mathscr{E}_2}+t}$ from B. □

Theorem 3.2.1 (EF Games on ES)**.** *(KHOUSSAINOV; LIU, 2009) Let r be a natural number, and $\mathscr{E}_1, \mathscr{E}_2$ be equivalence structures. The Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{E}_1, \mathscr{E}_2)$ iff $\mathscr{G}_r(\mathscr{E}_1, \mathscr{E}_2)$ has a small or large disparity.*

By Theorem 3.2.1, the minimum number of rounds such that the Spoiler has a winning strategy can be computed in the following way. As we assume that $\mathscr{E}_1$ and $\mathscr{E}_2$ are not isomorphic, such a minimum number exists.

$$MinRound(\mathscr{E}_1, \mathscr{E}_2) := \begin{array}{l} min(min\{min(q_t^{\mathscr{E}_1}, q_t^{\mathscr{E}_2}) + t \mid q_t^{\mathscr{E}_1} \neq q_t^{\mathscr{E}_2}\} + 1, \\ min\{min(q_{\geq t}^{\mathscr{E}_1}, q_{\geq t}^{\mathscr{E}_2}) + t \mid q_{\geq t}^{\mathscr{E}_1} \neq q_{\geq t}^{\mathscr{E}_2}\}). \end{array}$$

Note that computing all equivalence classes of $\mathscr{E}_1$ and $\mathscr{E}_2$ takes time $O((|E_1| + |E_2|)^2)$. Then, it takes quadratic time to compute $q_t^{\mathscr{E}_1}$, $q_t^{\mathscr{E}_2}$, $q_{\geq t}^{\mathscr{E}_1}$, $q_{\geq t}^{\mathscr{E}_2}$, for $t \in \{1, ..., r\}$. Clearly, it is takes linear time to compute $MinRound(\mathscr{E}_1, \mathscr{E}_2)$ from $q_t^{\mathscr{E}_1}$, $q_t^{\mathscr{E}_2}$, $q_{\geq t}^{\mathscr{E}_1}$, $q_{\geq t}^{\mathscr{E}_2}$, for $t \in \{1, ..., r\}$. Therefore, the overall procedure to determine $MinRound(\mathscr{E}_1, \mathscr{E}_2)$ takes time $O((|E_1| + |E_2|)^2)$.

3.3 Strings

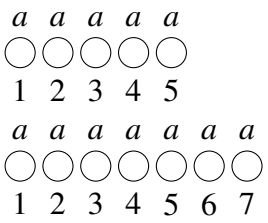
We consider strings over a given alphabet Σ. We denote the prefix (resp. suffix) of length k of a string w by $pref_k(w)$ (resp. $suff_k(w)$). Let i and j be positions in a string. The distance between i and j, denoted by $d(i, j)$, is $|i - j|$.

First, we consider EF games $\mathcal{G}_r(u,v)$ such that only one symbol occurs in u and v, i.e., $u = a^m$, $v = a^n$, and $a \in \Sigma$. Then, it is not necessary to take into account the unary predicates. Therefore, the following result depends only on the successor relation. Furthermore, even when u and v are arbitrary strings such that $|u| \neq |v|$, the Spoiler has a winning strategy based on the lengths of u and v for a small number of rounds relative to the length of the strings.

Lemma 3.3.1. *(MONTANARI et al., 2005) Let r be a positive integer, u and v be strings. If* $|u| \neq |v|$ *and* $|u| < 2^r - 2$ *or* $|v| < 2^r - 2$*), then the Spoiler has a winning strategy in* $\mathcal{G}_r(u,v)$.

Example 3.3.1. *Let* $r = 3$*,* $u = aaaaa$*, and* $v = aaaaaaaaa$ *be the strings in Figure 11. Observe that* $|u| < 2^r - 2$*. The Spoiler has the following winning strategy. First, the Spoiler chooses* 5 *in v. The best option for the Duplicator is to choose* 3 *in u. Then, the Spoiler selects* 7 *in v. The Duplicator must choose* 5 *in u. Now, the Spoiler wins by choosing* 8 *in v.*

Figure 11 – EF Game $\mathcal{G}_3(aaaaa, aaaaaaaaa)$



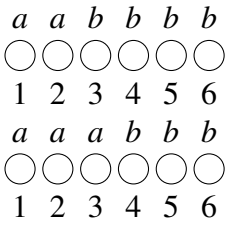
Source: Own elaboration

Now, we deal with the unary predicates. In order for the Duplicator to be able to reply to a move made by the Spoiler, the Duplicator must choose an element in a region such that the substrings in corresponding regions must be the same. First, we consider prefixes and suffixes.

Lemma 3.3.2. *Let r be a positive integer, u and v be strings. If* $pref_{2^r-2}(u) \neq pref_{2^r-2}(v)$ *or* $suff_{2^r-2}(u) \neq suff_{2^r-2}(v)$*, then the Spoiler has a winning strategy in* $\mathcal{G}_r(u,v)$.

Example 3.3.2. *Let* $u = aabbbb$ *and* $v = aaabbb$ *be the strings in Figure 12. Note that* $|u| = |v|$*. Then, Lemma 3.3.1 can not be used. Let* $r = 3$*. Therefore,* $pref_{2^r-2}(u) \neq pref_{2^r-2}(v)$*. Moreover,* $pref_3(u) \neq pref_3(v)$*. Now, we describe the Spoiler's winning strategy. First, the Spoiler chooses position* 3 *from u. The Duplicator must choose position* 4 *from v. Then, the Spoiler chooses position* 2 *from v. The Duplicator must select position* 1 *from u. Now, the Spoiler wins by choosing position* 1 *from v.*

Figure 12 – EF Game $\mathscr{G}_3(aabbbb, aaabbb)$



Source: Own elaboration

Now, we consider infixes. Intuitively, in order to guarantee a winning strategy for the Duplicator, not only must u and v have the same substrings of suitable length, but there must be enough substrings distributed in a similar way in both u and v.

First, we need the following definitions. Let α, w be strings. The set of starting positions of the occurrences of α in w is $\Gamma(\alpha, w) = \{i \mid w_i ... w_{i+|\alpha|-1} = \alpha\}$. The multiplicity of α in w, denoted by $\gamma(\alpha, w)$, is the number of occurrences of α in w, i.e., $|\Gamma(\alpha, w)|$.

Example 3.3.3. *Let $\alpha = aba$ and $w = aaababababbabaababaaa$ be strings. The set of starting positions of the occurrences of α in w is*

$$\Gamma(\alpha, w) = \{3, 5, 7, 12, 15, 17\}.$$

The multiplicity of α in w is

$$\gamma(\alpha, w) = |\Gamma(\alpha, w)| = 6.$$

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 partition, $d(i, j) \leq l$ and if i, j are in the same partition X and $i \leq h \leq j$, then $h \in X$. Each partition X in the segmentation is called a segment. We use segmentations to define the scattering of a substring in a string. The scattering of α in w, denoted by $\sigma(\alpha, w)$, is the number of segments in a $(|\alpha| + 1)$-segmentation of $\Gamma(\alpha, w)$. Therefore, the scattering represents a distribution of occurrences of α in w. In the following, we give an example of scattering.

Example 3.3.4. *Let $\alpha = aba$ and $w = aaababababbabaababaaa$ be the strings from Example 3.3.3. A 4-segmentation of $\Gamma(\alpha, w)$ consists of the following partitions:*

$$\{3, 5, 7\}, \{12, 15\} \text{ and } \{17\}.$$

The scattering of α *in* w *is* $\sigma(\alpha, w) = 3$. *Observe that* $\sigma(\alpha, w) \neq \gamma(\alpha, w)$.

In what follows, we consider only substrings α over Σ^* such that $|\alpha| = 2^{q_\alpha} - 1$, for $q_\alpha > 0$. One should observe that the Spoiler needs at most q_α rounds to distinguish α from any other substring β such that $|\beta| = |\alpha|$ and $\alpha \neq \beta$. For example, let *abc* be a substring which occurs in a string w. First, the Spoiler chooses the posision in which b occurs. Then, depending on the choice of the Duplicator, the Spoiler chooses the position in which a occurs or the position in which c occurs.

Lemma 3.3.3. *Let* u *and* v *be strings. Let* α *be a string such that* $|\alpha| = 2^{q_\alpha} - 1$ *and* $\gamma(\alpha, u) \neq \gamma(\alpha, v)$ *or* $\sigma(\alpha, u) \neq \sigma(\alpha, v)$. *Then, for* $r = q_\alpha + min(\sigma(\alpha, \mathscr{A}), \sigma(\alpha, \mathscr{B}))$, *the Spoiler has a winning strategy in* $\mathscr{G}_r(\mathscr{A}, \mathscr{B})$.

Lemma 3.3.4. *Let* r *be a natural number,* u *and* v *be strings. If there exists* α *such that* $|\alpha| = 2^{q_\alpha} - 1$, *for some* $q_\alpha > 0$, $\sigma(\alpha, u) + q_\alpha \leq r$ *or* $\sigma(\alpha, v) + q_\alpha \leq r$, *and* $\sigma(\alpha, u) \neq \sigma(\alpha, v)$ *or* $\gamma(\alpha, u) \neq \gamma(\alpha, v)$, *then the Spoiler has a winning strategy in* $\mathscr{G}_r(u, v)$.

Theorem 3.3.1. *(MONTANARI et al., 2005) Let* r *be a natural number,* u *and* v *be strings. The Spoiler has a winning strategy in* $\mathscr{G}_r(u, v)$ *if and only if at least one of following conditions holds:*

1. $|u| \neq |v|$ *and* $(|u| < 2^r - 2$ *or* $|v| < 2^r - 2)$;
2. $pref_{2^r-2}(u) \neq pref_{2^r-2}(v)$;
3. $suff_{2^r-2}(u) \neq suff_{2^r-2}(v)$;
4. *there exists* α *such that* $|\alpha| = 2^{q_\alpha} - 1$ *for some* $q_\alpha > 0$ *and* $(\sigma(\alpha, u) \neq \sigma(\alpha, v)$ *or* $\gamma(\alpha, u) \neq \gamma(\alpha, v))$ *such that* $\sigma(\alpha, u) + q_\alpha \leq r$ *or* $\sigma(\alpha, v) + q_\alpha \leq r$.

In (MONTANARI *et al.*, 2005), Theorem 3.3.1 is also used to define a notion of similarity between strings. This notion is based on the minimum number of rounds r such that the Spoiler has a winning strategy in the game $\mathscr{G}_r(u, v)$. First, assume that $u \neq v$. Then, $MinRound(u, v)$ can

be computed in polynomial time in the length of the strings in the following way.

$$
\begin{aligned}
MinRound(u,v) := \; & min\{MinLength(u,v), MinPref(u,v), \\
& MinSuff(u,v), MinSub(u,v)\}, \text{ such that} \\
MinLength(u,v) := \; & \lceil log_2(min(|u|,|v|)-1)\rceil, \\
MinPref(u,v) := \; & min\{\lceil log_2(k)\rceil \mid pref_k(u) \neq pref_k(v)\}, \\
MinSuff(u,v) := \; & min\{\lceil log_2(k)\rceil \mid suff_k(u) \neq suff_k(v)\}, \\
MinSub(u,v) := \; & min\{q_\alpha + min(\sigma(\alpha,u),\sigma(\alpha,v)) \mid \gamma(\alpha,u) \neq \gamma(\alpha,v) \\
& \text{ or } \sigma(\alpha,u) \neq \sigma(\alpha,v)\}.
\end{aligned}
$$

Given two strings u and v, $MinRound(u,v)$ can be computed in $O((|u|+|v|)^2 log(|u|+|v|))$, that is, it can be computed in polynomial time (MONTANARI *et al.*, 2005).

In the following chapter, we review the related work on synthesis of propositional formulas, first-order formulas and deterministic finite automata.

4 SYNTHESIS OF FORMULAS AND AUTOMATA

In this chapter, we present the related work concerning synthesis of formulas from a logic and synthesis of deterministic finite automata.

4.1 Boolean Function Synthesis Problem

The Boolean Function Synthesis (BFS) problem (TRIANTAPHYLLOU, 2010) is the task to find a propositional formula in disjunctive normal form (DNF) with the minimum number of conjunctive clauses that is consistent with a given sample of valuations. The valuations are described by the presence or absence of certain features and are divided into two groups: positive and negative. Table 4 represents an instance of the problem. This sample contains information concerning credit approval for individuals. Each valuation represents an individual, and it is defined by three features: has children (c), married (m), and graduated (g). The column Group encodes whether credit was approved for an individual.

Table 4 – A sample of examples.

c	m	g	Group
True	False	True	Positive
False	False	True	Negative
True	False	False	Positive
True	True	True	Negative
False	True	True	Positive

Source: Own elaboration

Observe that the formula $(\neg c \wedge m) \vee (c \wedge \neg m)$ holds in all positive valuations while it is false in all negative ones. Then, this formula is a solution to the instance presented in Table 4. There are many reasons why one may be interested in a formula in DNF with the minimum number of clauses. One may be interested in obtaining a compact set of rules which satisfy the requirements of valuations. A solution is a formula that expresses separations among groups, and it is intended to represent logical relations connecting features with groups. For example, solutions to this problem and its variations are used in finance, industrial and medical applications (KNIJNENBURG *et al.*, 2016; LEJEUNE *et al.*, 2018).

Observe that, in this work, we consider a variation of the BFS problem where positive and negative valuations are finite relational structures. In addition, we consider first-

order sentences instead of propositional logic.

Now, we formally define the BFS problem. A sample of valuations $E = (E^+, E^-)$ consists of two disjoint, finite sets $E^+ = \{\mathscr{V}_1^+, ..., \mathscr{V}_s^+\}$, $E^- = \{\mathscr{V}_1^-, ..., \mathscr{V}_t^-\}$ of valuations. The positive examples are the valuations in E^+, and the negative examples are the valuations in E^-. The BFS problem is defined formally in the following way:

Definition 4.1.1 (BFS Problem)**.** *Given a set of propositional variables* Ψ *and a sample E of valuations over* Ψ*, the goal is to find a propositional formula* ψ *in DNF with the minimum number of conjunctive clauses such that* $\mathscr{V}^+ \models \varphi$*, for all* $\mathscr{V}^+ \in E^+$*,* $\mathscr{V}^- \not\models \varphi$*, for all* $\mathscr{V}^- \in E^-$*, and* $A(\psi) \subseteq \Psi$*.*

We also use BFS for the decision version of the problem where the goal is to find a propositional DNF formula φ with m clauses such that φ is consistent with S. Although the BFS problem is NP-complete (UMANS *et al.*, 2006), there are many solutions effective in practice to the BFS problem and its variations (KAMATH *et al.*, 1992; FELICI; TRUEMPER, 2005; KNIJNENBURG *et al.*, 2016; IGNATIEV *et al.*, 2018; LEJEUNE *et al.*, 2018; MALIOUTOV; MEEL, 2018; GHOSH; MEEL, 2019). An effective solution in practice is to translate it into the satisfiability (SAT) problem of propositional logic (KAMATH *et al.*, 1992; IGNATIEV *et al.*, 2018).

Our approach to solve the synthesis problem for quantifier-free normal form sentences by using a Boolean satisfiability (SAT) encoding is based on a method to solve the BFS problem. Also, we show that the synthesis problem for QDNF sentences is NP-complete by a reduction from the BFS problem. Recall that the difference is that we consider classified finite relational structures, whereas BFS deals with positive and negative valuations. Also, our goal is to find a first-order sentence, while the goal in BFS is to find a propositional formula.

4.2 Distinguishing Relational Structures

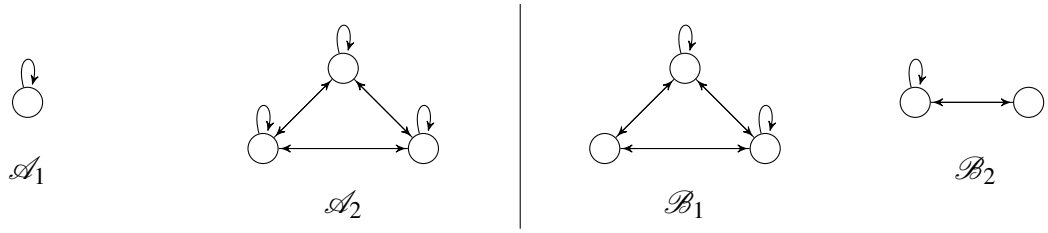
In this section, we show the framework defined in (KAISER, 2012). In this framework, for a fixed logic $\mathscr{L}$, given two sets of structures P and N, the goal is to find a sentence of $\mathscr{L}$ with minimum quantifier rank which distinguishes P from N. Then, this sentence must hold in all structures belonging to P, and it must be false in all structures belonging to N. We formally define the problem below.

Definition 4.2.1 (Distinguishability Problem)**.** *Let $\mathscr{L}$ be a logic. Given two disjoint, finite sets $P = \{\mathscr{A}_1, ..., \mathscr{A}_k\}$ and $N = \{\mathscr{B}_1, ..., \mathscr{B}_l\}$ of structures such that $\mathscr{A}$ and $\mathscr{B}$ are not isomorphic, for all $\mathscr{A} \in P$, $\mathscr{B} \in N$, the Distinguishability problem is to find a sentence φ of $\mathscr{L}$ with minimum quantifier rank such that $\mathscr{A} \models \varphi$, for all $\mathscr{A} \in P$ and, $\mathscr{B} \not\models \varphi$, for all $\mathscr{B} \in N$.*

Observe that, in the above definition, $\mathscr{A}$ and $\mathscr{B}$ are not isomorphic, for all $\mathscr{A} \in P$, $\mathscr{B} \in N$. Otherwise, no first-order sentence distinguishes P from N. In the following, we give an example of an instance of the problem.

Example 4.2.1. *Let $P = \{\mathscr{A}_1, \mathscr{A}_2\}$ and $N = \{\mathscr{B}_1, \mathscr{B}_2\}$ be two sets of structures pictured in Figure 13. The first-order sentence $\forall x_1 E(x_1, x_1)$ is a solution to the Distinguishability problem.*

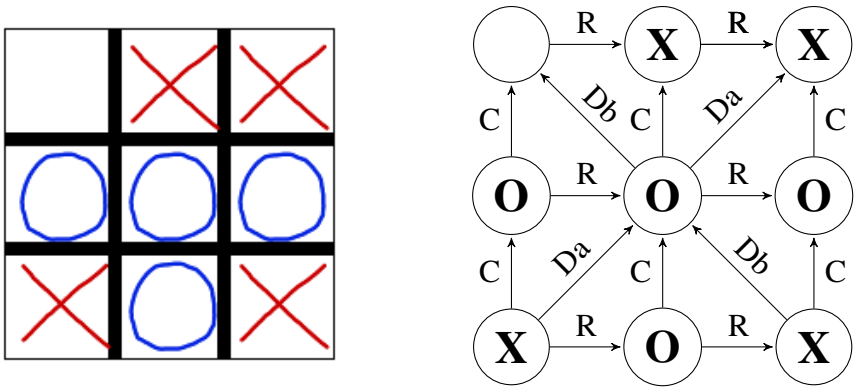
Figure 13 – Example of P and N.



Source: Own elaboration

Kaiser used this framework in the problem of learning winning conditions for both players in board games such as Connect4, Breakthrough, and Tic-Tac-Toe. First, one may represent states of board games as relational structures. For example, Figure 14 represents a state of Tic-Tac-Toe as a relational structure.

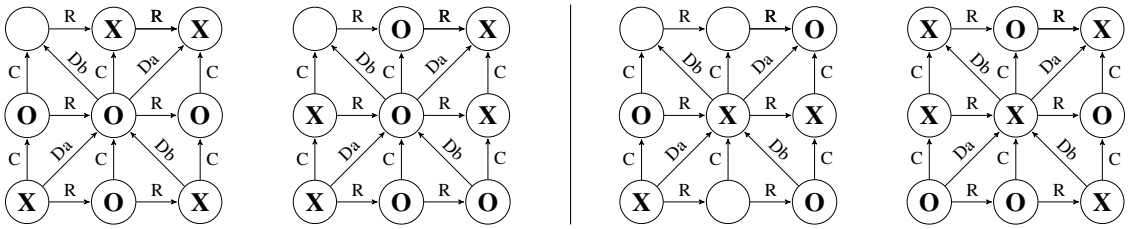
Figure 14 – Tic-Tac-Toe as a relational structure.



Source: Own elaboration

In Figure 15, we give another example with 2 positive structures in P and 2 negative structures in N. Each positive structure represents a winning state of Tic-Tac-Toe, while a negative structure corresponds to a not winning state.

Figure 15 – 2 winning states for the player "O" and 2 not winning.



Source: Own elaboration

Example 4.2.2. *Let P and N be the sets of structures defined in Figure 15. Therefore, the first-order sentence*

$$\exists x_1 \exists x_2 \exists x_3 (O(x_1) \wedge O(x_2) \wedge O(x_3) \wedge ((R(x_1,x_2) \wedge R(x_2,x_3)) \vee (C(x_1,x_2) \wedge C(x_2,x_3))))$$

holds in all positive structures, and does not hold in all negative ones. This first-order sentence specifies winning conditions for the player "O" in Tic-Tac-Toe. Observe that this sentence describes the winning conditions of placing the mark "O" in a row or column of the board.

Unfortunately, in terms of complexity, already for first-order logic and singleton sets P and N, the problem is hard. The decision version, where the quantifier rank is also given, is PSPACE-complete (PEZZOLI, 1998). In Chapter 6, we show that this problem is solved in polynomial time for first-order logic over some fixed classes of structures such as monadic structures, equivalence structures, disjoint unions of linear orders, and strings.

The formula returned by the procedure defined in (KAISER, 2012) uses the minimum number of variables k and has minimum quantifier rank among k-variable formulas distinguishing P from N. Furthermore, it belongs to the guarded fragment if possible, and is existential if possible. These fragments are used because they have good algorithmic properties (GRÄDEL, 1999). As the returned formula uses the minimum number of variables, the procedure in (KAISER, 2012) adopts Hintikka formulas for the k-variable fragments of first-order logic.

Since, in this work, we are not worried about the number of variables, we present an adapted version of the procedure in (KAISER, 2012) which uses Hintikka formulas for full first-order logic. Also, as we do not consider the fragments of first-order logic used in (KAISER,

2012), the version presented in this work uses only full first-order logic. In what follows, we present this setting for the algorithm that returns a formula φ that holds in all positive structures and on none of the negative ones.

We start with a procedure that receives as input the quantifier rank r, the set of positive structures P, and the set of negative structures N. The algorithm computes the set $\mathscr{P}$ of r-Hintikka formulas of all structures in P. Then, for every structure $\mathscr{B}$ in N, it checks whether $\varphi^r_{\mathscr{B}} \in \mathscr{P}$. If there exists $\mathscr{B}$ such that $\varphi^r_{\mathscr{B}} \in \mathscr{P}$, then it is not possible to distinguish P from N with a first-order sentence with quantifier rank r. Otherwise, the algorithm returns the sentence $\varphi = \bigvee \mathscr{P}$ which have quantifier rank r. Then, clearly, φ hold in each structure in P. Moreover, φ does not hold in any structure in N, otherwise, $\varphi^r_{\mathscr{B}} \in \mathscr{P}$, for some $\mathscr{B} \in N$. The procedure is described in Algorithm 1.

Algorithm 1: Distinguish(P, N, r)

```
1: 𝒩 ← {φ^r_ℬ | ℬ ∈ N}
2: 𝒫 ← {φ^r_𝒜 | 𝒜 ∈ P}
3: if 𝒫 ∩ 𝒩 ≠ ∅ then
4:     return False
5: end if
6: return ⋁ 𝒫
```

Algorithm 1 is used iteratively, starting from the smallest r. The overall procedure is described in Algorithm 2.

Algorithm 2: MinDistinguish(P,N)

```
1: r ← 1
2: while True do
3:     φ ← Distinguish(P,N,r)
4:     if φ ≠ False then
5:         return φ
6:     end if
7:     r ← r + 1
8: end while
```

Algorithm 2 finds first-order sentences distinguishing P from N with minimum quantifier rank. However, as Algorithm 2 returns a disjunction of Hintikka formulas, it returns very long sentences which seem hard to read. Then, the algorithm in (KAISER, 2012) greedily remove literals that are not necessary to distinguish P from N.

4.3 Grammatical Inference and DFA Synthesis

Grammatical inference is the scientific area that investigates the task of finding a language model consistent with a given sample of strings (WIECZOREK, 2016). For instance, a language model can be deterministic finite automata (DFA) or context-free grammars. Grammatical inference can be applied in linguistics (STROTHER-GARCIA *et al.*, 2017), robotic planning (RAWAL *et al.*, 2011), and classification of biological sequences (WIECZOREK; UNOLD, 2014; WIECZOREK; UNOLD, 2016). For example, strings in Table 1, repeated below for convenience, represent short segments of proteins.

Table 1 – A sample of strings.

String	Class
stviil	Positive
ktvive	Negative
stviie	Positive
stpiie	Negative

Source: Own elaboration

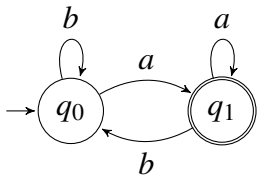
One of the most explored problems in grammatical inference is the DFA synthesis (HIGUERA, 2005). The goal of this problem is to find a DFA with the minimum number of states that is consistent with a given sample of classified strings.

In the DFA synthesis, the task is to find a DFA with the minimum number of states that is consistent with a given sample of classified strings. A sample S over an alphabet Σ is a pair $S = (S_+, S_-)$ such that S_+ and S_- are finite subsets of Σ^* and $S_+ \cap S_- = \emptyset$. Intuitively, S_+ contains positively classified strings, while S_- contains negatively classified strings. An automaton A is consistent with a sample S if and only if $S_+ \subseteq L(A)$ and $S_- \cap L(A) = \emptyset$. Then, A is consistent with S when A accepts all strings in S_+ and rejects all strings in S_-. Now, we can formally define the DFA synthesis task.

Definition 4.3.1. *Given a sample $S = (S_+, S_-)$ over an alphabet Σ and a positive integer k, the DFA synthesis task consists in finding a deterministic finite automaton A with k states such A is consistent with S.*

Example 4.3.1. *Let $k = 2$ and $S = (S_+, S_-)$ such that $S_+ = \{a, aa, ba\}$ and $S_- = \{\varepsilon, ab\}$. The deterministic finite automaton A_1 in Figure 16 is consistent with S. Moreover, it is a minimal consistent DFA since every DFA with just one state accepts either all strings or none.*

Figure 16 – DFA A_1



Source: Own elaboration

Unfortunately, Gold (GOLD, 1978) showed that the corresponding decision problem is hard, namely NP-complete. Intuitively, the reason is that the behavior of a consistent DFA on words not belonging to S_+ and S_- can be arbitrary. Then, an algorithm must choose this behavior carefully in order to find a DFA with the minimum number of states.

Theorem 4.3.1. *(GOLD, 1978) Let S be a sample and k a positive integer. Determining whether there is an automaton with k states and consistent with S is NP-complete.*

However, several methods to solve this problem or a relaxed version were developed (LUCAS; REYNOLDS, 2003; BUGALHO; OLIVEIRA, 2005; HEULE; VERWER, 2010). The methods in (LUCAS; REYNOLDS, 2003; BUGALHO; OLIVEIRA, 2005) cannot guarantee that the found DFA is one with the minimum number of states. The method DFASAT introduced in (HEULE; VERWER, 2010) is guaranteed to find a minimum-size DFA. This approach is based on a series of translations to SAT. The motivation of this approach is that SAT solvers offer a feasible manner for finding solutions. In the following subsection, we describe the method DFASAT (HEULE; VERWER, 2010)

4.3.1 SAT-Based DFA Synthesis

The main part of the method in (HEULE; VERWER, 2010) is to encode the DFA synthesis problem into a propositional CNF formula. This translation is inspired by an encoding of the graph coloring problem into SAT. Given a sample S and a positive integer n, the first step is to construct a propositional formula φ_n^S. This formula φ_n^S is satisfiable if and only if there exists a DFA with n states and consistent with S. Furthermore, it is possible to derive a consistent DFA with n states from a valuation that satisfies φ_n^S. Given the positive integer n, we set $Q = \{q_1, ..., q_n\}$. Given S and n, the propositional formula φ_n^S consists of the following types of propositional variables:

- $x_{u,q}$, for $u \in Pref(S_+ \cup S_-)$ and $q \in Q$;

- f_q, for $q \in Q$;
- $t_{p,a,q}$, for $a \in \Sigma$ and $p,q \in Q$.

If $x_{u,q}$ is assigned to true, then the corresponding DFA reaches state q after processing string u. If f_q is set to true, then q is a final state. Moreover, if the propositional variable $t_{p,a,q}$ is assigned to true, then the corresponding DFA contains the transition $\delta(p,a) = q$. The constraints of these variables are as follows. For each string $u \in Pref(S_+ \cup S_-)$, the DFA must reach exactly one state.

$$\begin{aligned} &\bigwedge_{u \in Pref(S_+ \cup S_-)} \bigvee_{q \in Q} x_{u,q} \\ &\bigwedge_{u \in Pref(S_+ \cup S_-)} \bigwedge_{q,p \in Q, q \neq p} \neg x_{u,q} \vee \neg x_{u,p} \end{aligned} \tag{4.1}$$

We also have to ensure that the variables $t_{p,a,q}$ encode a deterministic function. We use the following constraints.

$$\begin{aligned} &\bigwedge_{p \in Q} \bigwedge_{a \in \Sigma} \bigvee_{q \in Q} t_{p,a,q} \\ &\bigwedge_{p \in Q} \bigwedge_{a \in \Sigma} \bigwedge_{q,q' \in Q, q \neq q'} \neg t_{p,a,q} \vee \neg x_{p,a,q'} \end{aligned} \tag{4.2}$$

Additionally, the following constraints guarantee that variables $x_{u,q}$ and $t_{p,a,q}$ are consistent, i.e., if the DFA reaches p after reading string u and $\delta(p,a,q)$, then $x_{ua,q}$ must be set to true. We also add a constraint ensuring that if the DFA reaches p after reading string u, and it reaches q after reading ua, then $t_{p,a,q}$ has to be set to true.

$$\begin{aligned} &\bigwedge_{ua \in Pref(S_+ \cup S_-)} \bigwedge_{q,p \in Q} \neg x_{u,p} \vee \neg t_{p,a,q} \vee x_{ua,q} \\ &\bigwedge_{ua \in Pref(S_+ \cup S_-)} \bigwedge_{q,p \in Q} \neg x_{u,p} \vee t_{p,a,q} \vee \neg x_{ua,q} \end{aligned} \tag{4.3}$$

Finally, the corresponding DFA must be consistent with S.

$$\begin{aligned} &\bigwedge_{u \in S_+} \bigwedge_{q \in Q} \neg x_{u,q} \vee f_q \\ &\bigwedge_{v \in S_-} \bigwedge_{q \in Q} \neg x_{u,q} \vee \neg f_q \end{aligned} \tag{4.4}$$

Let φ_n^S be the conjunction of formulas in Equations 4.1-4.4. Then, φ_n^S consists of $O(n^3|\Sigma| + |Pref(S_+ \cup S_-)|n^2)$ clauses and $O(n^2|\Sigma| + |Pref(S_+ \cup S_-)|n)$ variables. To find a DFA with the minimum number of states, one can use a SAT solver in an iterative way. To show that the minimum number of states is n, φ_n^S must be satisfiable and φ_{n-1}^S unsatisfiable. Then, one alternative is set n to 1 and then successively increase n by one until φ_n^S is satisfiable.

4.3.2 SAT-Based DFA Synthesis from Noisy Samples

The method described in the previous subsection handles DFA synthesis from noiseless samples. In this subsection, we explain the constraints defined in (ULYANTSEV *et al.*, 2015) to solve the problem of learning DFA from noisy samples. We will call this method DFASAT-Noise. The method presented in (ULYANTSEV *et al.*, 2015) also introduces constraints to reduce SAT search space by enforcing an enumeration of the states in breadth-first search order. However, we do not present this technique in order to keep the presentation as simple as possible.

For the noiseless case, Equation 4.4 imposes that a DFA must be consistent with the sample S. In (ULYANTSEV *et al.*, 2015), the authors adapt these constraints to deal with noisy samples. The idea is to assume that not more than K strings of the sample were erroneous classified. For each string w in the sample, we use a variable e_w such that e_w is set to true if the class of string w can (but does not have to) be incorrect. The formulas in Equations 4.5-4.6 below express that if a string w is in the correct class, then the target DFA must cover w.

$$\bigwedge_{u \in S_+} \bigwedge_{q \in Q} \neg e_u \rightarrow (\neg x_{u,q} \vee f_q). \tag{4.5}$$

$$\bigwedge_{u \in S_-} \bigwedge_{q \in Q} \neg e_v \rightarrow (\neg x_{u,q} \vee \neg f_q). \tag{4.6}$$

Now, we need to limit the number of strings that can be in the wrong class. We use variables $r_{i,w}$, for $1 \leq i \leq K$ and $w \in S_+ \cup S_-$, such that $r_{i,w}$ is set to true if w is the ith string that can be in the wrong class. Equation 4.7 imposes that the class of w can be incorrect iff w is the ith string that can be in the wrong class, for some $i \in \{1, ..., K\}$.

$$\bigwedge_{w \in S_+ \cup S_-} e_w \leftrightarrow (\bigvee_{i=1}^{K} r_{i,w}). \tag{4.7}$$

Clearly, there exists exactly one ith string that can be in the wrong class, for $i \in \{1,...,K\}$. Let $S_+ \cup S_- = \{w_1,...,w_{|S_+ \cup S_-|}\}$. Then, we need to enforce that exactly one of the variables r_{i,w_1}, ..., $r_{i,w_{|S_+ \cup S_-|}}$ is true, for each $i \in \{1,...,K\}$. We use the encoding defined in (BARAHONA *et al.*, 2014) since it is the same considered in (ULYANTSEV *et al.*, 2015). This encoding uses auxiliary order variables $o_{i,w}$, for $i \in \{1,...,K\}$ and $w \in S_+ \cup S_-$. We also need to assume an order on $S_+ \cup S_- = \{w_1,...,w_{|S_+ \cup S_-|}\}$. We assume an arbitrary order $w_1 \prec w_2 \prec \ldots \prec w_{|S_+ \cup S_-|}$. Then, to specify that w_j is the ith string in the order, the first j variables o_{i,w_1}, ..., o_{i,w_j} are set to true and the remaining $|S_+ \cup S_-| - j$ variables $o_{i,w_{j+1}}$, ..., $o_{i,w_{|P \cup N|}}$ are assigned to false. This can be defined by the formula in Equation 4.8.

$$\bigwedge_{1 \leq i \leq K} \bigwedge_{1 \leq j < |S_+ \cup S_-|} o_{i,w_{j+1}} \rightarrow o_{i,w_j}. \tag{4.8}$$

Furthermore, in order to avoid permutations of the strings that can be in the incorrect class, the formula in Equation 4.9 enforces that if w is the ith string in the order, then the $(i+1)$th string w' in the order is such that $w \prec w'$.

$$\bigwedge_{1 \leq i < K} \bigwedge_{1 \leq j < |S_+ \cup S_-|} o_{i,w_j} \rightarrow o_{i+1,w_{j+1}}. \tag{4.9}$$

Finally, we need the formulas in Equations 4.10-4.11 to impose that exactly one $r_{i,w}$ is true, for $i \in \{1,...,K\}$. Variables $o_{i,w}$ enforce this in variables $r_{i,w}$.

$$\bigwedge_{1 \leq i \leq K} \bigwedge_{1 \leq j < |S_+ \cup S_-|} r_{i,w_j} \leftrightarrow o_{i,w_j} \wedge \neg o_{i,w_{j+1}}. \tag{4.10}$$

$$o_{K,w_{|P \cup N|}} \leftrightarrow r_{K,w_{|S_+ \cup S_-|}}. \tag{4.11}$$

Formulas in Equations 4.5-4.11 can easily be translated to CNF. Let $\varphi^S_{n,K}$ be the conjunction of formulas in Equations 4.1-4.11 in CNF. Then, $\varphi^S_{n,K}$ has $O(n^2|\Sigma| + |Pref(S_+ \cup S_-)|n + K \times |S_+ \cup S_-|)$ variables, and it consists of $O(n^3|\Sigma| + |Pref(S_+ \cup S_-)|n^2 + |S_+ \cup S_-| \times K)$ clauses.

4.4 Grammatical Inference and Finite Model Theory

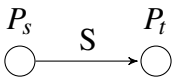
A recent logical approach to grammatical inference defined in (STROTHER-GARCIA *et al.*, 2017) uses a propositional language in which the atomic sentences represent substrings which are taken to be true for a string if and only if they occur in that string as infix, prefix or suffix. Formulas in (STROTHER-GARCIA *et al.*, 2017) are defined by conjunctions of negative and positive literals (CNPL). In our approach, we also contemplate multiplicity and scattering of substrings, disjunctions, and length of strings. Then, the main difference is that our framework deals with the expressive power of full first-order logic, while the approach in (STROTHER-GARCIA *et al.*, 2017) uses a less fragment less expressive.

In our work, we represent strings as logical structures with a successor order relation such that symbols are represented by unary relations. Clearly, not all structures under this vocabulary may be seen as strings. For example, in the alphabet $\Sigma = \{a,b\}$, the structure $\mathscr{A} = \langle \{1,2,3\} P_a^{\mathscr{A}} = \{1,3\}, P_b^{\mathscr{A}} = \{3\} \rangle$ can not be seen as a string in Σ^*. It is clear that in order to be seen as a string, each position in a structure must belong to exactly one unary relation.

The work in (STROTHER-GARCIA *et al.*, 2017) consider strings as logical structures that do not assume each position in a string belongs to exactly one unary predicate. They argue that this non-standard string model can be considered in some fields where a flexible model is required. For example, in biology, sequences of symbols in $\{a,c,g,t\}$ are used to represent sequences of nucleotides in DNA structures. However, a and t are complementary while a and g are considered independent. This feature can be represented in the richer model by positions which are true for both P_a and P_t.

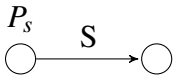
For a structure $\mathscr{A}$, we define $substructs_k(\mathscr{A})$ as the set of all substructures $\mathscr{B}$ of $\mathscr{A}$ such that S is the successor order relation and $|B| \leq k$. Furthermore, we let $allsubstructs(k)$ be the set of all substructures of all structures $\mathscr{A}$ such that S is the successor order relation and $|A| \leq k$. It is clear that, for a standard string model $\mathscr{A}_w$, not all substructures of $\mathscr{A}_w$ are standard string models. For example, the structure in $\mathscr{A}_{stv}$ in Figure 17 has as substructure the structure in Figure 18.

Figure 17 – String stv as a logical structure $\mathscr{A}_{stv}$



Source: Own elaboration

Figure 18 – A substructure of $\mathscr{A}_{stv}$



Source: Own elaboration

Clearly, the structure in Figure 18 is a non-standard string structure. Let $\mathscr{A}$ be a non-standard string structure. We define first-order sentences

$$\phi_{\mathscr{A}} := \exists x_1 ... \exists x_{|A|} (\bigwedge_{1 \leq i \leq |A|-1} S(x_i, x_{i+1}) \wedge \bigwedge_{1 \leq i \leq k, j \in P_{a_i}} P_{a_i}(x_j)).$$

For example, for the structure $\mathscr{B}$ in Figure 18, $\phi_{\mathscr{B}} = \exists x_1 \exists x_2 (S(x_1, x_2) \wedge P_s(x_1))$. When $\mathscr{B}$ represents some string w, we use the notation ϕ_w. Then, ϕ_w is similar to the formula defined in Section 2.5. Similarly, we use the notions of propositional sentences, literals and conjunction of negative literals for first-order logic over non-standard string structures.

Lemma 4.4.1. *Let w be a string and $\mathscr{B}$ a non-standard string structure. $\mathscr{B}$ is a substructure of $\mathscr{A}_w$ iff $w \models \phi_{\mathscr{B}}$.*

For example, $stv \models \phi_{\mathscr{B}}$ for $\mathscr{B}$ in Figure 18. It follows that a string u is a substring of a string w iff $w \models \phi_u$. A propositional first-order sentence is a k-sentence if it is a Boolean combination of formulas of the form ϕ_w such that the longest string w has length at most k. Now, we present an example from phonology in order to show the advantage of using a non-standard string model.

In phonology, stress is a relative emphasis given to a certain syllable in a word. Usually, there are two syllable weights: light (L) and heavy (H). We use the acute accent to denote stress. Then, $\acute{H}$ represents a heavy stressed syllable. We consider the alphabet $\Sigma = \{L, H, \acute{L}, \acute{H}\}$. Then, we are interested in languages over Σ^* which represent stress patterns in natural languages.

For example, in the Cambodian language, there are no consecutive light syllables (LAMBERT; ROGERS, 2019). Moreover, a universal constraint across languages is that each word has at least one syllable with stress. The following sentence defines these constraints.

$$\phi_1 = \neg\phi_{LL} \wedge \neg\phi_{L\acute{L}} \wedge \neg\phi_{\acute{L}L} \wedge \neg\phi_{\acute{L}\acute{L}} \wedge (\phi_{\acute{L}} \vee \phi_{\acute{H}}).$$

Clearly, ϕ_1 is a propositional first-order 2-sentence that defines a LT language. Furthermore, ϕ_1 is a CNF sentence which has a clause with two literals. Now, we show that

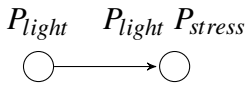
using non-standard string structures, we can define an equivalent formula such that all clauses have only one literal.

First, we consider a class of languages defined by conjunctions of negative and positive literals (CNPL). Then, CNPL sentences are CNF formulas such that each clause contains exactly one literal. Then, the class of languages defined by CNPL is between LT and SL.

Now, we consider a non-standard string model for stress patterns. Then, we use the following unary predicates P_{light}, P_{heavy} and P_{stress}. Therefore, for a string $w = w_1...w_n$ such that position $w_i = \acute{L}$, it follows that $i \in P_{light}$ and $i \in P_{stress}$.

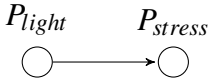
We use a shorthand for structures in this non-standard string model. A position i in a structure such that $i \notin P_{light} \cup P_{heavy} \cup P_{stress}$ is represented by $*$. A position i in a structure such that $i \notin P_{light} \cup P_{heavy}$ and $i \in P_{stress}$ is represented by $\acute{*}$. H and L represent heavy and light positions that are unspecified for stress. A position $i \in P_{stress} \cap P_{heavy}$ is represented by $\acute{H}$. It is analogous for $\acute{L}$. Figure 19 pictures a structures represented by $L\acute{L}$. The substructure of $L\acute{L}$ in Figure 20 is represented by $L\acute{*}$

Figure 19 – Structure represented by $L\acute{L}$



Source: Own elaboration

Figure 20 – Structure represented by $L\acute{*}$



Source: Own elaboration

Observe that $L\acute{*}$ is a substructure of $L\acute{L}$ and $L\acute{H}$. Now, the following CNPL 2-sentence ϕ_2 is equivalent to ϕ_1.

$$\phi_2 = \neg\phi_{LL} \wedge \neg\phi_{L\acute{L}} \wedge \neg\phi_{\acute{L}L} \wedge \neg\phi_{\acute{L}\acute{L}} \wedge \phi_{\acute{*}}.$$

The sentence ϕ_2 shows an advantage of using the non-standard string model. Formulas can be simplified by using this model. As another example, let $\phi_3 = \neg\phi_{LL} \wedge \neg\phi_{L\acute{L}} \wedge \neg\phi_{LH} \wedge \neg\phi_{L\acute{H}}$ be a CNL 2-sentence. Clearly, $\neg\phi_{L*}$ is equivalent to ϕ_3.

Now, we show how stress patterns can be obtained from positive data using non-standard string structures. In the following, we show an algorithm that, given a sample of positive strings P and a positive integer k, returns a CNPL k-sentence consistent with P.

The algorithm finds a CNPL sentence by building a set of permissible structures G_{per} and a set of required structures G_{req}. The set G_{req} consists of substructures of size k that are common to all strings in P. Then, all strings which have these substructures are required in the language. G_{per} is the set of all substructures of size k of all strings in P. Therefore, structures in $\overline{G_{per}} = allsubstructs(k) - G_{per}$ may be seen as forbidden substructures. The idea of learning forbidden substructures from permissible ones was also presented in (HEINZ, 2010) and (HEINZ *et al.*, 2012). The overall procedure is defined in Algorithm 3. We finish this section by giving an example to illustrate a run of Algorithm 3.

Algorithm 3:

Input: Sample P, length k
$G_{per} \leftarrow \bigcup_{w \in P} substructs_k(\mathscr{A}_w)$
$G_{req} \leftarrow \bigcap_{w \in P} substructs_k(\mathscr{A}_w)$
$\overline{G_{per}} \leftarrow allsubstructs(k) - G_{per}$
$\phi \leftarrow \bigwedge_{\mathscr{A} \in G_{req}} \phi_{\mathscr{A}} \wedge \bigwedge_{\mathscr{B} \in \overline{G_{per}}} \neg \phi_{\mathscr{B}}$
return ϕ

Example 4.4.1. *Let* $P = \{\acute{L}\acute{H}\acute{L}, \acute{L}, \acute{H}\acute{H}\}$ *and* $k = 2$. *Then,*

$$substructs_2(\acute{L}) = \{\acute{L}, *, \acute{*}\};$$

$$substructs_2(\acute{L}\acute{H}\acute{L}) = \{\acute{L}\acute{H}, \acute{L}H, \acute{L}*, \acute{L}\acute{*}, L\acute{H}, LH, L*, L\acute{*}, *\acute{H}, *H, **, *\acute{*}, \acute{*}\acute{H}, \acute{*}H, \acute{*}*, \acute{*}\acute{*},$$
$$\acute{H}\acute{L}, \acute{H}L, \acute{H}\acute{*}, \acute{H}*, H\acute{L}, HL, H\acute{*}, H*, \acute{*}\acute{L}, \acute{*}L, *\acute{L}, *L, L, H, \acute{L}, \acute{H}, *, \acute{*}\};$$

$$substructs_2(\acute{H}\acute{H}) = \{\acute{H}\acute{H}, \acute{H}H, \acute{H}*, \acute{H}\acute{*}, H\acute{H}, HH, H*, H\acute{*}, *\acute{H}, *H, **, *\acute{*}, \acute{*}\acute{H}, \acute{*}H, \acute{*}*, \acute{*}\acute{*},$$
$$H, \acute{H}, *, \acute{*}\}.$$

Clearly, $G_{req} = \{*, \acute{*}\}$. *Also,* $\overline{G_{per}} = \{LL, L\acute{L}, \acute{L}L, \acute{L}\acute{L}\}$. *Therefore, given P and k, Algorithm 3 returns*

$$\phi = \neg\phi_{LL} \wedge \neg\phi_{L\acute{L}} \wedge \neg\phi_{\acute{L}L} \wedge \neg\phi_{\acute{L}\acute{L}} \wedge \phi_{*} \wedge \phi_{\acute{*}}$$

which is equivalent to ϕ_2.

In the following chapter, we present our first contribution in this work. We show results on EF games for disjoint unions of linear orders.

5 EF GAMES FOR DISJOINT UNIONS OF LINEAR ORDERS

In this chapter, we show conditions characterizing the winning strategies for both players on disjoint unions of linear orders (DULO). DULO are appealing because they may be used to model states of the elementary blocks world.A state of the elementary blocks world consists of cubic blocks, with the same size and color, sitting on a table. Our result on characterizing the winning strategies on DULO follows that of EF games on equivalence structures. First, we turn to linear orders since they are a particular case of DULO. The results in this chapter were published in (ROCHA *et al.*, 2019).

5.1 EF Games for Linear Orders

A linear order is a structure $\mathscr{L} = \langle L, <^{\mathscr{L}} \rangle$ such that $<^{\mathscr{L}}$ is a linear order on L. In what follows, for a linear order $\mathscr{L}$, let $q^{\mathscr{L}}$ be the number of elements in the domain of $\mathscr{L}$. Also, let $min^{\mathscr{L}}$ and $max^{\mathscr{L}}$ be the least element and the greatest element in $\mathscr{L}$, respectively. For linear orders $\mathscr{L}_1$ and $\mathscr{L}_2$, we assume that the elements in $\mathscr{L}_1$ and $\mathscr{L}_2$ are $a_1, ..., a_n$ and $b_1, ..., b_m$, respectively. Also, we assume that $a_i < a_j$ and $b_i < b_j$, for $i < j$. The results in this section are well known in the literature (GRÄDEL *et al.*, 2005).

If the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$, then (s)he has a winning strategy in which (s)he responds to the least element of one linear order by the least element of the other linear order. This also holds for the greatest element. For example, suppose that the Spoiler chooses $min^{\mathscr{L}_1}$ in $\mathscr{L}_1$. If this is the last round, then the Duplicator can select $min^{\mathscr{L}_2}$. If there is at least one round left and the Duplicator chooses b in $\mathscr{L}_2$ such that $min^{\mathscr{L}_2} < b$, then the Spoiler selects $min^{\mathscr{L}_2}$ and wins the game.

Example 5.1.1. *Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be the linear orders below.*

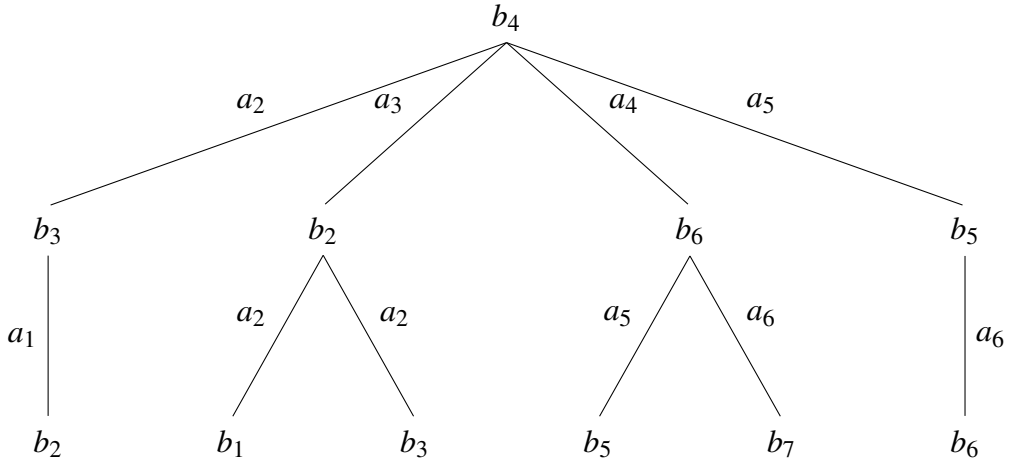
$$\mathscr{L}_1 : a_1 < a_2 < a_3 < a_4 < a_5 < a_6$$

$$\mathscr{L}_2 : b_1 < b_2 < b_3 < b_4 < b_5 < b_6 < b_7$$

Let $r = 3$. The Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$. Observe that $q^{\mathscr{L}_1} \neq q^{\mathscr{L}_2}$ and $q^{\mathscr{L}_1} < 2^r - 1$. The Spoiler's first move is to choose b_4 in $\mathscr{L}_2$. If the Duplicator chooses an element distinct from a_3 and a_4 in $\mathscr{L}_1$, (s)he clearly loses in the last round. Then, the Duplicator chooses a_3 or a_4 in $\mathscr{L}_1$. Both cases are analogous. Then, assume that the Duplicator chooses a_3 in $\mathscr{L}_1$. Now, the Spoiler chooses b_2 in $\mathscr{L}_2$. Then, the Duplicator must choose a_1 or a_2 in $\mathscr{L}_1$. If

the Duplicator chooses a_1 in $\mathscr{L}_1$, then the Spoiler wins by selecting b_1 in $\mathscr{L}_2$. If the Duplicator chooses a_2 in $\mathscr{L}_1$, then the Spoiler chooses b_3 in $\mathscr{L}_2$ and wins the game. Figure 21 shows the Spoiler's winning strategy in $\mathscr{G}_3(\mathscr{L}_1, \mathscr{L}_2)$.

Figure 21 – The Spoiler's winning strategy in $\mathscr{G}_3(\mathscr{L}_1, \mathscr{L}_2)$



Source: Own elaboration

By the symmetry in Figure 21, a EF game on linear orders may be seen as a composition of simpler games. For an element a, we define $\mathscr{L}^{>a}$ ($\mathscr{L}^{<a}$) as a substructure of $\mathscr{L}$ such that $\{b \in L \mid b > a\}$ ($\{b \in L \mid b < a\}$) is the domain of $\mathscr{L}^{>a}$ ($\mathscr{L}^{<a}$).

Lemma 5.1.1. *Let k be a positive integer, and $\mathscr{L}_1$, $\mathscr{L}_2$ be two linear orders. The Duplicator has a winning strategy in $\mathscr{G}_{k+1}(\mathscr{L}_1, \mathscr{L}_2)$ if and only if the following two conditions hold:*

- *For every $a \in \mathscr{L}_1$, there exists $b \in \mathscr{L}_2$ such that the Duplicator has a winning strategy in $\mathscr{G}_k(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$ and $\mathscr{G}_k(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$;*
- *For every $b \in \mathscr{L}_2$, there exists $a \in \mathscr{L}_1$ such that the Duplicator has a winning strategy in $\mathscr{G}_k(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$ and $\mathscr{G}_k(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$.*

Proof. ($\Rightarrow$) Assume that the Duplicator has a winning strategy in $\mathscr{G}_{k+1}(\mathscr{L}_1, \mathscr{L}_2)$. By definition, for every $a \in \mathscr{L}_1$, there is a $b \in \mathscr{L}_2$ such that the Duplicator has a winning strategy in $\mathscr{G}_k((\mathscr{L}_1, a), (\mathscr{L}_2, b))$ and for every $b \in \mathscr{L}_2$, there is an $a \in \mathscr{L}_1$ such that the Duplicator has a winning strategy in $\mathscr{G}_k((\mathscr{L}_1, a), (\mathscr{L}_2, b))$. Then, the Duplicator can use the same winning strategies in the simpler games $\mathscr{G}_k(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$, $\mathscr{G}_k(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$, $\mathscr{G}_k(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$ and $\mathscr{G}_k(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$.
($\Leftarrow$) Now, we need to show that the Duplicator has a winning strategy in $\mathscr{G}_{k+1}(\mathscr{L}_1, \mathscr{L}_2)$. If the Spoiler chooses a in $\mathscr{L}_1^{<a}$, then the Duplicator uses his/her winning strategy in $\mathscr{G}_k(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$.

There is such a b by hypothesis. If the Spoiler chooses $a \in \mathscr{L}_1^{>a}$, then the Duplicator uses his/her winning strategy in $\mathscr{G}_k(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$. The case such that the Spoiler picks an element in $\mathscr{L}_2$ is analogous. □

Before presenting the main theorem, we give another example.

Example 5.1.2. *Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be the linear orders below.*

$$\mathscr{L}_1 : a_1 < a_2 < a_3 < a_4 < a_5 < a_6 < a_7$$
$$\mathscr{L}_2 : b_1 < b_2 < b_3 < b_4 < b_5 < b_6 < b_7 < b_8$$

Different from Example 5.1.1, $q^{\mathscr{L}_1} \neq q^{\mathscr{L}_2}$, $q^{\mathscr{L}_1} \geq 2^r - 1$, and $q^{\mathscr{L}_2} \geq 2^r - 1$, for $r = 3$. The Duplicator has a winning strategy in $\mathscr{G}_3(\mathscr{L}_1, \mathscr{L}_2)$. If the Spoiler chooses b_4 in $\mathscr{L}_2$, then the Duplicator chooses a_4 in $\mathscr{L}_1$. If the Spoiler chooses b_6 in $\mathscr{L}_2$, then the Duplicator selects a_6 in $\mathscr{L}_1$. Clearly, there is one more round left, and the Duplicator wins this run of the game.

Theorem 5.1.1 (EF Games on LO)**.** *Let r be a natural number, $\mathscr{L}_1$ and $\mathscr{L}_2$ be linear orders. The Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$ if and only if $q^{\mathscr{L}_1} \neq q^{\mathscr{L}_2}$ and ($q^{\mathscr{L}_1} < 2^r - 1$ or $q^{\mathscr{L}_2} < 2^r - 1$).*

Proof. ($\Rightarrow$) By contrapositive, assume that $q^{\mathscr{L}_1} = q^{\mathscr{L}_2}$ or ($q^{\mathscr{L}_1} \geq 2^r - 1$ and $q^{\mathscr{L}_2} \geq 2^r - 1$). If $q^{\mathscr{L}_1} = q^{\mathscr{L}_2}$, then, the Duplicator's winning strategy is the following. If the Spoiler chooses an element a_i in $\mathscr{L}_1$, then the Duplicator chooses b_i in $\mathscr{L}_2$. The case such that the Spoiler chooses b_i in $\mathscr{L}_2$ is analogous.

For the other case, suppose that $q^{\mathscr{L}_1} \geq 2^r - 1$ and $q^{\mathscr{L}_2} \geq 2^r - 1$. We proceed by induction on r. If $r = 1$, then $q^{\mathscr{L}_1} \geq 1, q^{\mathscr{L}_2} \geq 1$. It is trivial to see that the Duplicator has a winning strategy.

As inductive hypothesis, we assume that if $q^{\mathscr{L}_1} \geq 2^r - 1$ and $n \geq 2^r - 1$, then the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$. Now, assume that $q^{\mathscr{L}_1} \geq 2^{r+1} - 1$ and $n \geq 2^{r+1} - 1$. Suppose that the Spoiler chooses a in $\mathscr{L}_1$. We need to show that the there exists b in $\mathscr{L}_2$ such that the Duplicator has a winning strategy in $\mathscr{G}_r((\mathscr{L}_1, a), (\mathscr{L}_2, b))$. We have three cases to consider.

- Assume that $q^{\mathscr{L}_1^{<a}} < 2^r - 1$. Let b be an element in $\mathscr{L}_2$ such that $d(min^{\mathscr{L}_1}, a) = d(min^{\mathscr{L}_2}, b)$. Then, the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$. Since $q^{\mathscr{L}_1^{<a}} = q^{\mathscr{L}_2^{<b}} <$

2^r-1, then $q^{\mathscr{L}_1^{>a}} \geq 2^r-1$ and $q^{\mathscr{L}_2^{>b}} \geq 2^r-1$. Then, by inductive hypothesis, the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$.

- The case such that $q^{\mathscr{L}_1^{>a}} < 2^r-1$ is analogous.
- Suppose that $q^{\mathscr{L}_1^{<a}} \geq 2^r-1$ and $q^{\mathscr{L}_1^{>a}} \geq 2^r-1$. Since $q^{\mathscr{L}_2} \geq 2^{r+1}-1$, there exists b in $\mathscr{L}_2$ such that $q^{\mathscr{L}_2^{<b}} \geq 2^r-1$ and $q^{\mathscr{L}_2^{>b}} \geq 2^r-1$. By inductive hypothesis, the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$ and $\mathscr{G}_r(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$.

The case such that the Spoiler chooses an element b in $\mathscr{L}_2$ is analogous. In order to conclude, it holds that for every $a \in \mathscr{L}_1$, there is $b \in \mathscr{L}_2$ such that the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$ and $\mathscr{G}_r(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$. Also, it holds that for every b in $\mathscr{L}_2$, there is an a in $\mathscr{L}_1$ such that the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$ and $\mathscr{G}_r(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$. By Lemma 5.1.1, it follows that the Duplicator has a winning strategy in $\mathscr{G}_{r+1}(\mathscr{L}_1, \mathscr{L}_2)$.

($\Leftarrow$) Conversely, assume that $q^{\mathscr{L}_1} \neq q^{\mathscr{L}_2}$ and ($q^{\mathscr{L}_1} < 2^r-1$ or $q^{\mathscr{L}_2} < 2^r-1$). We show that the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$ by induction on r. If $r=1$, then $q^{\mathscr{L}_1} = q^{\mathscr{L}_2} = 0$. It follows that the implication holds by vacuity.

As inductive step, we assume that $q^{\mathscr{L}_1} \neq q^{\mathscr{L}_2}$ and ($q^{\mathscr{L}_1} < 2^{r+1}-1$ or $q^{\mathscr{L}_2} < 2^{r+1}-1$). Also, we assume, without loss of generality, that $q^{\mathscr{L}_1} > q^{\mathscr{L}_2}$. The Spoiler chooses $a = a_{\lceil \frac{q^{\mathscr{L}_1}}{2} \rceil}$ in $\mathscr{L}_1$. Then, the Duplicator selects an element b in $\mathscr{L}_2$. Then, $q^{\mathscr{L}_2^{>b}} < 2^r-1$ or $q^{\mathscr{L}_2^{<b}} < 2^r-1$. Also, if $d(min^{\mathscr{L}_1}, a) = d(min^{\mathscr{L}_2}, b)$, then $d(a, max^{\mathscr{L}_1}) \neq d(b, max^{\mathscr{L}_2})$ and $q^{\mathscr{L}_2^{>b}} < 2^r-1$. Similarly, if $d(a, max^{\mathscr{L}_1}) = d(b, max^{\mathscr{L}_2})$. Then, without loss of generality, we assume that $d(min^{\mathscr{L}_1}, a) \neq d(min^{\mathscr{L}_2}, b)$ and $q^{\mathscr{L}_2^{<b}} < 2^r-1$. By inductive hypothesis, the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$. Then, there exists a in $\mathscr{L}_1$ such that for every b in $\mathscr{L}_2$, the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1^{<a}, \mathscr{L}_2^{<b})$. Then, by Lemma 5.1.1, the Spoiler has a winning strategy in $\mathscr{G}_{r+1}(\mathscr{L}_1, \mathscr{L}_2)$. □

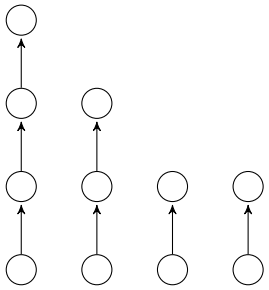
One should note that, in Theorem 5.1.1, the Spoiler's winning strategy consists of choosing, in his/her first round, an element from the linear order with more elements.

5.2 Disjoint Unions of Linear Orders and EF Games

Now, we define disjoint unions of linear orders. Assume that $\mathscr{L}_1$ and $\mathscr{L}_2$ are linear orders such that $L_1 \cap L_2 = \emptyset$. Then, $\mathscr{L}_1 \uplus \mathscr{L}_2$, the disjoint union of $\mathscr{L}_1$ and $\mathscr{L}_2$, is the structure with domain $L_1 \cup L_2$ and $<^{\mathscr{L}_1 \uplus \mathscr{L}_2} = <^{\mathscr{L}_1} \cup <^{\mathscr{L}_2}$. We represent a disjoint union of linear orders $\mathscr{W} = \langle W, <^{\mathscr{W}} \rangle$ by disjoint unions $(...(\mathscr{L}_1 \uplus \mathscr{L}_2) \uplus ... \uplus \mathscr{L}_l)$. An example of disjoint unions of

linear orders is given in Figure 22.

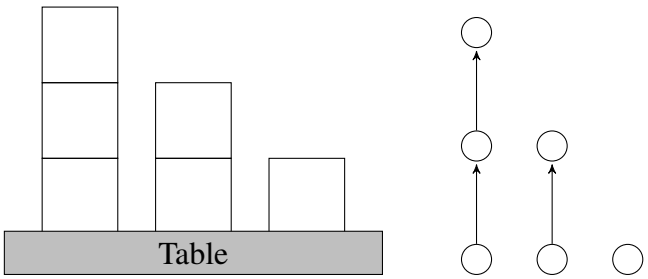
Figure 22 – A disjoint union of linear orders $\mathscr{W}$



Source: Own elaboration

In (COOK; LIU, 2003), disjoint unions of linear orders model states of the elementary blocks world. A state of the elementary blocks world consists of a set of cubic blocks, with the same size and color, sitting on a table. The elementary blocks world is a simple and well-known version of the blocks world. The blocks world is one of the most famous planning domains in artificial intelligence. In artificial intelligence, automated planning is the process of automatically constructing a sequence of actions that achieve a goal given some initial state (GHALLAB *et al.*, 2004). In the elementary blocks world domain, a robot can pick up a block and moves it to another position, either onto the table or on the top of some other block (GUPTA; NAU, 1992). In Figure 23, a state of the elementary blocks world is represented as disjoint unions of linear orders.

Figure 23 – Elementary Blocks World



Source: Own elaboration

Equivalence structures also can be seen as disjoint unions of structures. Our results on DULO are inspired by the case of EF games on equivalence structures. In what follows, for a disjoint union of linear orders $\mathscr{W}$, let $q_t^{\mathscr{W}}$ be the number of linear orders $\mathscr{L}$ in $\mathscr{W}$ such that $q^{\mathscr{L}} = t$. Also, let $q_{\geq t}^{\mathscr{W}}$ be the number of linear orders $\mathscr{L}$ in $\mathscr{W}$ such that $q^{\mathscr{L}} \geq t$. Given an

element a in $\mathscr{W}_1$, $\mathscr{L}(a)$ denotes the linear order in $\mathscr{W}_1$ such that a is in the domain of $\mathscr{L}(a)$. Given a disjoint union of linear orders $\mathscr{W}$, let $\mathscr{W}(a_1,...,a_k)$ be the disjoint union of linear orders obtained by removing $\mathscr{L}(a_1)$, ..., $\mathscr{L}(a_k)$ from $\mathscr{W}$.

Definition 5.2.1 (Disparity)**.** *Let r be a natural number, $\mathscr{W}_1$ and $\mathscr{W}_2$ be disjoint unions of linear orders. We say that*

- *$\mathscr{G}_r(\mathscr{W}_1,\mathscr{W}_2)$ has a small disparity if there exists t such that $1 \leq t < 2^r - 2$, $q_t^{\mathscr{W}_1} \neq q_t^{\mathscr{W}_2}$, and $r \geq min\{q_t^{\mathscr{W}_1}, q_t^{\mathscr{W}_2}\} + \lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 2$.*
- *$\mathscr{G}_r(\mathscr{W}_1,\mathscr{W}_2)$ has a large disparity if there exists t such that $1 \leq t \leq 2^r - 1$, $q_{\geq t}^{\mathscr{W}_1} \neq q_{\geq t}^{\mathscr{W}_2}$, and $r \geq min\{q_{\geq t}^{\mathscr{W}_1}, q_{\geq t}^{\mathscr{W}_2}\} + \lfloor log(t) \rfloor + 1$.*

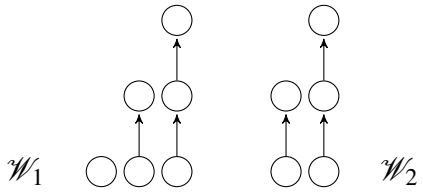
Lemma 5.2.1. *Let r be a natural number, $\mathscr{W}_1$ and $\mathscr{W}_2$ be two disjoint unions of linear orders. If $\mathscr{G}_r(\mathscr{W}_1,\mathscr{W}_2)$ has a small or large disparity, then the Spoiler has a winning strategy.*

Proof. Suppose that $q_t^{\mathscr{W}_1} > q_t^{\mathscr{W}_2}$, and $r \geq q_t^{\mathscr{W}_2} + \lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 2$. The Spoiler has the following winning strategy: first, she chooses elements $a_1, a_2, ..., a_{q_t^{\mathscr{W}_2}}$ from distinct linear orders of size t in $\mathscr{W}_1$. The Duplicator must choose elements $b_1, ..., b_{q_t^{\mathscr{W}_2}}$ in $\mathscr{W}_2$ from distinct linear orders of size t. Next, the Spoiler chooses an element a from a distinct linear order $\mathscr{L}(a)$ of size t in $\mathscr{W}_1$ such that $q^{\mathscr{L}(a)^{>a}} < 2^{r-1} - 1$ and $q^{\mathscr{L}(a)^{<a}} < 2^{r-1} - 1$. Then, the Duplicator must select an element b from a linear order $\mathscr{L}(b)$ such that $q^{\mathscr{L}(b)} \neq t$. Then, the Spoiler has a winning strategy in $\mathscr{G}_{\lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 1}(\mathscr{L}(a)^{<a}, \mathscr{L}(b)^{<b})$ by Theorem 5.1.1. Furthermore, the Spoiler has a winning strategy in $\mathscr{G}_{\lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 2}(\mathscr{L}(a), \mathscr{L}(b))$. Therefore, the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{W}_1,\mathscr{W}_2)$.

Now, suppose that $q_{\geq t}^{\mathscr{W}_1} > q_{\geq t}^{\mathscr{W}_2}$, and $r \geq q_{\geq t}^{\mathscr{W}_2} + \lfloor log(t) \rfloor + 1$. The Spoiler has the following winning strategy: first, she chooses elements $a_1, a_2, ..., a_{q_{\geq t}^{\mathscr{W}_2}}$ from distinct linear orders of size at least t in $\mathscr{W}_1$. The Duplicator must choose elements $b_1, ..., b_{q_{\geq t}^{\mathscr{W}_2}}$ in $\mathscr{W}_2$ from distinct linear orders of size at least t. Next, by Theorem 5.1.1, the Spoiler uses her winning strategy in $\mathscr{G}_{\lfloor log(t) \rfloor + 1}(\mathscr{L}_1, \mathscr{L}_2)$ such that $\mathscr{L}$ is a linear order in $\mathscr{W}_1$, $q^{\mathscr{L}_1} \geq t$, $q^{\mathscr{L}_2} < t$. Then, the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{W}_1,\mathscr{W}_2)$. □

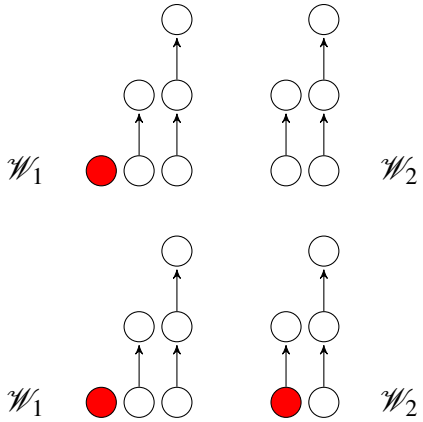
Example 5.2.1. *Let $\mathscr{W}_1$ and $\mathscr{W}_2$ be disjoint unions of linear orders such that $\mathscr{W}_1 = \mathscr{L}_1 \uplus \mathscr{L}_2 \uplus \mathscr{L}_3$, $\mathscr{W}_2 = \mathscr{L}_2 \uplus \mathscr{L}_3$, $q^{\mathscr{L}_1} = 1$, $q^{\mathscr{L}_2} = 2$, and $q^{\mathscr{L}_3} = 3$. Figure 24 shows $\mathscr{W}_1$ and $\mathscr{W}_2$. Then, $\mathscr{G}_2(\mathscr{W}_1,\mathscr{W}_2)$ has a small disparity because, for $t = 1$, $1 \leq t < 2^r - 2$, $q_t^{\mathscr{W}_1} = 1$, $q_t^{\mathscr{W}_2} = 0$, $q_t^{\mathscr{W}_1} \neq q_t^{\mathscr{W}_2}$, and $r \geq 0 + \lfloor log_2(0) \rfloor + 2$. Figures 25 and 26 depict the Spoiler's winning strategy.*

Figure 24 – EF game $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$



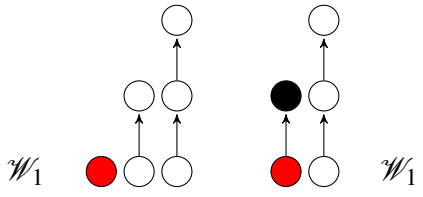
Source: Own elaboration

Figure 25 – First round of $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$



Source: Own elaboration

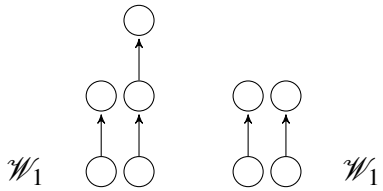
Figure 26 – Second round of $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$



Source: Own elaboration

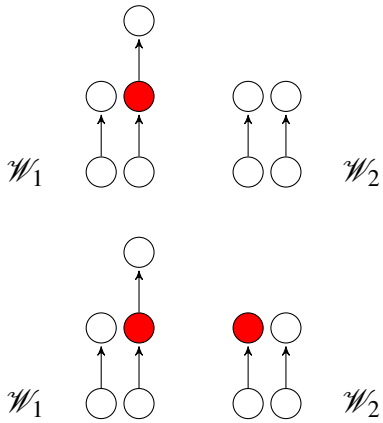
Example 5.2.2. *Let $\mathscr{W}_1 = \mathscr{L}_1 \uplus \mathscr{L}_2$ and $\mathscr{W}_2 = \mathscr{L}_1 \uplus \mathscr{L}_1$ such that $q^{\mathscr{L}_1} = 2$ and $q^{\mathscr{L}_2} = 3$. Figure 27 represents $\mathscr{W}_1$ and $\mathscr{W}_2$. Then, $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$ has a large disparity because, for $t = 3$, $t \leq 8$, $q^{\mathscr{W}_1}_{\geq t} = 1$ e $q^{\mathscr{W}_2}_{\geq t} = 0$, and $r \geq 0 + \lfloor log(t) \rfloor + 1$. Figures 28 and 29 depict the Spoiler's winning strategy.*

Figure 27 – EF game $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$



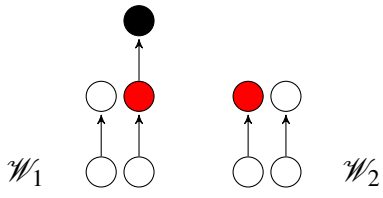
Source: Own elaboration

Figure 28 – First round of $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$



Source: Own elaboration

Figure 29 – Second round of $\mathscr{G}_2(\mathscr{W}_1, \mathscr{W}_2)$



Source: Own elaboration

In Lemma 5.2.1, different from the proof of Theorem 5.1.1, the Spoiler has a winning strategy which consists of choosing an element from a linear order with fewer elements. Then, first, we need the following lemma in order to guarantee a winning strategy for the Duplicator.

Lemma 5.2.2. *Let r be a natural number, $\mathscr{L}_1$ and $\mathscr{L}_2$ be linear orders.*

- *If $q^{\mathscr{L}_1} \geq 2^r - 1$ and the Spoiler chooses an element from $\mathscr{L}_1$ in her first round, then the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$ if $q^{\mathscr{L}_2} \geq 2^r - 1$.*
- *If $q^{\mathscr{L}_1} = 2^r - 2$ and the Spoiler chooses an element from $\mathscr{L}_1$ in her first round, then the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$ if $q^{\mathscr{L}_2} \geq 2^r - 2$.*
- *If $q^{\mathscr{L}_1} < 2^r - 2$ and the Spoiler chooses an element from $\mathscr{L}_1$ in her first round, then the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{L}_1, \mathscr{L}_2)$ if $q^{\mathscr{L}_2} = q^{\mathscr{L}_1}$.*

Proof. For the first and third part of the lemma, the Duplicator has a winning strategy as in Theorem 5.1.1. To prove the second part of the lemma, assume that $q^{\mathscr{L}_2} \geq 2^r - 2$. Let $a \in L_1$ be the element chosen by the Spoiler. Then, $q^{\mathscr{L}_1^{>a}} \geq 2^{r-1} - 1$ or $\mathscr{L}_1^{<a} \geq 2^{r-1} - 1$. Assume that $q^{\mathscr{L}_1^{>a}} \geq 2^{r-1} - 1$. The Duplicator chooses an element $b \in L_2$ such that $q^{\mathscr{L}_2^{>b}} \geq 2^{r-1} - 1$ and $q^{\mathscr{L}_2^{<b}} = q^{\mathscr{L}_2^{<a}}$. Then, the Duplicator has a winning strategy in $\mathscr{G}_{r-1}(\mathscr{L}_1^{>a}, \mathscr{L}_2^{>b})$. Therefore, the Duplicator has a winning strategy in r rounds. □

Theorem 5.2.1 (EF Games on DULO)**.** *Let r be a natural number, $\mathscr{W}_1$ and $\mathscr{W}_2$ be disjoint unions of linear orders. The Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{W}_1, \mathscr{W}_2)$ iff $\mathscr{G}_r(\mathscr{W}_1, \mathscr{W}_2)$ has a small or large disparity.*

Proof. One direction is Lemma 5.2.1. Conversely, assume that $\mathscr{G}_r(\mathscr{W}_1, \mathscr{W}_2)$ has neither small nor large disparity. Let $(a_1, b_1), ..., (a_k, b_k)$ be such that $a_i \in W_1$, $b_i \in W_2$ for $i \in \{1, ..., k\}$, and if the Spoiler has chosen a_i (b_i), then the Duplicator has selected b_i (a_i). We show that the Duplicator has a winning strategy by induction on k. Let $\mathscr{W}_1' = \mathscr{W}_1(a_1, ..., a_k)$ and $\mathscr{W}_2' = \mathscr{W}_2(b_1, ..., b_k)$. The following are our inductive hypotheses:

(i) For all $i, j \leq k$, $a_i < a_j$ iff $b_i < b_j$.

(ii) For all $i \leq k$, $q^{\mathscr{L}(a_i)} \geq 2^{r-i} - 1$, then $q^{\mathscr{L}(b_i)} \geq 2^{r-i} - 1$.

(iii) For all $i \leq k$, $q^{\mathscr{L}(a_i)} = 2^{r-i} - 2$, then $q^{\mathscr{L}(b_i)} \geq 2^{r-i} - 2$.

(iv) For all $i \leq k$, $q^{\mathscr{L}(a_i)} < 2^{r-i} - 2$, then $q^{\mathscr{L}(b_i)} = q^{\mathscr{L}(a_i)}$.

(v) $\mathscr{G}_{r-k}(\mathscr{W}_1', \mathscr{W}_2')$ has neither small nor large disparity.

If the Spoiler chooses an element $a_{k+1} \in W_1$ such that $a_i < a_{k+1}$, then the Duplicator responds by b_{k+1} such that $b_i < b_{k+1}$. The case where $a_{k+1} < a_i$ is analogous. Note that, by inductive hypotheses (ii), (iii), (iv), the Duplicator can select such an element. Now, assume $a_{k+1} \in W_1$ is in a linear order $\mathscr{L}(a_{k+1})$ different from $\mathscr{L}(a_i)$, for $i \leq k$. Suppose $q^{\mathscr{L}(a_{k+1})} \geq 2^{r-k} - 1$. By contradiction, assume that any linear order $\mathscr{L}$ in $\mathscr{W}_2'$ is such that $|L| < 2^{r-k} - 1$. Then, $\mathscr{G}_{r-k}(\mathscr{W}_1', \mathscr{W}_2')$ would have a large disparity as witnessed by $t = 2^{r-k} - 1$. Therefore, the

Duplicator can choose b_{k+1} in $\mathscr{W}_2$ such that $\mathscr{L}(b_{k+1})$ is different from $\mathscr{L}(b_i)$, for $i \leq k$, and $q^{\mathscr{L}(b_{k+1})} \geq 2^{r-k} - 1$. Now, assume that $q^{\mathscr{L}(a_{k+1})} = 2^{r-k} - 2$. We assume, for the sake of reaching a contradiction, that any linear order $\mathscr{L}$ in $\mathscr{W}_2'$ is such that $|L| < 2^{r-k} - 2$. Then, $\mathscr{G}_{r-k}(\mathscr{W}_1', \mathscr{W}_2')$ would have a large disparity as witnessed by $t = 2^{r-k} - 2$. Therefore, there exists b_{k+1} in $\mathscr{W}_2$ such that $\mathscr{L}(b_{k+1})$ is different from $\mathscr{L}(b_i)$, for $i \leq k$, and $q^{\mathscr{L}(b_{k+1})} \geq 2^{r-k} - 2$. Duplicator chooses b_{k+1}. Finally, suppose that $q^{\mathscr{L}(a_{k+1})} < 2^{r-k} - 2$. Then, b_{k+1} must exist in $\mathscr{W}_2$ such that $\mathscr{L}(b_{k+1})$ is different from $\mathscr{L}(b_i)$, for $i \leq k$, and $q^{\mathscr{L}(b_{k+1})} = q^{\mathscr{L}(a_{k+1})}$. Otherwise $\mathscr{G}_{r-k}(\mathscr{W}_1', \mathscr{W}_2')$ would have a small disparity as witnessed by $t = q^{\mathscr{L}(a_{k+1})}$. Then, the Duplicator chooses b_{k+1}. The cases where the Spoiler chooses an element from $\mathscr{W}_2$ are analogous.

Now, we check each of the inductive hypotheses on $(a_1, b_1), ..., (a_{k+1}, b_{k+1})$. Clearly, inductive hypothesis (i) holds. Now, assume that $q^{\mathscr{L}(a_{k+1})} \geq 2^{r-k-1} - 1$. If $q^{\mathscr{L}(a_{k+1})} \geq 2^{r-k} - 1$, then the strategy we describe ensures that $q^{\mathscr{L}(b_{k+1})} \geq 2^{r-k} - 1$. If $q^{\mathscr{L}(a_{k+1})} = 2^{r-k} - 2$, then the strategy ensures that $q^{\mathscr{L}(b_{k+1})} \geq 2^{r-k} - 2$. Finally, if $q^{\mathscr{L}(a_{k+1})} < 2^{r-k} - 2$, then $q^{\mathscr{L}(b_{k+1})} = q^{\mathscr{L}(a_{k+1})}$. Clearly, in all cases $q^{\mathscr{L}(b_{k+1})} \geq 2^{r-k-1} - 1$. Now, assume that $q^{\mathscr{L}(a_{k+1})} = 2^{r-k-1} - 2$. Obviously, $q^{\mathscr{L}(a_{k+1})} < 2^{r-k} - 2$. Then, $q^{\mathscr{L}(b_{k+1})} = q^{\mathscr{L}(a_{k+1})}$, and $q^{\mathscr{L}(b_{k+1})} \geq 2^{r-k-1} - 2$. At last, assume $q^{\mathscr{L}(a_{k+1})} < 2^{r-k-1} - 2$. Then, $q^{\mathscr{L}(a_{k+1})} < 2^{r-k} - 2$ and $q^{\mathscr{L}(b_{k+1})} = q^{\mathscr{L}(a_{k+1})}$. Therefore, $q^{\mathscr{L}(b_{k+1})} < 2^{r-k-1} - 2$.

Now, it remains to prove that (v) holds. Consider $\mathscr{W}_1'' = \mathscr{W}_1(a_1, ..., a_{k+1})$ and $\mathscr{W}_2'' = \mathscr{W}_2(b_1, ..., b_{k+1})$. By contradiction, assume that $\mathscr{G}_{r-k-1}(\mathscr{W}_1'', \mathscr{W}_2'')$ has a small disparity. Assume $t < 2^{r-k-1} - 2$ such that $q_t^{\mathscr{W}_1''} > q_t^{\mathscr{W}_2''}$ and $r - k - 1 \geq min\{q_t^{\mathscr{W}_1''}, q_t^{\mathscr{W}_2''}\} + \lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 2$. Then, $t < 2^{r-k} - 2$. If $q_t^{\mathscr{W}_1'} = q_t^{\mathscr{W}_2'}$, then $q^{\mathscr{L}}(b_k) = t$ and, as $t < 2^{r-k} - 2$, $q^{\mathscr{L}}(a_k) = t$. Therefore, $q_t^{\mathscr{W}_1'} \neq q_t^{\mathscr{W}_2'}$. Thus, there exists $t < 2^{r-k}$ such that $q_t^{\mathscr{W}_1'} \neq q_t^{\mathscr{W}_2'}$, and $r - k \geq min\{q_t^{\mathscr{W}_1'}, q_t^{\mathscr{W}_2'}\} \lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 2$ because $min\{q_t^{\mathscr{W}_1'}, q_t^{\mathscr{W}_2'}\} \leq min\{q_t^{\mathscr{W}_1''}, q_t^{\mathscr{W}_2''}\} + 1$. Then, $\mathscr{G}_{r-k-1}(\mathscr{W}_1'', \mathscr{W}_2'')$ do not have a small disparity. By an analogous way, $\mathscr{G}_{r-k-1}(\mathscr{W}_1'', \mathscr{W}_2'')$ do not have a large disparity. Finally, the Duplicator has a winning strategy because (i) holds at each round. □

Given disjoint unions of linear orders $\mathscr{W}_1$ and $\mathscr{W}_2$, by Theorem 5.2.1, the minimum number of rounds such that the Spoiler has a winning strategy can be found as follows.

$$MinRound(\mathscr{W}_1, \mathscr{W}_2) = \begin{array}{l} min(min\{min(q_t^{\mathscr{W}_1}, q_t^{\mathscr{W}_2}) + \lfloor log(\lceil \frac{t+1}{2} \rceil) \rfloor + 2 \mid q_t^{\mathscr{E}_1} \neq q_t^{\mathscr{E}_2}\}, \\ min\{min(q_{\geq t}^{\mathscr{E}_1}, q_{\geq t}^{\mathscr{E}_2}) + \lfloor log(t) \rfloor + 1 \mid q_{\geq t}^{\mathscr{E}_1} \neq q_{\geq t}^{\mathscr{E}_2}\}). \end{array}$$

As in the case of equivalence structures, it takes time $O((|W_1| + |W_2|)^2)$ to compute $q_t^{\mathscr{W}_1}$, $q_t^{\mathscr{W}_2}$, $q_{\geq t}^{\mathscr{W}_1}$, $q_{\geq t}^{\mathscr{W}_2}$, for $t \in \{1, ..., 2^r - 1\}$. The rest of the computation of $MinRound(\mathscr{W}_1, \mathscr{W}_2)$ takes linear time. Therefore, the overall computation of $MinRound(\mathscr{W}_1, \mathscr{W}_2)$ takes time $O((|W_1| + |W_2|)^2)$.

In the following chapter, we use the above results to solve the problem of synthesis of minimum quantifier rank sentences for disjoint unions of linear orders. We also show results on the synthesis problem for the other classes of structures we are considering in this work.

6 SYNTHESIS OF MINIMUM QUANTIFIER RANK SENTENCES

An algorithm to deal with the problem of finding a first-order sentence of minimum quantifier rank that distinguishes two sets of structures over an arbitrary vocabulary is presented in (KAISER, 2012). In this chapter, we study a variation of the problem introduced in (KAISER, 2012) when the class of structures is fixed. We call this problem the Synthesis Problem. We consider the following classes of structures: monadic structures, equivalence structures, disjoint unions of linear orders, and strings represented by finite structures with a built-in successor relation and a finite number of pairwise disjoint unary predicates. First, we introduce distinguishability sentences based on necessary and sufficient conditions for a winning strategy in an Ehrenfeucht–Fraïssé game. We also show that any first-order sentence is equivalent to a Boolean combination of distinguishability sentences. Using these results, we define a polynomial time algorithm for the Synthesis Problem. The results in this chapter were published in (ROCHA *et al.*, 2019) and (ROCHA *et al.*, 2018a).

6.1 Synthesis for a Fixed Class of Structures

For a fixed class of structures $\mathscr{C}$, a sample $S = (P,N)$ consists of two finite sets $P,N \subseteq \mathscr{C}$ such that for each $\mathscr{A} \in P$, $\mathscr{B} \in N$, $\mathscr{A}$ and $\mathscr{B}$ are not isomorphic. Intuitively, P contains positively classified structures and N contains negatively classified structures. A sentence φ is consistent with a sample $S = (P,N)$ if $P \subseteq MOD(\varphi)$ and $N \cap MOD(\varphi) = \emptyset$. Therefore, φ 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 sizes of all structures in the sample. In what follows, we formally define the synthesis problem.

Definition 6.1.1 (Synthesis Problem)**.** *Let $\mathscr{C}$ be a fixed class of structures. Given a sample S, the problem consists of finding a first-order sentence φ of minimum quantifier rank that is consistent with S.*

When the class of structures is not fixed, the decision version of the synthesis problem, i.e., given r, determining whether there exists a first-order sentence of quantifier rank r and consistent with the sample, is hard. This problem is *PSPACE*-complete even when P and N are singleton sets (PEZZOLI, 1998).

It is well known that every finite relational structure can be characterized in first-

order logic up to isomorphism. In other words, for every finite structure $\mathscr{A}$, there is a first-order sentence $\varphi_{\mathscr{A}}$ such that for all structures $\mathscr{B}$ we have $\mathscr{B} \models \varphi_{\mathscr{A}}$ if and only if $\mathscr{A}$ and $\mathscr{B}$ are isomorphic (Proposition 2.1.1 from (EBBINGHAUS; FLUM, 1995)). Since we are considering a fixed vocabulary and samples are finite sets of finite relational structures, one can easily build in polynomial time a first-order sentence consistent with a given sample. For example, let $\tau = \{P,R\}$ be a fixed vocabulary such that P has arity one, and R has arity two. Clearly, the size of $\varphi_{\mathscr{A}}$ is $O(n^2)$. Also, it takes polynomial time to check whether $\mathscr{A} \models \psi$ when ψ is atomic. Therefore, we build $\varphi_{\mathscr{A}}$ in polynomial time. Unfortunately, the quantifier rank of $\varphi_{\mathscr{A}}$ is the number of elements in the domain of $\mathscr{A}$ plus one. Therefore, we can not use these formulas in a solution to the synthesis problem.

It is easy to build a minimum quantifier rank first-order sentence that consists of a disjunction of Hintikka formulas and that is consistent with a given sample. Let $P = \{\mathscr{M}_1\}$, $N = \{\mathscr{M}_2, \mathscr{M}_3\}$, $r = max\{MinRound(\mathscr{M}_1, \mathscr{M}_2), MinRound(\mathscr{M}_1, \mathscr{M}_3)\}$, and $S = (P,N)$, for example. For the classes of structures we are considering, r can be found in polynomial time in the size of the sample. Then, the sentence $\varphi^r_{\mathscr{M}_1}$ is a first-order sentence of minimum quantifier rank that is consistent with S. Unfortunately, the size of $\varphi^r_{\mathscr{M}_1}$ is exponential in the size of S. Therefore, $\varphi^r_{\mathscr{M}_1}$ can not be built in polynomial time in the size of the sample. This motivates the introduction of the distinguishability sentences in what follows.

6.2 Distinguishability Sentences

In this section, we define the distinguishability sentences for structures $\mathscr{A}$, $\mathscr{B}$ in a class of structures $\mathscr{C}$, and a natural number r. The distinguishability sentences hold in $\mathscr{A}$, do not hold in $\mathscr{B}$, and they have quantifier rank at most r. We define the set of distinguishability sentences $\Phi^r_{\mathscr{A},\mathscr{B}}$ in a way such that the Spoiler has a winning strategy in $\mathscr{G}_r(\mathscr{A},\mathscr{B})$ if and only if there exists $\varphi \in \Phi^r_{\mathscr{A},\mathscr{B}}$. This result follows from Theorem 2.3.1. The first step is to show that the conditions characterizing winning strategies for the Spoiler can be expressed by first-order sentences of size polynomial in the size of the structures. This result is important to guarantee that our algorithm runs in polynomial time.

6.2.1 Distinguishability Sentences for MS

First, we define $|P_i| \geq n$, for $i \in \{1,...,k+1\}$, as a sentence describing that the number of elements in P_i is at least n:

$$|P_i| \geq n := \exists x_1 ... \exists x_n (\bigwedge_{l \neq j}^{n} x_l \neq x_j \wedge \bigwedge_{l=1}^{n} P_i(x_l)).$$

Clearly, $qr(|P_i| \geq n) = n$, the size of $|P_i| \geq n$ is $O(n^2)$, and $\mathscr{A} \models |P_i| \geq n$ iff $|P_i^{\mathscr{A}}| \geq n$. We also define abbreviations $|P_i| \leq n := \neg |P_i| \geq n+1$ and $|P_i| = n := |P_i| \geq n \wedge |P_i| \leq n$. Now, we can define the distinguishability sentences for monadic structures.

Definition 6.2.1 (Distinguishability Sentences for MS)**.** *Let $\mathscr{M}_1$, $\mathscr{M}_2$ be monadic structures. Let r be a natural number.*

$$\Phi^r_{\mathscr{M}_1,\mathscr{M}_2} := \begin{array}{l} \{|P_i| < m \mid 1 \leq i \leq k+1, |P_i^{\mathscr{M}_1}| < |P_i^{\mathscr{M}_2}|, |P_i^{\mathscr{M}_1}| + 1 \leq m \leq min(r, |P_i^{\mathscr{M}_2}|)\} \cup \\ \{|P_i| \geq m \mid 1 \leq i \leq k+1, |P_i^{\mathscr{M}_1}| > |P_i^{\mathscr{M}_2}|, |P_i^{\mathscr{M}_2}| + 1 \leq m \leq min(r, |P_i^{\mathscr{M}_1}|)\}. \end{array}$$

Given $\mathscr{M}_1$, $\mathscr{M}_2$, and r, the size of a sentence $\varphi \in \Phi^r_{\mathscr{M}_1,\mathscr{M}_2}$ is $O((|M_1| + |M_2|)^2)$, and $|\Phi^r_{\mathscr{M}_1,\mathscr{M}_2}|$ is $O(k(|M_1| + |M_2|))$. Now, we give an example of the distinguishability sentences for MS. Then, we show results ensuring adequate properties of the distinguishability sentences.

Example 6.2.1. *Let $\mathscr{M}_1 = \langle M_1, P_1^{\mathscr{M}_1} \rangle$ and $\mathscr{M}_2 = \langle M_2, P_1^{\mathscr{M}_2} \rangle$ such that $|P_1^{\mathscr{M}_1}| = 2$, $|P_2^{\mathscr{M}_1}| = 3$, $|P_1^{\mathscr{M}_2}| = 2$, $|P_2^{\mathscr{M}_1}| = 2$, and $r = 4$. Then, $|P_2| \geq 3 \in \Phi^r_{\mathscr{M}_1,\mathscr{M}_2}$ because $|P_2^{\mathscr{M}_1}| > |P_2^{\mathscr{M}_2}|$ and, for $m = 3$, $|P_2^{\mathscr{M}_2}| + 1 \leq m \leq min(r, |P_2^{\mathscr{M}_1}|)$. Also, observe that $|P_2| \geq 4 \notin \Phi^r_{\mathscr{M}_1,\mathscr{M}_2}$ since, for $m = 4$, $m > min(r, |P_2^{\mathscr{M}_1}|)$. Finally, $|P_1| \geq 2 \notin \Phi^r_{\mathscr{M}_1,\mathscr{M}_2}$ because $|P_1^{\mathscr{M}_1}| = |P_1^{\mathscr{M}_2}|$.*

Lemma 6.2.1. *Let $\mathscr{M}_1$, $\mathscr{M}_2$ be structures in* MS, and r be a natural number. Let $\varphi \in \Phi^r_{\mathscr{M}_1,\mathscr{M}_2}$. Then, $\mathscr{M}_1 \models \varphi$ and $\mathscr{M}_2 \not\models \varphi$.

Proof. Suppose $\varphi = |P_i| < m$. Then, $|P_i^{\mathscr{M}_1}| < |P_i^{\mathscr{M}_2}|$ and $|P_i^{\mathscr{M}_1}| + 1 \leq m \leq min(r, |P_i^{\mathscr{M}_2}|)$. Therefore, $\mathscr{M}_1 \models \varphi$ because $|P_i^{\mathscr{M}_1}| < m$. Clearly, $m \leq |P_i^{\mathscr{M}_2}|$. Then, $\mathscr{M}_1 \not\models \varphi$. The case in which $\varphi = |P_i| \geq m$ is similar. □

Lemma 6.2.2. *Let $\mathscr{M}_1$, $\mathscr{M}_2$ be structures in* MS, and r be a natural number. Let $\varphi \in \Phi^r_{\mathscr{M}_1,\mathscr{M}_2}$. Then, $qr(\varphi) \leq r$.

Proof. Let $\varphi = |P_i| \trianglelefteq m$ where $\trianglelefteq \in \{<, \geq\}$. Hence, $qr(\varphi) = m$. As $m \leq r$, then $qr(\varphi) \leq r$. □

Now, we show that, over MS, any first-order sentence is equivalent to a Boolean combination of distinguishability sentences. First, we define formulas equivalent to Hintikka formulas over MS.

Lemma 6.2.3. $\models \varphi^r_{\mathscr{M}} \leftrightarrow \bigwedge_{i=1}^{k+1} \varphi^{r,i}_{\mathscr{M}}$ *such that*

$$\varphi^{r,i}_{\mathscr{M}} := \begin{cases} |P_i| = |P_i^{\mathscr{M}}|, & \text{if } |P_i^{\mathscr{M}}| < r \\ |P_i| \geq r, & \text{otherwise.} \end{cases}$$

Proof. Let $\mathscr{M}' \models \varphi^r_{\mathscr{M}}$. Therefore, the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{M}, \mathscr{M}')$. Then, for all i, $|P_i^{\mathscr{M}}| = |P_i^{\mathscr{M}'}|$ or ($|P_i^{\mathscr{M}}| \geq r$ and $|P_i^{\mathscr{M}'}| \geq r$). Let i such that $|P_i^{\mathscr{M}}| < r$. Then, $|P_i^{\mathscr{M}}| = |P_i^{\mathscr{M}'}|$. Therefore, $\mathscr{M}' \models \varphi^{r,i}_{\mathscr{M}}$. Let i such that $|P_i^{\mathscr{M}}| \geq r$. Then, $|P_i^{\mathscr{M}'}| \geq r$. Therefore, $\mathscr{M}' \models \varphi^{r,i}_{\mathscr{M}}$. Finally, $\mathscr{M}' \models \bigwedge_{i=1}^{k+1} \varphi^{r,i}_{\mathscr{M}}$. Conversely, let $\mathscr{M}' \models \bigwedge_{i=1}^{k+1} \varphi^{r,i}_{\mathscr{M}}$. Then, $\mathscr{M}' \models \varphi^{r,i}_{\mathscr{M}}$, for all i. If $\varphi^{r,i}_{\mathscr{M}} = (|P_i| = |P_i^{\mathscr{M}}|)$, then $|P_i^{\mathscr{M}}| = |P_i^{\mathscr{M}'}|$. If $\varphi^{r,i}_{\mathscr{M}} = |P_i| \geq r$, then ($|P_i^{\mathscr{M}}| \geq r$ and $|P_i^{\mathscr{M}'}| \geq r$). Then, for all i, $|P_i^{\mathscr{M}}| = |P_i^{\mathscr{M}'}|$ or ($|P_i^{\mathscr{M}}| \geq r$ and $|P_i^{\mathscr{M}'}| \geq r$). Therefore, $\mathscr{M}' \models \varphi^r_{\mathscr{M}}$. □

Now, we need the following lemmas.

Lemma 6.2.4. *Let r be a natural number and $\mathscr{M}$ be a structure in* MS. There exists a set of monadic structures V_i such that $\varphi^{r,i}_{\mathscr{M}}$ is equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{M}' \in V_i} \Phi^r_{\mathscr{M},\mathscr{M}'}$.

Proof. If $\varphi^{r,i}_{\mathscr{M}} = |P_i| \geq r$, then $|P_i^{\mathscr{M}}| \geq r$. Let $V_i = \{\mathscr{M}'\}$ such that $|P_i^{\mathscr{M}'}| = r - 1$. Then, $|P_i| \geq r \in \Phi^r_{\mathscr{M},\mathscr{M}'}$ because $|P_i^{\mathscr{M}}| > |P_i^{\mathscr{M}'}|$ and $m = r$. If $\varphi^{r,i}_{\mathscr{M}} = (|P_i| = |P_i^{\mathscr{M}}|)$, then $|P_i^{\mathscr{M}}| < r$. Let $V_i = \{\mathscr{M}_1, \mathscr{M}_2\}$ such that $|P_i^{\mathscr{M}_1}| = |P_i^{\mathscr{M}}| - 1$ and $|P_i^{\mathscr{M}_2}| = |P_i^{\mathscr{M}}| + 1$. Then, $|P_i| \geq |P_i^{\mathscr{M}}| \in \Phi^r_{\mathscr{M},\mathscr{M}_1}$ because $|P_i^{\mathscr{M}}| > |P_i^{\mathscr{M}_1}|$ and $m = |P_i^{\mathscr{M}}|$. Also, $|P_i| < |P_i^{\mathscr{M}}| + 1 \in \Phi^r_{\mathscr{M},\mathscr{M}_2}$ because $|P_i^{\mathscr{M}}| < |P_i^{\mathscr{M}_2}|$ and $m = |P_i^{\mathscr{M}}| + 1$. Therefore, $\varphi^{r,i}_{\mathscr{M}}$ is equivalent to $|P_i| \geq |P_i^{\mathscr{M}}| \wedge |P_i| < |P_i^{\mathscr{M}}| + 1$. □

Lemma 6.2.5. *Let r be a natural number and $\mathscr{M}$ be a structure in* MS. There exists a set of monadic structures V such that $\varphi^r_{\mathscr{M}}$ is a Boolean combination of sentences in $\bigcup_{\mathscr{M}' \in V} \Phi^r_{\mathscr{M},\mathscr{M}'}$.

Proof. By Lemma 6.2.3, $\models \varphi^r_{\mathscr{M}} \leftrightarrow \bigwedge_{i=1}^{k+1} \varphi^{r,i}_{\mathscr{W}}$. Let $V = \bigcup_{i=1}^{k+1} V_i$ such that V_i is as in Lemma 6.2.4. It follows that, $\varphi^r_{\mathscr{M}}$ is equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{M}' \in V} \Phi^r_{\mathscr{M},\mathscr{M}'}$. □

By Theorem 2.3.2, any first-order sentence is a disjunction of Hintikka formulas. Thus, the following result states that, over MS, any first-order sentence is equivalent to a Boolean combination of distinguishability sentences.

Theorem 6.2.1. *Let φ be a first-order sentence over* MS. Then, there exists two sets U, V of monadic structures such that φ is equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{M} \in U, \mathscr{M}' \in V} \Phi^r_{\mathscr{M},\mathscr{M}'}$.

Proof. Let r such that $qr(\varphi) = r$. From Theorem 2.3.2 it follows that $\models \varphi \leftrightarrow \varphi^r_{\mathscr{M}_1} \vee ... \varphi^r_{\mathscr{M}_s}$. Let $U = \{\mathscr{M}_1, ..., \mathscr{M}_s\}$. In accord to Lemma 6.2.5, let V_j be such that $\varphi^r_{\mathscr{M}_j}$ is equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{M}' \in V_j} \Phi^r_{\mathscr{M}_j,\mathscr{M}'}$. Therefore, φ is equivalent to a Boolean combination of sentences in $\bigcup_{j=1}^{s}(\bigcup_{\mathscr{M}' \in V_j} \Phi^r_{\mathscr{M}_j,\mathscr{M}'})$ and $\bigcup_{j=1}^{s}(\bigcup_{\mathscr{M}' \in V_j} \Phi^r_{\mathscr{M}_j,\mathscr{M}'}) \subseteq \bigcup_{\mathscr{M} \in U, \mathscr{M}' \in V} \Phi^r_{\mathscr{M},\mathscr{M}'}$. □

6.2.2 Distinguishability Sentences for ES

Now, we deal with the distinguishability sentences for equivalence structures. First, we define the following formulas:

$$\varphi_{q_{\geq t} \geq p} = \exists x_1 ... \exists x_p (\bigwedge_{l \neq j} \neg E(x_l, x_j) \wedge \bigwedge_{k=1}^{p} \exists y_2 ... \exists y_t (\bigwedge_{j=2}^{t} y_j \neq x_k \wedge \bigwedge_{l<j} y_l \neq y_j \wedge \bigwedge_{j=2}^{t} E(y_j, x_k))).$$

$$\varphi_{q_t \geq p} := \begin{array}{l} \exists x_1 ... \exists x_p (\bigwedge_{l<j} \neg E(x_l, x_j) \wedge \bigwedge_{k=1}^{p} \exists y_2 ... \exists y_t (\bigwedge_{j=2}^{t} y_j \neq x_k \wedge \bigwedge_{l \neq j} y_l \neq y_j \wedge \\ \bigwedge_{i=2}^{t} E(y_i, x_k) \wedge \forall z (E(z, x_k) \rightarrow (z = x_k \vee \bigvee_{j=2}^{t} z = y_j)))). \end{array}$$

Sentences of the form $\varphi_{q_{\geq t} \geq p}$ hold on equivalence structures such that the number of equivalence classes of size at least t is at least p. Each variable x_k represents an element in a distinct equivalence class. Variables y_i guarantee that a class has at least t elements. Formulas $\varphi_{q_t \geq p}$ are true when the number of equivalence classes of size t is at least p. Variable z forces that any element from an equivalence class represented by an element x_k is x_k or one of y_i for $i \in \{2, ..., t\}$. We also define $\varphi_{q_t < p} := \neg \varphi_{q_t \geq p}$ and $\varphi_{q_{\geq t} < p} := \neg \varphi_{q_{\geq t} \geq p}$. Clearly, $qr(\varphi_{q_{\geq t} \geq p}) = t + p - 1$ and $qr(\varphi_{q_t \geq p}) = t + p$. Also, the size of $\varphi_{q_{\geq t} \geq p}$ and $\varphi_{q_t \geq p}$ is $O((p+t)^3)$. Next, we define the distinguishability sentences for equivalence structures.

Definition 6.2.2 (Distinguishability Sentences for ES)**.** *Let $\mathscr{E}_1$, $\mathscr{E}_2$ be equivalence structures and r be a natural number.*

$$\Phi^r_{\mathscr{E}_1,\mathscr{E}_2} := \begin{array}{l} \{\varphi_{q_t < m} \mid 1 \leq t \leq r, q_t^{\mathscr{E}_1} < q_t^{\mathscr{E}_2}, q_t^{\mathscr{E}_1} + 1 \leq m \leq min(r-t, q_t^{\mathscr{E}_2})\} \cup \\ \{\varphi_{q_t \geq m} \mid 1 \leq t \leq r, q_t^{\mathscr{E}_1} > q_t^{\mathscr{E}_2}, q_t^{\mathscr{E}_2} + 1 \leq m \leq min(r-t, q_t^{\mathscr{E}_1})\} \cup \\ \{\varphi_{q_{\geq t} < m} \mid 1 \leq t \leq r, q_{\geq t}^{\mathscr{E}_1} < q_{\geq t}^{\mathscr{E}_2}, q_{\geq t}^{\mathscr{E}_1} + 1 \leq m \leq min(r-t+1, q_{\geq t}^{\mathscr{E}_2})\} \cup \\ \{\varphi_{\geq q_t \geq m} \mid 1 \leq t \leq r, q_{\geq t}^{\mathscr{E}_1} > q_{\geq t}^{\mathscr{E}_2}, q_{\geq t}^{\mathscr{E}_2} + 1 \leq m \leq min(r-t+1, q_{\geq t}^{\mathscr{E}_1})\}. \end{array}$$

For equivalence structures $\mathscr{E}_1$, $\mathscr{E}_2$, and a natural number r, the size of a sentence $\varphi \in \Phi^r_{\mathscr{E}_1,\mathscr{E}_2}$ is $O((|E_1| + |E_2|)^3)$. Furthermore, $|\Phi^r_{\mathscr{E}_1,\mathscr{E}_2}|$ is $O((|E_1| + |E_2|)^2)$. In what follows, we give examples of the distinguishability sentences for equivalence structures.

Example 6.2.2. *Let $\mathscr{E}_1$ and $\mathscr{E}_2$ be equivalence structures such that $q_2^{\mathscr{E}_1} = 3$ and $q_2^{\mathscr{E}_2} = 2$, and $r = 5$. Then, $\varphi_{q_2 \geq 3} \in \Phi^r_{\mathscr{E}_1,\mathscr{E}_2}$ because $q_2^{\mathscr{E}_1} > q_2^{\mathscr{E}_2}$ and, for $m = 3$, $q_2^{\mathscr{E}_2} + 1 \leq m \leq min(r-2, q_2^{\mathscr{E}_1})$.*

Now, we show results ensuring that the distinguishability sentences for equivalence structures hold in the adequate equivalence structures and have quantifier rank at most r.

Lemma 6.2.6. *Let $\mathscr{E}_1, \mathscr{E}_2$ be equivalence structures, and r be a natural number. Let $\varphi \in \Phi^r_{\mathscr{E}_1,\mathscr{E}_1}$. Then, $\mathscr{E}_1 \models \varphi$ and $\mathscr{E}_2 \not\models \varphi$.*

Proof. Suppose $\varphi = \varphi_{q_t < m}$. Then, $q_t^{\mathscr{E}_1} < q_t^{\mathscr{E}_2}$ and $q_t^{\mathscr{E}_1} + 1 \leq m \leq min(r-t, q_t^{\mathscr{E}_2})$. Then, $\mathscr{E}_1 \models \varphi$ because $q_t^{\mathscr{E}_1} < m$. Also, as $m \leq q_t^{\mathscr{E}_2}$, $\mathscr{E}_2 \not\models \varphi$. The other cases are analogous. □

Lemma 6.2.7. *Let $\mathscr{E}_1, \mathscr{E}_2$ be equivalence structures, and r be a natural number. Let $\varphi \in \Phi^r_{\mathscr{E}_1,\mathscr{E}_2}$. Then, $qr(\varphi) \leq r$.*

Proof. If $\varphi = \varphi_{q_t \triangleright m}$ where $\triangleright \in \{<, \geq\}$, then $qr(\varphi) = t + m$. As $m \leq r - t$, then $qr(\varphi) \leq t + r - t = r$. If $\varphi = \varphi_{q_{\geq t} \triangleright m}$, then $qr(\varphi) = t + m - 1$. Therefore, as $m \leq r - t + 1$, $qr(\varphi) \leq t + r - t + 1 - 1 = r$. □

Now, we show that, over ES, any first-order sentence is equivalent to a Boolean combination of distinguishability sentences. First, we need the following lemmas.

Lemma 6.2.8. $\models \varphi^r_{\mathscr{E}} \leftrightarrow (\bigwedge_{t=1}^r \varphi^{r,q_t}_{\mathscr{E}} \wedge \bigwedge_{t=1}^r \varphi^{r,q_{\geq t}}_{\mathscr{E}})$ *such that*

$$\varphi^{r,q_t}_{\mathscr{E}} := \begin{cases} \varphi_{q_t = q_t^{\mathscr{E}}}, & \text{if } q_t^{\mathscr{E}} + t + 1 \leq r \\ \varphi_{q_t > r-t-1}, & \text{otherwise.} \end{cases}$$

$$\varphi^{r,q_{\geq t}}_{\mathscr{E}} := \begin{cases} \varphi_{q_{\geq t} = q_{\geq t}^{\mathscr{E}}}, & \text{if } q_{\geq t}^{\mathscr{E}} + t \leq r \\ \varphi_{q_{\geq t} > r-t}, & \text{otherwise.} \end{cases}$$

Proof. Let $\mathscr{E}' \models \varphi^r_{\mathscr{E}}$. Then, the Duplicator has a winning strategy in $\mathscr{G}_r(\mathscr{E}, \mathscr{E}')$. Then, for all t, $q_t^{\mathscr{E}} = q_t^{\mathscr{E}'}$ or $r < min\{q_t^{\mathscr{E}}, q_t^{\mathscr{E}'}\} + t + 1$, and for all t, $q_{\geq t}^{\mathscr{E}} = q_{\geq t}^{\mathscr{E}'}$ or $r < min\{q_{\geq t}^{\mathscr{E}}, q_{\geq t}^{\mathscr{E}'}\} + t$. First, let t such that $r < q_t^{\mathscr{E}} + t + 1$. Then, $r < q_t^{\mathscr{E}'} + t + 1$ because $q_t^{\mathscr{E}} = q_t^{\mathscr{E}'}$ or $r < min\{q_t^{\mathscr{E}}, q_t^{\mathscr{E}'}\} + t + 1$. Besides, $\varphi^{r,q_t}_{\mathscr{E}} = \varphi_{q_t > r-t-1}$. Therefore, $\mathscr{E}' \models \varphi^{r,q_t}_{\mathscr{E}}$. If $r \geq q_t^{\mathscr{E}} + t + 1$, then $q_t^{\mathscr{E}} = q_t^{\mathscr{E}'}$. Therefore, $\mathscr{E}' \models \varphi_{q_t = q_t^{\mathscr{E}}}$. The case for $q_{\geq t}^{\mathscr{E}}$ is analogous. Then, $\mathscr{E}' \models (\bigwedge_{t=1}^r \varphi^{r,q_t}_{\mathscr{E}} \wedge \bigwedge_{t=1}^r \varphi^{r,q_{\geq t}}_{\mathscr{E}})$. Conversely, suppose that $\mathscr{E}' \models (\bigwedge_{t=1}^r \varphi^{r,q_t}_{\mathscr{E}} \wedge \bigwedge_{t=1}^r \varphi^{r,q_{\geq t}}_{\mathscr{E}})$. Let t such that $\varphi^{r,q_t}_{\mathscr{E}} = \varphi_{q_t > r-t-1}$. Then, $r < q_t^{\mathscr{E}} + t + 1$ and $r < q_t^{\mathscr{E}'} + t + 1$. Therefore, $r < min\{q_t^{\mathscr{E}}, q_t^{\mathscr{E}'}\} + t + 1$. Let t such that $\varphi^{r,q_t}_{\mathscr{E}} = \varphi_{q_t = q_t^{\mathscr{E}}}$. Clearly, $q_t^{\mathscr{E}} = q_t^{\mathscr{E}'}$. The case for $\varphi^{r,q_{\geq t}}_{\mathscr{E}}$ is analogous. Then, for all t, $q_t^{\mathscr{E}} = q_t^{\mathscr{E}'}$ or $r < min\{q_t^{\mathscr{E}}, q_t^{\mathscr{E}'}\} + t + 1$, and for all t, $q_{\geq t}^{\mathscr{E}} = q_{\geq t}^{\mathscr{E}'}$ or $r < min\{q_{\geq t}^{\mathscr{E}}, q_{\geq t}^{\mathscr{E}'}\} + t$. Therefore, $\mathscr{E}' \models \varphi^r_{\mathscr{E}}$. □

Lemma 6.2.9. *Let r be a natural number, and $\mathscr{E}$ be an equivalence structure. There exists sets of equivalence structures V_t, $V_{\geq t}$ such that $\varphi^{r,q_t}_{\mathscr{E}}$, $\varphi^{r,q_{\geq t}}_{\mathscr{E}}$ are equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{E}'\in V_t}\Phi^r_{\mathscr{E},\mathscr{E}'}$ and $\bigcup_{\mathscr{E}'\in V_{\geq t}}\Phi^r_{\mathscr{E},\mathscr{E}'}$, respectively.*

Proof. If $\varphi^{r,q_t}_{\mathscr{E}} = \varphi_{q_t>r-t-1}$, then $q^{\mathscr{E}}_t \geq r-t$. Let $V_t = \{\mathscr{E}'\}$ such that $q^{\mathscr{E}'}_t = r-t-1$. Then, $\varphi_{q_t\geq r-t} \in \Phi^r_{\mathscr{E},\mathscr{E}'}$ because $r-t-1+1 \leq r-t \leq min(r-t, q^{\mathscr{E}}_t)$. If $\varphi^{r,q_t}_{\mathscr{E}} = \varphi_{q_t=q^{\mathscr{E}}_t}$, then $q^{\mathscr{E}}_t < r-t$. $V_t = \{\mathscr{E}_1, \mathscr{E}_2\}$ such that $q^{\mathscr{E}_1}_t = q^{\mathscr{E}}_t - 1$ and $q^{\mathscr{E}_2}_t = q^{\mathscr{E}}_t + 1$. Then, $\varphi_{q_t\geq q^{\mathscr{E}}_t} \in \Phi^r_{\mathscr{E},\mathscr{E}_1}$ because, for $m = q^{\mathscr{E}}_t$, $q^{\mathscr{E}_1}_t \leq m \leq min(r-t, q^{\mathscr{E}}_t)$ and $q^{\mathscr{E}}_t < r-t$. Also, $\varphi_{q_t<q^{\mathscr{E}}_t+1} \in \Phi^r_{\mathscr{E},\mathscr{E}_2}$ as long as $q^{\mathscr{E}}_t + 1 \leq min(r-t, q^{\mathscr{B}}_t)$ and $q^{\mathscr{E}}_t + 1 \geq r-t$. For $\varphi^{r,q_{\geq t}}_{\mathscr{E}}$, we define $V_{\geq t}$ in an analogous way. □

Lemma 6.2.10. *Let r be a natural number, and $\mathscr{E}$ be an equivalence structure. There exists a set of equivalence structures V such that $\varphi^r_{\mathscr{E}}$ is a Boolean combination of sentences in $\bigcup_{\mathscr{E}'\in V}\Phi^r_{\mathscr{E},\mathscr{E}'}$.*

Proof. This proof can be directly adapted from Lemma 6.2.5. □

The following result states that, over equivalence structures, any first-order sentence is equivalent to a Boolean combination of distinguishability sentences.

Theorem 6.2.2. *Let φ be a first-order sentence over ES. Then, there exists a natural number r, and two sets U, V of equivalence structures such that φ is equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{E}\in U,\mathscr{E}'\in V}\Phi^r_{\mathscr{E},\mathscr{E}'}$.*

Proof. This proof can be directly adapted from Theorem 6.2.1. □

6.2.3 Distinguishability Sentences for DULO

Now, we define the distinguishability sentences for disjoint unions of linear orders. First, we define the following formulas:

$$\varphi^{<x,>y}_{\geq n} = \begin{cases} \exists z(z=z), & \text{se } n=0 \\ \exists z(z<x \wedge y<z), & \text{se } n=1 \\ \exists z(z<x \wedge y<z \wedge \varphi^{<z,>y}_{\geq \lfloor \frac{n-1}{2} \rfloor} \wedge \varphi^{<x,>z}_{\geq \lfloor \frac{n}{2} \rfloor}), & \text{c.c.} \end{cases}$$

$$\varphi^{<x}_{\geq n} = \begin{cases} \exists y(y=y), & \text{se } n=0 \\ \exists y(y<x), & \text{se } n=1 \\ \exists y(y<x \wedge \varphi^{<y}_{\geq \lfloor \frac{n-1}{2} \rfloor} \wedge \varphi^{<x,>y}_{\geq \lfloor \frac{n}{2} \rfloor}), & \text{c.c.} \end{cases}$$

$$\varphi^{>x}_{\geq n} = \begin{cases} \exists y(y = y), & \text{se } n = 0 \\ \exists y(x < y), & \text{se } n = 1 \\ \exists y(y > x \wedge \varphi^{>y}_{\geq \lfloor \frac{n-1}{2} \rfloor} \wedge \varphi^{<y,>x}_{\geq \lfloor \frac{n}{2} \rfloor}), & \text{c.c.} \end{cases}$$

$$\varphi_{q_t \geq p} := \exists x_1 ... \exists x_p (\bigwedge_{l \neq j} (x_l \neq x_j \wedge \neg(x_l < x_j) \wedge \neg(x_l > x_j)) \wedge \bigwedge_{k=1}^{p} (\varphi^{<x_k}_{=\lfloor \frac{t-1}{2} \rfloor} \wedge \varphi^{>x_k}_{=\lfloor \frac{t}{2} \rfloor})).$$

$$\varphi_{q_{\geq t} \geq p} = \exists x_1 ... \exists x_p (\bigwedge_{l \neq j} (x_l \neq x_j \wedge \neg(x_l < x_j) \wedge \neg(x_l > x_j)) \wedge \bigwedge_{k=1}^{p} (\varphi^{<x_k}_{\geq \lfloor \frac{t-1}{2} \rfloor} \wedge \varphi^{>x_k}_{\geq \lfloor \frac{t}{2} \rfloor})).$$

Formulas of the form $\varphi^{\star x}_{\geq n}$ such that $\star \in \{<, >\}$ and $\varphi^{<x,>y}_{\geq n}$ are used in the definition of $\varphi_{q_t \geq p}$ and $\varphi_{q_{\geq t} \geq p}$. We also define abbreviations $\varphi^{*x}_{<n} := \neg\varphi^{*x}_{\geq n}$, $\varphi^{*x}_{\leq n} := \neg\varphi^{*x}_{\geq n+1}$, and $\varphi^{*x}_{=n} := \varphi_{\geq n} \wedge \varphi_{\leq n}$, for $* \in \{<, >\}$. These formulas are defined recursively in order to obtain the adequate 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 minimum. Note that $qr(\varphi^{*x}_{\geq n}) = \lfloor log_2(n) \rfloor + 1$. Sentences of the form $\varphi_{q_t \geq p}$ express that the number q_t of linear orders of size t is at least p. Each variable x_k represents an element in a linear order. Analogously, $\varphi_{q_{\geq t} \geq p}$ holds in disjoint unions of linear orders such that the number $q_{\geq t}$ of linear orders of size at least t is at least p. We also define $\varphi_{q_t < p} := \neg\varphi_{q_t \geq p}$, $\varphi_{q_{\geq t} < p} := \neg\varphi_{q_{\geq t} \geq p}$. Finally, regarding the quantifier rank, $qr(\varphi_{q_t \geq p}) = p + \lfloor log_2(\lfloor \frac{t}{2} + 1 \rfloor) \rfloor + 1 = p + \lfloor log_2(\lceil \frac{t+1}{2} \rceil) \rfloor + 1$, and $qr(\varphi_{q_{\geq t} \geq p}) = p + \lfloor log_2(t) \rfloor$. Furthermore, the size of $\varphi_{q_t \geq p}$ and $\varphi_{q_{\geq t} \geq p}$ is $O((p+n)^2)$. Now, we define the distinguishability sentences for disjoint unions of linear orders.

Definition 6.2.3 (Distinguishability Sentences for DULO)**.** *Let $\mathscr{W}_1$, $\mathscr{W}_2$ be disjoint unions of linear orders and r be a natural number.*

$$\Phi^r_{\mathscr{W}_1,\mathscr{W}_2} := \begin{array}{l} \{\varphi_{q_t<p} \mid t < 2^r - 2, q_t^{\mathscr{W}_1} < q_t^{\mathscr{W}_2}, q_t^{\mathscr{W}_1} + 1 \leq p \leq min(q_t^{\mathscr{W}_2}, r - \lfloor log_2(\lceil \frac{t+1}{2} \rceil) \rfloor - 1)\} \cup \\ \{\varphi_{q_t \geq p} \mid t < 2^r - 2, q_t^{\mathscr{W}_2} < q_t^{\mathscr{W}_1}, q_t^{\mathscr{W}_2} + 1 \leq p \leq min(q_t^{\mathscr{W}_1}, r - \lfloor log_2(\lceil \frac{t+1}{2} \rceil) \rfloor - 1)\} \cup \\ \{\varphi_{q_{\geq t}<p} \mid t \leq 2^r - 1, q_{\geq t}^{\mathscr{W}_1} < q_{\geq t}^{\mathscr{W}_2}, q_t^{\mathscr{W}_1} + 1 \leq p \leq min(q_t^{\mathscr{W}_2}, r - \lfloor log_2(t) \rfloor)\} \cup \\ \{\varphi_{q_{\geq t} \geq p} \mid t \leq 2^r - 1, q_{\geq t}^{\mathscr{W}_2} < q_{\geq t}^{\mathscr{W}_1}, q_t^{\mathscr{W}_2} + 1 \leq p \leq min(q_t^{\mathscr{W}_1}, r - \lfloor log_2(t) \rfloor)\}. \end{array}$$

Given $\mathscr{W}_1$, $\mathscr{W}_2$, and r, the size of a sentence $\varphi \in \Phi^r_{\mathscr{W}_1,\mathscr{W}_2}$ is $O((|W_1| + |W_2|)^2)$. Since $t, p \leq |W_1| + |W_2|$, $|\Phi^r_{\mathscr{W}_1,\mathscr{W}_2}|$ is $O((|W_1| + |W_2|)^2)$. Now, we show results ensuring adequate properties of the distinguishability sentences for disjoint unions of linear orders. These results can be directly adapted from Lemma 6.2.6 and Lemma 6.2.7.

Lemma 6.2.11. *Let $\varphi \in \Phi^r_{\mathscr{W}_1,\mathscr{W}_2}$. Then, $\mathscr{W}_1 \models \varphi$ and $\mathscr{W}_2 \not\models \varphi$.*

Lemma 6.2.12. *Let $\varphi \in \Phi^r_{\mathscr{W}_1,\mathscr{W}_2}$. Then, $qr(\varphi) \leq r$.*

Now we show that, over DULO, any first-order sentence is equivalent to a Boolean combination of distinguishability sentences. The following results can be directly adapted from Lemma 6.2.8, Lemma 6.2.9, Lemma 6.2.10, and Theorem 6.2.2.

Lemma 6.2.13. $\models \varphi^r_{\mathscr{W}} \leftrightarrow (\bigwedge_{t=1}^{2^r-3} \varphi^{r,q_t}_{\mathscr{W}} \wedge \bigwedge_{t=1}^{2^r-1} \varphi^{r,q_{\geq t}}_{\mathscr{W}})$ *such that*

$$\varphi^{r,q_t}_{\mathscr{W}} := \begin{cases} \varphi_{q_t=q_t^{\mathscr{W}}}, & \text{if } q_t^{\mathscr{W}} + \lfloor log_2(\lceil \frac{t+1}{2} \rceil) \rfloor + 2 \leq r \\ \varphi_{q_t > r - \lfloor log_2(\lceil \frac{t+1}{2} \rceil) \rfloor - 2}, & \text{otherwise.} \end{cases}$$

$$\varphi^{r,q_{\geq t}}_{\mathscr{W}} := \begin{cases} \varphi_{q_{\geq t}=q^{\mathscr{W}}_{\geq t}}, & \text{if } q^{\mathscr{W}}_{\geq t} + \lfloor log(t) \rfloor + 1 \leq r \\ \varphi_{q_{\geq t} > r - \lfloor log(t) \rfloor - 1}, & \text{otherwise.} \end{cases}$$

Theorem 6.2.3. *Let φ be a first-order sentence over disjoint unions of linear orders. Then, there exists sets V, U of disjoint unions of linear orders such that φ is equivalent to a Boolean combination of sentences in $\bigcup_{\mathscr{W} \in V, \mathscr{W}' \in U} \Phi^r_{\mathscr{W},\mathscr{W}'}$.*

6.2.4 Distinguishability Sentences for Strings

Now, we define distinguishability sentences for strings u, v and a natural number r. Distinguishability sentences 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 3.3.1 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.

First, we introduce $\varphi^{d(t_1,t_2)}_{\leq n}$. 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 3.3.1.

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

We also set $\varphi^{d(t_1,t_2)}_{>n} := \neg\varphi^{d(t_1,t_2)}_{\leq n}$, $\varphi^{d(t_1,t_2)}_{\geq n} := \varphi^{d(t_1,t_2)}_{>n-1}$, $\varphi^{d(t_1,t_2)}_{<n} := \neg\varphi^{d(t_1,t_2)}_{\geq n}$, and $\varphi^{d(t_1,t_2)}_{=n} := \varphi^{d(t_1,t_2)}_{\leq n} \wedge \varphi^{d(t_1,t_2)}_{\geq n}$. Clearly, $w \models \varphi^{d(t_1,t_2)}_{\rhd n}$ iff $d(t_1,t_2) \rhd n$, for $\rhd \in \{<,>,\leq,\geq,=\}$. Also, the size of $\varphi^{d(t_1,t_2)}_{\rhd n}$ is $O(n)$. Furthermore, $qr(\varphi^{d(t_1,t_2)}_{\leq n}) = qr(\varphi^{d(t_1,t_2)}_{>n}) = \lceil log_2(n) \rceil$ and $qr(\varphi^{d(t_1,t_2)}_{\geq n}) = qr(\varphi^{d(t_1,t_2)}_{\leq n-1}) = \lceil log_2(n-1) \rceil$.

We also define the following formulas:

$$\varphi_{\geq n}^{<x} = \begin{cases} \exists y(y = y), & \text{if } n = 0 \\ \exists y S(y,x), & \text{if } n = 1 \\ \exists y(\varphi_{\geq \lfloor \frac{n}{2} \rfloor}^{<y} \wedge \varphi_{\geq \lceil \frac{n}{2} \rceil}^{d(y,x)}), & \text{otherwise} \end{cases}$$

$$\varphi_{\geq n}^{>x} = \begin{cases} \exists y(y = y), & \text{if } n = 0 \\ \exists y S(x,y), & \text{if } n = 1 \\ \exists y(\varphi_{\geq \lfloor \frac{n}{2} \rfloor}^{>y} \wedge \varphi_{\geq \lceil \frac{n}{2} \rceil}^{d(x,y)}), & \text{otherwise} \end{cases}$$

Observe that $qr(\varphi_{\geq n}^{<x}) = qr(\varphi_{\geq n}^{>x}) = \lceil log(n+1) \rceil$, and the size of $\varphi_{\geq n}^{<x}$ (and of $\varphi_{\geq n}^{>x}$) is $O(n)$. Now, we can define sentences $\varphi_{\geq n}$ which describes that a string has length at least n.

$$\varphi_{\geq n} = \begin{cases} \exists y(y = y), & \text{if } n \in \{0,1\} \\ \exists y(\varphi_{\geq \lfloor \frac{n}{2} \rfloor}^{>y} \wedge \varphi_{\geq \lfloor \frac{n-1}{2} \rfloor}^{<y}), & \text{otherwise} \end{cases}$$

It is important to note that $qr(\varphi_{\geq n}) = \lceil log(n+1) \rceil$, and the size of $\varphi_{\geq n}$ is $O(n)$. We also define $\varphi_{\leq n} := \neg \varphi_{\geq n+1}$ and $\varphi_{=n} := \varphi_{\geq n} \wedge \varphi_{\leq n}$. Then, $qr(\varphi_{=n}) = \lceil log(n+2) \rceil$.

We give an example to show that sentences $\varphi_{\geq n}$ represent condition 1 of Theorem 3.3.1. Let $r = 3$ and u, v be strings such that $|u| = 5$ and $|v| = 9$. Then, $u \not\models \varphi_{\geq 2^r-2}$, $v \models \varphi_{\geq 2^r-2}$, and $qr(\varphi_{\geq 6}) = 3$. Then, $|u| \neq |v|$ and $|u| < 2^r - 2$. Therefore, the Spoiler has a winning strategy for $\mathscr{G}_3(u,v)$.

Now, we turn to the cases in which substrings are important. These cases are conditions 2-4 from Theorem 3.3.1. Formulas $\varphi_{t_1 a_1 \ldots a_k t_2}$ hold in a string w when the string between t_1 and t_2 is $a_1 \ldots a_k$. Formulas $\varphi_{t a_1 \ldots a_k}$ and $\varphi_{a_1 \ldots a_k t}$ express that a string $a_1 \ldots a_k$ occurs immediately on the right and immediately on the left of a term t, respectively.

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

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

$$\varphi_{a_1\ldots a_k t} := \begin{cases} \exists y(S(y,t) \wedge P_{a_1}(y)), & \text{if } k = 1 \\ \exists y(P_{a_1}(y) \wedge \varphi_{ya_2 t}), & \text{if } k = 2 \\ \exists y(P_{a_{\lceil \frac{k}{2} \rceil}}(y) \wedge \varphi_{a_1\ldots a_{\lceil \frac{k}{2} \rceil - 1} y} \wedge \varphi_{ya_{\lceil \frac{k}{2} \rceil + 1}\ldots a_k t}), & \text{otherwise.} \end{cases}$$

Concerning the quantifier rank, we have $qr(\varphi_{t_1 a_1\ldots a_k t_2}) = qr(\varphi_{t a_1\ldots a_k}) = qr(\varphi_{a_1\ldots a_k t}) = \lceil log_2(k+1) \rceil$. 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 $a_1\ldots a_k$ and the suffix of length k is $a_1\ldots a_k$, respectively.

$$\varphi_{pref_k = a_1\ldots a_k} := \begin{cases} \exists x(P_{a_1}(x) \wedge \forall y \neg S(y,x)), & \text{if } k = 1 \\ \exists x(P_{a_1}(x) \wedge \forall y \neg S(y,x) \wedge \varphi_{xa_2\ldots a_k}), & \text{if } k \in \{2,3\} \\ \exists x_1(P_{a_1}(x_1) \wedge \forall y \neg S(y,x_1) \wedge & \\ \exists x_2(P_{a_{\lceil \frac{k+1}{2} \rceil}}(x_2) \wedge \varphi_{x_1 a_2\ldots a_{\lceil \frac{k+1}{2} \rceil - 1} x_2} \wedge \varphi_{x_2 a_{\lceil \frac{k+1}{2} \rceil + 1}\ldots a_k})), & \text{otherwise.} \end{cases}$$

$$\varphi_{suff_k = a_1\ldots a_k} := \begin{cases} \exists x(P_{a_1}(x) \wedge \forall y \neg S(x,y)), & \text{if } k = 1 \\ \exists x(P_{a_1}(x) \wedge \forall y \neg S(x,y) \wedge \varphi_{a_1\ldots a_{k-1} x}), & \text{if } k \in \{2,3\} \\ \exists x_1(P_{a_1}(x_1) \wedge \forall y \neg S(x_1,y) \wedge & \\ \exists x_2(P_{a_{\lfloor \frac{k}{2} \rfloor}}(x_2) \wedge \varphi_{a_1\ldots a_{\lfloor \frac{k}{2} \rfloor - 1} x_2} \wedge \varphi_{x_2 a_{\lfloor \frac{k}{2} \rfloor + 1}\ldots a_{k-1} x_1})), & \text{otherwise.} \end{cases}$$

We also set abbreviations $\varphi_{pref_k \neq a_1\ldots a_k} := \neg\varphi_{pref_k = a_1\ldots a_k}$ and $\varphi_{suff_k \neq a_1\ldots a_k} := \neg\varphi_{suff_k = a_1\ldots a_k}$. Therefore, $qr(\varphi_{pref_k \blacktriangleright a_1\ldots a_k}) = \lceil log_2(k+2) \rceil$ and $w \models \varphi_{pref_k \blacktriangleright a_1\ldots a_k}$ iff $pref_k(w) \blacktriangleright a_1\ldots a_k$, where $\blacktriangleright \in \{=, \neq\}$. Also, the size of $\varphi_{pref_k = a_1\ldots a_k}$ is $O(k)$. Analogously, for $\varphi_{suff_k \blacktriangleright a_1\ldots a_k}$.

We use these formulas to express conditions 2 and 3 from Theorem 3.3.1. To see why, let $r = 2$, $u = baabb$ and $v = bbabb$. Thus, $u \models \varphi_{pref_{2^r-2} \neq bb}$ and $v \not\models \varphi_{pref_{2^r-2} \neq bb}$. Then, $pref_{2^r-2}(u) \neq pref_{2^r-2}(v)$ and, from condition 2 of Theorem 3.3.1, it follows that the Spoiler has a winning strategy in $\mathscr{G}_2(u,v)$.

Now, we need sentences regarding multiplicity and scattering. Let $\alpha = a_1\ldots a_k$ be a string such that each $a_i \in \Sigma$, and $k = 2^{q_\alpha} - 1$ for $q_\alpha > 0$ as in condition 4 from Theorem 3.3.1. Now, we set the formula $\varphi_{a_1\ldots a_k}(x)$ describing that a string $a_1\ldots a_k$ occurs centered on position x. Then, we give an example of a formula $\varphi_\alpha(x)$.

$$\varphi_{a_1\ldots a_k}(x) := \begin{cases} P_{a_1}(x), & \text{if } k = 1 \\ P_{a_{\lceil \frac{k}{2} \rceil}}(x) \wedge \varphi_{a_1\ldots a_{\lceil \frac{k}{2} \rceil - 1} x} \wedge \varphi_{xa_{\lceil \frac{k}{2} \rceil + 1}\ldots a_k}, & \text{if } k = 2^{q_\alpha} - 1, q_\alpha > 1. \end{cases}$$

Example 6.2.3. *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 $O(|\alpha|)$. Now, we can use formulas $\varphi_\alpha(x)$ to define $\varphi_{\gamma(\alpha)\geq n}$ expressing that α has at least n occurrences. Then, we need to use n pairwise different variables.

$$\varphi_{\gamma(\alpha)\geq n} := \ \exists x_1 \exists x_2 ... \exists x_n (\textstyle\bigwedge_{1\leq i<j\leq n} x_i \neq x_j \wedge \bigwedge_{i=1}^{n} \varphi_\alpha(x_i)).$$

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 define auxiliary formulas in order to make the presentation simpler. The first 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. The second formula expresses that α occurs centered on a position between x_1 and x_2, on the right of x_1, at a distance greater than $2^{q_\alpha-1}$, on the left of x_2, and at least $2^{q_\alpha-1}$ distant from x_2. These formulas are important in ensuring a proper distance from other occurrences of α, i.e, greater than 2^{q_α}.

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

$$\varphi^{d(x_1,\alpha)>2^{q_\alpha-1}}_{d(\alpha,x_2)\geq 2^{q_\alpha-1}} := \exists y(\varphi_\alpha(y) \wedge \varphi^{d(x_1,y)}_{>2^{q_\alpha-1}} \wedge \varphi^{d(y,x_2)}_{\geq 2^{q_\alpha-1}}).$$

Observe that, with respect to the quantifier rank, $qr(\varphi_\alpha^{d(\alpha,x)\geq 2^{q_\alpha-1}}) = qr(\varphi^{d(x_1,\alpha)>2^{q_\alpha-1}}_{d(\alpha,x_2)\geq 2^{q_\alpha-1}}) = q_\alpha$. Furthermore, the size of $\varphi^{d(\alpha,x)\geq 2^{q_\alpha-1}}$ (and of $\varphi^{d(x_1,\alpha)>2^{q_\alpha-1}}_{d(\alpha,x_2)\geq 2^{q_\alpha-1}}$) is $O(|\alpha|)$. Now, we can define the sentence $\varphi_{\sigma(\alpha)\geq n}$. After that, we give an example of $\varphi_{\gamma(\alpha)\geq n}$ and $\varphi_{\sigma(\alpha)\geq n}$.

$$\varphi_{\sigma(\alpha)\geq n} := \begin{cases} \varphi_{\gamma(\alpha)\geq 1}, & \text{if } n = 1 \\ \exists x_1(\varphi^{d(\alpha,x_1)\geq 2^{q_\alpha-1}} \wedge \exists x_2(\varphi^{d(x_1,\alpha)>2^{q_\alpha-1}}_{d(\alpha,x_2)\geq 2^{q_\alpha-1}} \wedge ... \wedge \\ \exists x_{n-1}(\varphi^{d(x_{n-2},\alpha)>2^{q_\alpha-1}}_{d(\alpha,x_{n-1})\geq 2^{q_\alpha-1}} \wedge \exists x_n(\varphi^{d(x_{n-1},x_n)}_{>2^{q_\alpha-1}} \wedge \varphi_\alpha(x_n)))...)), & \text{if } n > 1. \end{cases}$$

Example 6.2.4. *Let $\alpha = abc$ and $n = 2$. Then, $q_\alpha = 2$. Thus,*

$$\varphi_{\gamma(\alpha)\geq n} = \ \exists x_1 \exists x_2(x_1 \neq x_2 \wedge \varphi_{abc}(x_1) \wedge \varphi_{abc}(x_2)).$$

$$\varphi_{\sigma(\alpha)\geq n} = \ \exists x_1(\varphi^{d(abc,x_1)\geq 2} \wedge \exists x_2(\varphi^{d(x_1,x_2)}_{>2} \wedge \varphi_{abc}(x_2))).$$

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, $qr(\varphi_{\gamma(\alpha)\trianglelefteq n}) = q_\alpha + n - 1$ and $w \models \varphi_{\gamma(\alpha)\trianglelefteq n}$ iff $\gamma(\alpha,w) \trianglelefteq n$, for $\trianglelefteq \in \{\geq, <\}$. It is analogous for $\varphi_{\sigma(\alpha)\trianglelefteq n}$. Besides, the size of $\varphi_{\gamma(\alpha)\trianglelefteq n}$ (and of $\varphi_{\sigma(\alpha)\trianglelefteq n}$) is $O((n+|\alpha|)^2)$.

The formulas $\varphi_{\sigma(\alpha)\geq n}$ and $\varphi_{\gamma(\alpha)\geq n}$ defined above are not sufficient for the case where there exists α such that $|\alpha| = 2^{q_\alpha} - 1$, for some $q_\alpha > 0$, $\sigma(\alpha,u) = \sigma(\alpha,v)$, $(\sigma(\alpha,u) + q_\alpha \leq r$, and $\gamma(\alpha,u) \neq \gamma(\alpha,v)$. In this case, we may have $\gamma(\alpha,u), \gamma(\alpha,v) + q_\alpha > r$. Then, formulas $\varphi_{\gamma(\alpha)\geq n}$ are not adequate since $qr(\varphi_{\gamma(\alpha)\geq\gamma(\alpha,v)+1}) > r$. Therefore, we now define formulas to represent the neighborhood of the segments in $(|\alpha|+1)$-segmentations. First, we use the following definitions.

Definition 6.2.4 (r-sphere)**.** *Let $w = w_1...w_n$, i be a position of w, and r be a positive integer. The r-sphere of i in w, denoted by $\mathscr{S}(i,w,r)$, is defined as follows:*

$$\mathscr{S}(i,w,r) = \{j \in \{1,...,n\} \mid i \leq j, d(i,j) \leq r\}.$$

Definition 6.2.5 (r-neighbourhood)**.** *Let $w = w_1...w_n$, i be a position of w, and r be a positive integer. The r-neighbourhood $\mathscr{N}(i,w,r)$ of i in w is the substring induced by $\mathscr{S}(i,w,r)$.*

We associate a substring β of w with each segment S of a $(|alpha|+1)$-segmentation of $\Gamma(\alpha,w)$. We formalize this notion in the following definition.

Definition 6.2.6 (Neighbourhood of Segments)**.** *Let w and α be strings. Let $\sigma(\alpha,w) = m$. Let $S_1,...,S_m$ be the segments in a $(|alpha|+1)$-segmentation of $\Gamma(\alpha,w)$. We define $\mathscr{N}(\alpha,w)$ as follows:*

$$\mathscr{N}(\alpha,w) = \{\mathscr{N}(i,w,2\times|\alpha|) \mid i = min(S_l), l \in \{1,...,m\}\}.$$

Observe that $\mathscr{N}(\alpha,w)$ may be a multiset of strings. We give an example below.

Example 6.2.5. *Let $\alpha = aba$ and $w = aaababababbabaababaaa$. Then, the neighborhood of the segments in a $(|alpha|+1)$-segmentation of $\Gamma(\alpha,w)$ is $\mathscr{N}(\alpha,w) = \{abababa, abaabab, abaaa\}$.*

In the following, we define formulas $\varphi_{\mathscr{N}(\alpha,w)}$ to represent the neighbourhood of segments. First, let α, β_1, ..., β_m be strings. Let $\{\beta_1,...,\beta_m\}$ the multiset which consists of strings β_1, ..., β_m. Then, it is possible that, for some $i,j \in \{1,...,m\}$ such that $i \neq j$, $\beta_i = \beta_j$. We define the following abbreviation.

$$\varphi_{\mathscr{N}(\alpha,\{\beta_1,...,\beta_m\})} = \exists x_1(\varphi_{\beta_1}(x_1) \wedge \exists x_2(\varphi_{>2^{q\alpha}}^{d(x_1,x_2)} \wedge \varphi_{\beta_2}(x_2) \wedge ...\exists x_m(\varphi_{>2^{q\alpha}}^{d(x_{m-1},x_m)} \wedge \varphi_{\beta_m}(x_m))...)).$$

Note that $qr(\varphi_{\mathscr{N}(\alpha,\{\beta_1,...,\beta_m\})}) = q_\alpha + m$. Let w and α be strings. Let $\sigma(\alpha,w) = m$. Let $S_1,...,S_m$ be the segments in a $(|\alpha|+1)$-segmentation of $\Gamma(\alpha,w)$. Let $S_l = \{i_1^l,...,i_{l_k}^l\}$, for $l \in \{1,...,m\}$, be

the positions of segment S_l. Let $\beta_l = u_{i_1^l} u_{i_1^l+1} ... u_{i_2^l} ... u_{i_{l_k}^l} u_{i_{l_k}^l+1} ... u_{i_1^l+2\times|\alpha|}$. We define the formula $\varphi_{\mathcal{N}(\alpha,w)}$ as $\varphi_{\mathcal{N}(\alpha,\mathcal{N}(\alpha,w))}$. In other words, $\varphi_{\mathcal{N}(\alpha,w)}$ is defined as follows:

$$\varphi_{\mathcal{N}(\alpha,w)} = \exists x_1(\varphi_{\beta_1}(x_1) \wedge \exists x_2(\varphi_{>2^{q\alpha}}^{d(x_1,x_2)} \wedge \varphi_{\beta_2}(x_2) \wedge ... \exists x_m(\varphi_{>2^{q\alpha}}^{d(x_{m-1},x_m)} \wedge \varphi_{\beta_m}(x_m))...)).$$

Note that $qr(\varphi_{\mathcal{N}(\alpha,w)}) = q_\alpha + \sigma(\alpha,u)$ and the size of $\varphi_{\mathcal{N}(\alpha,w)}$ is $O(|w|)$. Moreover, $w \models \varphi_{\mathcal{N}(\alpha,u)}$ iff $\mathcal{N}(\alpha,u) \subseteq \mathcal{N}(\alpha,w)$. Since $\mathcal{N}(\alpha,u)$ and $\mathcal{N}(\alpha,w)$ are multisets, for $\beta \in \mathcal{N}(\alpha,u)$, the number of occurrences of β in $\mathcal{N}(\alpha,w)$ must be at least the number of occurrences of β in $\mathcal{N}(\alpha,u)$.

Example 6.2.6. *Let $\alpha = aba$ and $w = aaababababbabaababaaa$. Then,*

$$\varphi_{\mathcal{N}(\alpha,w)} = \exists x_1(\varphi_{abababa}(x_1) \wedge \exists x_2(\varphi_{>4}^{d(x_1,x_2)} \wedge \varphi_{abaabab}(x_2) \wedge \exists x_3(\varphi_{>4}^{d(x_2,x_3)} \wedge \varphi_{abaaa}(x_3)))).$$

Now, we can define the distinguishability sentences for strings. Distinguishability sentences 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 6.2.7 (Distinguishability Sentences for Strings)**.** *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*

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

such that

$$\Phi_{u,v}^{r,length} := \begin{array}{l} \{\varphi_{\leq n} \mid |u| < |v|, |u| \leq n \leq min(2^r-2, |v|-1)\} \cup \\ \{\varphi_{\geq n} \mid |u| > |v|, |v|+1 \leq n \leq min(2^r-1, |u|)\} \end{array}$$

$$\Phi_{u,v}^{r,pref} := \begin{array}{l} \{\varphi_{pref_k = pref_k(u)} \mid pref_k(u) \neq pref_k(v), k \leq min(2^r-2, |u|, |v|)\} \cup \\ \{\varphi_{pref_k \neq pref_k(v)} \mid pref_k(u) \neq pref_k(v), k \leq min(2^r-2, |u|, |v|)\} \end{array}$$

$$\Phi_{u,v}^{r,suff} := \begin{array}{l} \{\varphi_{suff_k = suff_k(u)} \mid suff_k(u) \neq suff_k(v), k \leq min(2^r-2, |u|, |v|)\} \cup \\ \{\varphi_{suff_k \neq suff_k(v)} \mid suff_k(u) \neq suff_k(v), k \leq min(2^r-2, |u|, |v|)\} \end{array}$$

$$\Phi_{u,v}^{r,sub} := \begin{array}{l} \{\varphi_{\sigma(\alpha)\geq n} \mid \sigma(\alpha,u) > \sigma(\alpha,v), \sigma(\alpha,v) < n \leq min(r-q_\alpha+1, \sigma(\alpha,u))\} \cup \\ \{\varphi_{\sigma(\alpha)<n} \mid \sigma(\alpha,u) < \sigma(\alpha,v), \sigma(\alpha,u) < n \leq min(r-q_\alpha+1, \sigma(\alpha,v))\} \cup \\ \{\varphi_{\gamma(\alpha)\geq n} \mid \gamma(\alpha,u) > \gamma(\alpha,v), \gamma(\alpha,v) < n \leq min(r-q_\alpha+1, \gamma(\alpha,u))\} \cup \\ \{\varphi_{\gamma(\alpha)<n} \mid \gamma(\alpha,u) < \gamma(\alpha,v), \gamma(\alpha,u) < n \leq min(r-q_\alpha+1, \gamma(\alpha,v))\} \cup \\ \{\varphi_{\mathcal{N}(\alpha,u)} \mid \sigma(\alpha,u) + q_\alpha \leq r, \sigma(\alpha,u) = \sigma(\alpha,v), \gamma(\alpha,u) > \gamma(\alpha,v)\} \cup \\ \{\neg\varphi_{\mathcal{N}(\alpha,v)} \mid \sigma(\alpha,u) + q_\alpha \leq r, \sigma(\alpha,u) = \sigma(\alpha,v), \gamma(\alpha,u) < \gamma(\alpha,v)\}. \end{array}$$

Observe that, given u, v, and r, the size of a formula $\varphi \in \Phi^r_{u,v}$ is $O((|u|+|v|)^2)$, and the number of elements $|\Phi^r_{u,v}|$ in $\Phi^r_{u,v}$ is $O((|u|+|v|)^3)$. This is crucial in order to guarantee that our algorithm runs in polynomial time. Also, by the definition of distinguishability sentences and Theorem 2.3.1, it follows that the Spoiler has a winning strategy in $\mathscr{G}_r(u,v)$ if and only if there exists $\varphi \in \Phi^r_{u,v}$. In what follows, we give examples of distinguishability sentences.

Example 6.2.7. *Let $u = acbbb$, $v = aabbbb$, and $r = 2$. Therefore, $\varphi_{pref_2=ac}$, $\varphi_{pref_2\neq aa} \in \Phi^r_{u,v}$. Furthermore, $\varphi_{suff_2=bbb}, \varphi_{suff_2\neq bbb} \notin \Phi^r_{u,v}$ because $suff_2(u) = suff_2(v)$. Also, $\varphi_{\gamma(c)\geq 1} \in \Phi^r_{u,v}$ because $\gamma(c,u) > \gamma(c,v)$ and $\gamma(c,v) < 1 \leq min(2,1)$. Besides, $\varphi_{\gamma(bbb)<1} \in \Phi^r_{u,v}$ because $\gamma(bbb,u) < \gamma(bbb,v)$ and $\gamma(bbb,u) < 1 \leq min(1,1)$. Regarding the length, $\Phi^{r,length}_{u,v} = \emptyset$ because $|u| > min(2,5)$.*

Example 6.2.8. *Now, let $u = bbaaaaaaaabb$, $v = bbaaaaaabb$, and $r = 4$. Then, $\varphi_{\sigma(aaa)\geq 2} \in \Phi^r_{u,v}$ since $\sigma(aaa,u) = 2$, $\sigma(aaa,v) = 1$, and $\sigma(aaa,v) < n \leq min(3,2)$. Besides, $\varphi_{\geq 12} \in \Phi^r_{u,v}$ because $|u| > |v|$ and $11 \leq 12 \leq min(15,12)$.*

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

Lemma 6.2.14. *Let u,v be strings and r be a natural number. Let $\varphi \in \Phi^r_{u,v}$. Then, $u \models \varphi$ and $v \not\models \varphi$.*

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

Now, let $\varphi = \varphi_{pref_k=pref_k(u)}$. Then, $pref_k(u) \neq pref_k(v)$. It also holds that $k \leq min(2^r-2,|u|,|v|)$. 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.

Finally, assume that $\varphi = \varphi_{\mathscr{N}(\alpha,u)}$. Clearly, $u \models \varphi$. Furthermore, $\sigma(\alpha,u) = \sigma(\alpha,v)$ and $\gamma(\alpha,u) > \gamma(\alpha,v)$. Therefore, $v \not\models \varphi$. It is analogous when $\varphi = \neg\varphi_{\mathscr{N}(\alpha,v)}$. □

Lemma 6.2.15. *Let u,v be strings and r be a natural number. Let $\varphi \in \Phi^r_{u,v}$. Then, the minimum number of rounds is $MinRound(u,v) \leq qr(\varphi) \leq r$.*

Proof. From Lemma 6.2.14, it follows that $MinRound(u,v) \leq qr(\varphi)$. Now, we need to show that $qr(\varphi) \leq r$.

If $\varphi = \varphi_{\geq n}$, then $n \leq 2^r - 1$. Hence, $qr(\varphi) = \lceil log_2(2^r - 1 + 1) \rceil$. It follows that $qr(\varphi) \leq \lceil log_2(2^r) \rceil = r$. It is analogous for $\varphi = \varphi_{\leq n}$.

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

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, $qr(\varphi) = q_\alpha + n - 1 \leq r$.

Finally, let $\varphi = \varphi_{\mathcal{N}(\alpha,u)}$ or $\varphi = \neg\varphi_{\mathcal{N}(\alpha,v)}$. Then, $\sigma(\alpha,u) + q_\alpha \leq r$. Therefore, $qr(\varphi) \leq r$. □

Distinguishability formulas are also representative for the set of first-order sentences over strings. 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, $\varphi_{\gamma(a)\geq 1} \in \Phi^r_{u,v_1}$ and $\varphi_{\gamma(a)<2} \in \Phi^r_{u,v_2}$. Clearly, φ is equivalent to $\varphi_{\gamma(a)\geq 1} \wedge \varphi_{\gamma(a)<2}$. By this example, it seems that a first-order sentences consists of Boolean combination of sentences $\varphi_{\geq n}$, $\varphi_{pref_k = a_1 \ldots a_k}$, $\varphi_{suff_k = a_1 \ldots a_k}$, $\varphi_{\gamma(\alpha)\geq n}$, $\varphi_{\sigma(\alpha)\geq n}$, and $\varphi_{\alpha,u}$. 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.

$$\varphi_w^{r,length} := \begin{cases} \varphi_{=|w|}, & \text{if } |w| < 2^r - 2 \\ \varphi_{\geq 2^r - 2}, & \text{otherwise.} \end{cases}$$

$$\varphi_w^{r,pref} := \varphi_{pref_{2^r} = pref_{2^r}(w)}$$

$$\varphi_w^{r,suff} := \varphi_{suff_{2^r} = suff_{2^r}(w)}$$

$$\varphi_w^{r,\alpha} := \begin{cases} \varphi_{\sigma(\alpha)=\sigma(\alpha,w)} \wedge \varphi_{\gamma(\alpha)=\gamma(\alpha,w)}, & \text{if } q_\alpha + \sigma(\alpha,w) \leq r \\ \varphi_{\sigma(\alpha)\geq r - q_\alpha + 1}, & \text{otherwise.} \end{cases}$$

$$\varphi_w^{r,sub} := \bigwedge \{\varphi_w^{r,\alpha} \mid |\alpha| = 2^q - 1, q > 0\}.$$

Lemma 6.2.16. $\models \varphi_w^r \leftrightarrow (\varphi_w^{r,length} \wedge \varphi_w^{r,pref} \wedge \varphi_w^{r,suff} \wedge \varphi_w^{r,sub})$.

Proof. Let $u \models \varphi_w^r$. Then, the Duplicator has a winning strategy in $\mathcal{G}_r(w,u)$. Then, $|w| = |u|$ or $|w| \geq 2^r - 2$ and $|u| \geq 2^r - 2$. We have two cases depending on the length of w:

1. $|w| < 2^r - 2$. Then, $|w| = |u|$, and it follows that $u \models \varphi_w^{r,length}$.

2. $|w| \geq 2^r - 2$. Then, $|w| \geq 2^r - 2$ and $|u| \geq 2^r - 2$. Therefore, $u \models \varphi_w^{r,length}$.

From Theorem 3.3.1, it also holds that $pref_{2^r-2}(w) = pref_{2^r-2}(u)$ and $suff_{2^r-2}(w) = suff_{2^r-2}(u)$. Then, $u \models \varphi_w^{r,pref} \wedge \varphi_w^{r,suff}$.

From Theorem 3.3.1, 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 $\sigma(\alpha,u) + q \leq r$, then $\sigma(\alpha,w) \neq \sigma(\alpha,u)$. Then, it follows that $\sigma(\alpha,u) + q > r$. 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 $pref_{2^r-2}(w) = pref_{2^r-2}(u)$ 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. $|w| < 2^r - 2$. Then, $|w| = |u|$.

2. $|w| \geq 2^r - 2$. Then, $|u| \geq 2^r - 2$.

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 $\sigma(\alpha,u) + q > r$. By Theorem 3.3.1, the Duplicator has a winning strategy in $\mathscr{G}_r(w,u)$. It follows that $u \models \varphi_w^r$. □

Now, we need the following lemmas.

Lemma 6.2.17. *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 $\varphi_w^{r,length} = \varphi_{\geq 2^r-2}$, then let $V_{length} = \{v\}$ such that $|v| = 2^r - 3$. Observe that $\varphi_{\geq 2^r-2} \in \Phi_{w,v}^r$ because $|w| > |v|$ and $|v| + 1 \leq 2^r - 2 \leq min(2^r - 1, |w|)$. If $\varphi_w^{r,length} = \varphi_{=|w|}$, then let $V_{length} = \{v_1, v_2\}$ such that $|v_1| = |w| - 1$ and $|v_2| = |w| + 1$. It follows that $\varphi_{\geq |w|} \in \Phi_{w,v_1}^r$. We also have that $\varphi_{\leq |w|} \in \Phi_{w,v_2}^r$. Obviously, $\varphi_w^{r,length}$ is equivalent to $\varphi_{\geq |w|} \wedge \varphi_{\leq |w|}$. □

Lemma 6.2.18. *Let r be a natural number and w a string. There is a set of strings $V_{prefsuff}$ such that $\varphi_w^{r,pref} \wedge \varphi_w^{r,suff}$ is equivalent to a Boolean combination of sentences in $\bigcup_{v \in V_{prefsuff}} \Phi_{w,v}^r$.*

Proof. If $|w| \leq 2^r - 2$, then $\models \varphi_{pref_{2^r-2} = pref_{2^r-2}(w)} \leftrightarrow \varphi_{pref_{|w|}=w}$. Let v_1 such that $|v_1| = |w|$ and $v_1 \neq w$. Hence, $\varphi_{pref_{|w|}=w} \in \Phi_{w,v_1}^r$. If $w = a_1 ... a_{2^r-3} a_{2^r-2} u$ such that $a_i \in \Sigma$, for $i \in \{1, ..., 2^r - 2\}$,

$u \in \Sigma^*$, and $u \neq \varepsilon$. Then, $\varphi_{pref_{2^r-2}=pref_{2^r-2}(w)} \in \Phi^r_{w,v_1}$. Obviously, the case for $\varphi_{suff_{2^r}=suff_{2^r}(w)}$ is analogous. Let v_2 such that $\varphi_{suff_{2^r}=suff_{2^r}(w)} \in \Phi^r_{w,v_2}$. Therefore, $V_{prefsuff} = \{v_1, v_2\}$. □

Lemma 6.2.19. *Let r be a natural number and w, α strings. There is a set of strings V_α such that $\varphi^{r,\alpha}_w$ is equivalent to a Boolean combination of sentences in $\bigcup_{v\in V_\alpha} \Phi^{r'}_{w,v}$, for some r'.*

Proof. If $\varphi^{r,\alpha}_w = \varphi_{\sigma(\alpha)\geq r-q_\alpha+1}$, 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, $\varphi_{\sigma(\alpha)\geq r-q_\alpha+1} \in \Phi^r_{w,v}$. If $\varphi^{r,\alpha}_w = \varphi_{\sigma(\alpha)=\sigma(\alpha,w)} \wedge \varphi_{\gamma(\alpha)=\gamma(\alpha,w)}$, 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, $\varphi_{\sigma(\alpha)\geq\sigma(\alpha,w)} \in \Phi^r_{w,v_1}$. 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, $\varphi_{\sigma(\alpha)<\sigma(\alpha,w)+1} \in \Phi^r_{w,v_1}$. Clearly, the case for $\varphi_{\gamma(\alpha)=\gamma(\alpha,w)}$ is analogous and the only difference is that we use $r' = q_\alpha + \gamma(\alpha,w)+1$. Let v_3 and v_4 such that $\varphi_{\gamma(\alpha)\geq\gamma(\alpha,w)} \in \Phi^{r'}_{u,v_3}$ and $\varphi_{\gamma(\alpha)<\gamma(\alpha,w)+1} \in \Phi^{r'}_{u,v_4}$. Therefore $V_\alpha = \{v_1, v_2, v_3, v_4\}$. □

Lemma 6.2.20. *Let r be a natural number and w a string. There is a set of strings V such that φ^r_w is a Boolean combination of sentences in $\bigcup_{v\in V} \Phi^{r'}_{w,v}$, for some r'.*

Proof. By Lemma 6.2.16, $\models \varphi^r_w \leftrightarrow (\varphi^{r,length}_w \wedge \varphi^{r,pref}_w \wedge \varphi^{r,suff}_w \wedge \varphi^{r,sub}_w)$. Let $V = V_{length} \cup \bigcup_{\{\alpha||\alpha|=2^q-1, q>0\}} V_\alpha \cup V_{prefsuff}$ as in Lemma 6.2.17, Lemma 6.2.18, and Lemma 6.2.19. From these lemmas, it follows that, φ^r_w is equivalent to a Boolean combination of sentences in $\bigcup_{v\in V} \Phi^{r'}_{w,v}$, for some r'. □

The following result is connected to Theorem 2.3.2. It suggests that our approach is likely to find any first-order sentence given a suitable sample of strings. Recall that, by Theorem 2.3.2, any first-order sentence is equivalent to a disjunction of Hintikka formulas. Thus, we have the following result.

Theorem 6.2.4. *Let φ be a first-order sentence over strings. Then, φ is equivalent to a Boolean combination of distinguishability formulas.*

Proof. Let r such that $qr(\varphi) = r$. From Theorem 2.3.2 it follows that $\models \varphi \leftrightarrow \varphi^r_{u_1} \vee ... \varphi^r_{u_s}$. Let $U = \{u_1, ..., u_s\}$. In accord to Lemma 6.2.20, let V_i be such that $\varphi^r_{u_i}$ is equivalent to a Boolean combination of sentences in $\bigcup_{v\in V_i} \Phi^r_{u_i,v}$. Therefore, the sentence φ is equivalent to a Boolean combination of sentences in $\bigcup_{i=1}^s (\bigcup_{v\in V_i} \Phi^r_{u_i,v})$. □

6.3 A Polynomial Time Algorithm

In this section, we define an algorithm for finding a first-order sentence φ_S from a sample of structures S. Subformulas of φ_S are distinguishability sentences from sets of the form $\Phi^r_{\mathscr{A},\mathscr{B}}$ such that $\mathscr{A} \in P$ and $\mathscr{B} \in N$. 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 sizes of all the structures it includes. 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 minimum quantifier rank consistent with S. The pseudocode of our algorithm is in Algorithm 4.

Algorithm 4:

Input: Sample $S = (P,N)$
$r \leftarrow max\{MinRound(\mathscr{A},\mathscr{B}) \mid \mathscr{A} \in P,\ \mathscr{B} \in N\}$
$\varphi_S \leftarrow \bigvee_{\mathscr{A}\in P} \bigwedge_{\mathscr{B}\in N}$ **choose** $\varphi \in \Phi^r_{\mathscr{A},\mathscr{B}}$
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 builds φ_S. For each pair of structures $\mathscr{A} \in P$, $\mathscr{B} \in N$, it chooses $\varphi \in \Phi^r_{\mathscr{A},\mathscr{B}}$. Any choice of a sentence in $\Phi^r_{\mathscr{A},\mathscr{B}}$ leads to a formula consistent with S. In what follows, we show examples of how this algorithm works on simple instances.

Figure 30 – A sample of DULO

Source: Own elaboration

Example 6.3.1. *Let $P = \{\mathscr{W}_1, \mathscr{W}_2\}$, and $N = \{\mathscr{W}_3, \mathscr{W}_4\}$ as described in Figure 30. Then, $max\{MinRound(\mathscr{A},\mathscr{B}) \mid \mathscr{A} \in P,\ \mathscr{B} \in N\} = 2$. Also, $\varphi_{q_1\geq 1} \in \Phi^2_{\mathscr{W}_1,\mathscr{W}_3}$, $\varphi_{q_{\geq 2}<1} \in \Phi^2_{\mathscr{W}_1,\mathscr{W}_4}$, $\varphi_{q_{\geq 2}<1} \in \Phi^2_{\mathscr{W}_2,\mathscr{W}_3}$, $\varphi_{q_{\geq 3}<1} \in \Phi^2_{\mathscr{W}_2,\mathscr{W}_4}$. Therefore, Algorithm 4 returns*

$$\varphi_S = (\varphi_{q_1\geq 1} \wedge \varphi_{q_{\geq 2}<1}) \vee (\varphi_{q_{\geq 2}<1} \wedge \varphi_{q_{\geq 3}<1}).$$

Now, we show an example in which the sample consists of strings.

Table 1 – A sample of strings.

String	Class
$stviil$	Positive
$ktvive$	Negative
$stviie$	Positive
$stpiie$	Negative

Source: Own elaboration

Example 6.3.2. *Let S be the sample in Table 1 repeated above for convenience. Note that $max\{MinRound(u,v) \mid u \in P,\ v \in N\} = 1$. Clearly, $\varphi_{\gamma(s)\geq 1} \in \Phi^1_{stviil,ktvive}$, $\varphi_{\gamma(e)<1} \in \Phi^1_{stviil,stpiie}$, $\varphi_{\gamma(k)<1} \in \Phi^1_{stviie,ktvive}$, and $\varphi_{\sigma(p)<1} \in \Phi^1_{stviie,stpiie}$. Therefore, Algorithm 4 returns φ_S below. Observe that φ_S is consistent with S and $qr(\varphi_S) = 1$.*

$$\varphi_S = (\varphi_{\gamma(s)\geq 1} \wedge \varphi_{\gamma(e)<1}) \vee (\varphi_{\gamma(k)<1} \wedge \varphi_{\sigma(p)<1}).$$

In what follows, we illustrate a run of Algorithm 4 on a sample of monadic structures. Figure 31 describes a sample of MS over the vocabulary $\tau = \{P_1\}$ and $P_2 = \overline{P_1}$. Axis $|P_1|$ represents the cardinality of P_1 for all monadic structures in the given sample. Axis $|P_2|$ is analogous. In Figure 31, blue squares represent positive structures, while red triangles represent negative structures.

Example 6.3.3. *Let S be the sample defined in Figure 31, and let $r = 14$. There exists a sequence of choices of first-order sentences in Algorithm 4 such that*

$$\varphi_S = (\varphi_{|P_1|\geq 11} \wedge \varphi_{|P_2|\geq 6}) \vee (\varphi_{|P_1|<11} \wedge \varphi_{|P_2|<6}).$$

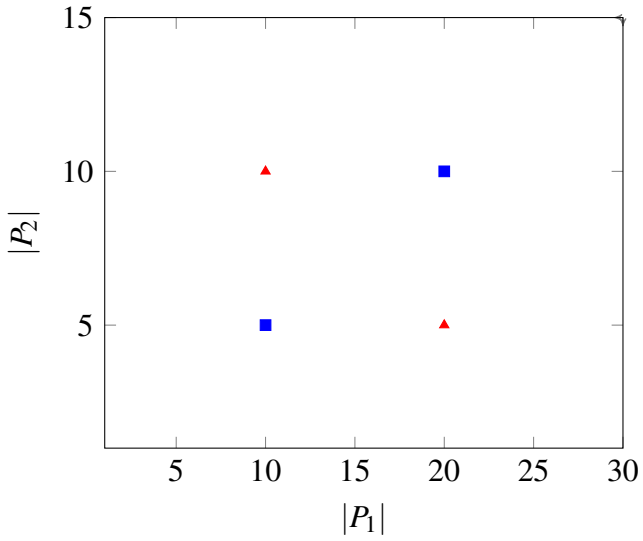
Finally, we consider a run of Algorithm 4 on a sample of equivalence structures.

Example 6.3.4. *Let S be the sample of equivalence structures in Figure 32, i.e., $P = \{\mathscr{E}_1\}$ and $N = \{\mathscr{E}_2, \mathscr{E}_3\}$. Let $r = 6$. Then, $\varphi_{q_1} < 1 \in \Phi^r_{\mathscr{E}_1,\mathscr{E}_2}$ and $\varphi_{q_{\geq 4}} < 1 \in \Phi^r_{\mathscr{E}_1,\mathscr{E}_3}$. Therefore, Algorithm 4 may return the following sentence:*

$$\varphi_S = \varphi_{q_1<1} \wedge \varphi_{q_{\geq 4}<1}.$$

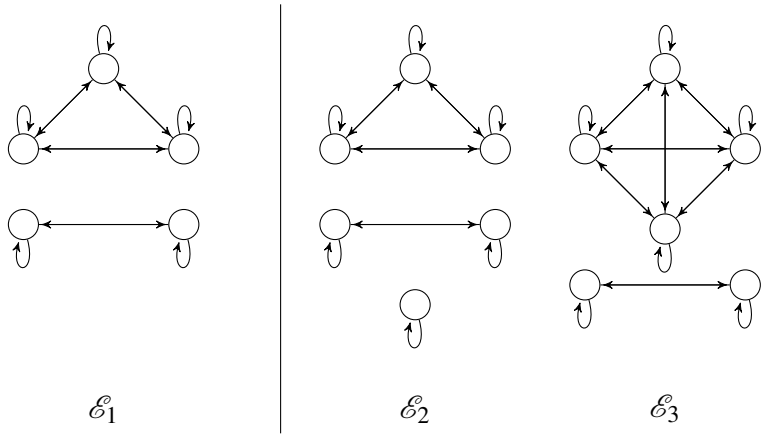
In what follows, we prove some properties of our algorithm. First, we show that it returns a sentence that is consistent with the sample. After that, we show that it returns a sentence of

Figure 31 – A sample of monadic structures



Source: Own elaboration

Figure 32 – A sample of equivalence structures



Source: Own elaboration

minimum quantifier rank. Then, we prove that the running time of our algorithm is polynomial in the size of the input sample.

Theorem 6.3.1. *Let S be a sample and φ_S returned by Algorithm 4. Then, φ_S is consistent with S.*

Proof. Let $\varphi_S = \bigvee_{\mathscr{A}\in P} \bigwedge_{\mathscr{B}\in N} \varphi^r_{\mathscr{A},\mathscr{B}}$ such that $r = max\{MinRound(\mathscr{A},\mathscr{B}) \mid \mathscr{A} \in P, \mathscr{B} \in N\}$, and $\varphi^r_{\mathscr{A},\mathscr{B}}$ is the sentence chosen from $\Phi^r_{\mathscr{A},\mathscr{B}}$. Let $\mathscr{A}' \in P$. In this way, $\mathscr{A}' \models \bigwedge_{\mathscr{B}\in N} \varphi^r_{\mathscr{A}',\mathscr{B}}$, and then $\mathscr{A}' \models \varphi_S$. Now, let $\mathscr{B}' \in N$ and assume that $\mathscr{B}' \models \varphi_S$, i.e, $\mathscr{B}' \models \bigwedge_{\mathscr{B}\in N} \varphi^r_{\mathscr{A}',\mathscr{B}}$, for some $\mathscr{A}' \in P$. Therefore, $\mathscr{B}' \models \varphi^r_{\mathscr{A}',\mathscr{B}'}$. This is an absurd because, from Lemma 6.2.14, it follows that

$\mathscr{B}' \not\models \varphi^r_{\mathscr{A}',\mathscr{B}'}$. □

Theorem 6.3.2. *The sentence φ_S returned by Algorithm 4 is a first-order sentence of minimum quantifier rank that is consistent with S.*

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

Theorem 6.3.3. *Given a sample S, Algorithm 4 returns φ_S in polynomial time in the size of S.*

Proof. First, the algorithm computes $max\{MinRound(\mathscr{A},\mathscr{B}) \mid \mathscr{A} \in P, \mathscr{B} \in N\}$ in order to use a suitable quantifier rank. This takes polynomial time because, for a given $\mathscr{A} \in P, \mathscr{B} \in N$, to compute $MinRound(\mathscr{A},\mathscr{B})$ takes polynomial time in the classes of structures we are considering in this work. Then, our algorithm loops over structures in the sample and, in each loop, it chooses a formula $\varphi \in \Phi^r_{\mathscr{A},\mathscr{B}}$. As the size of each $\varphi \in \Phi^r_{\mathscr{A},\mathscr{B}}$ is polynomial in the size of $\mathscr{A}$ and $\mathscr{B}$, and $|\Phi^r_{\mathscr{A},\mathscr{B}}|$ is polynomial in the size of S, the overall complexity of Algorithm 4 is polynomial time in the size of S. □

6.4 Concluding Remarks and Comparisons

Our algorithm is an improvement, with respect to computational complexity, over the work in (KAISER, 2012), for this particular problem on the classes of structures we are considering, i.e., MS, ES, DULO, and strings. We defined an algorithm that takes polynomial time in the size of the input sample. Furthermore, our algorithm returns a sentence of size polynomial in the size of the sample.

Regarding CNPL (STROTHER-GARCIA *et al.*, 2017), our work deals with the full expressive power of first-order logic over strings, whereas CNPL is less expressive than full first-order logic. Also, our methods find a first-order sentence from positive and negative strings, while the input of the method in (STROTHER-GARCIA *et al.*, 2017) is only a set of positive strings.

However, observe that the sentence returned by Algorithm 4 is a disjunction of $|P|$ conjunctions of distinguishability sentences. Then, our algorithm may return large sentences.

The algorithm in (KAISER, 2012) also returns formulas which are long and hard to read. Then, they greedily remove subformulas that are not necessary to distinguish P from N. Moreover, the method in (STROTHER-GARCIA *et al.*, 2017) also returns CNPL sentences which contain a large number of conjunctions. In Chapter 7, we show how to overcome this disadvantage of our method. We consider an approach in which the number of conjunctive clauses is also given.

Furthermore, since our algorithm and the one in (KAISER, 2012) return first-order sentences, some sentences are hard to read. For example, the sentence $\varphi_S = (\varphi_{q_1\geq 1} \wedge \varphi_{q_{\geq 2}<1}) \vee (\varphi_{q_{\geq 2}<1} \wedge \varphi_{q_{\geq 3}<1})$ returned by Algorithm 4 in Example 6.3.1 is an abbreviation for

$$\begin{aligned}\varphi_S =&(\exists x_1(\neg\exists y_1(y_1 < x_1) \wedge \neg\exists y_2(x_1 < y_2)) \wedge \neg\exists x_1\exists y_1(x_1 < y_1)) \vee \\ &(\neg\exists x_1\exists y_1(x_1 < y_1) \wedge \neg\exists x_1(\exists y_1(x_1 < y_1) \wedge \exists y_2(y_2 < x_1))).\end{aligned}$$

Moreover, the problem of deciding, given a structure $\mathscr{A}$ and a sentence φ, whether $\mathscr{A} \models \varphi$ is PSPACE-complete (STOCKMEYER, 1974; VARDI, 1982; GRÄDEL *et al.*, 2005). This problem is PSPACE-complete even when the structure $\mathscr{A}$ is fixed. Then, it is also PSPACE-complete for the classes of structures we are considering. In the following chapter, we define a quantifier-free disjunctive normal form for each class of structures we are considering. This normal form seems to be easier to read than general first-order sentences. Besides, we show that determining whether $\mathscr{A} \models \varphi$ takes polynomial time when φ is in this quantifier-free normal form.

7 SYNTHESIS OF QUANTIFIER-FREE SENTENCES IN DNF

In this chapter, we define a quantifier-free normal form (QNF) for first-order sentences over strings. The process is similar for the other classes of structures we are considering, i.e., MS, ES, and DULO. Recall that the standard vocabulary for strings is $\tau = \{S, (P_a)_{a\in\Sigma}\}$. QNF sentences are defined over a richer vocabulary such that atomic sentences are an abbreviation of distinguishability sentences over the vocabulary τ.

We also define a synthesis problem for QDNF sentences, i.e., QNF sentences in disjunctive normal form. We solve this problem by a translation to the SAT problem. Furthermore, we show that the synthesis of QDNF sentences is an NP-complete problem. These results presented in this chapter were published in (ROCHA *et al.*, 2018b) and submitted to (ROCHA *et al.*, 2020). In a first round of review, our paper (ROCHA *et al.*, 2020) received overall positive feedback.

We also contemplate a synthesis problem for l-QDNF sentences, i.e., QDNF sentences such that the number of literals per clause is at most l. Therefore, we define a synthesis problem in which the maximum number of literals per clause is given as input as well. We also use a SAT encoding to solve this problem. Moreover, we show that the synthesis of l-QDNF sentence is NP-complete as well. These results were accepted for publication in (ROCHA; MARTINS, 2019).

7.1 Quantifier-Free Normal Forms

In this section, we define a normal form for first-order logic over strings that considers the multiplicity and scattering of substrings, the occurrence of substrings as prefix or suffix, and the length of strings. We call this normal form as quantifier-free normal form (QNF). We define the atomic QNF sentences inspired by the distinguishability sentences of Chapter 6. This is important in order to show that every first-order sentence over strings can be converted into an equivalent QNF sentence. Sentences in QNF are first-order sentences over the vocabulary $\tau_{QNF} = \{lgeq, pref, suff, \mathscr{N}, \geq, \gamma, \sigma, (w)_{w\in\Sigma^*}, (n)_{n\in\mathbb{N}^*}\}$ such that $lgeq$, $pref$ and $suff$ are built-in unary predicates, $\mathscr{N}$ is a built-in binary predicate, $\geq$ is the built-in linear order relation, γ and σ are built-in unary functions, each $w \in \Sigma^*$ is a built-in constant, and each $n \in \mathbb{N}^*$ is a built-in constant. In what follows, we define the syntax and semantics of QNF sentences.

Definition 7.1.1 (Quantifier-Free Normal Form (QNF))**.** *Let $u, v_1, \ldots, v_k \in \Sigma^*$ be strings over an*

alphabet Σ *such that* $|v_i| \leq 2|w|+1$*, for* $i \in \{1,...,k\}$*, and* $n \in \mathbb{N}$*. The normal form is defined in BNF in the following way:*

$$\varphi := \begin{array}{l} pref(u) \mid suff(u) \mid \gamma(u) \geq n \mid \sigma(u) \geq n \mid \mathscr{N}(u,\{v_1,...,v_k\}) \mid lgeq(n) \mid \\ (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi). \end{array}$$

Parentheses are omitted whenever possible. Then, we allow sentences such as

$$(pref(ab) \vee \neg\gamma(ba) \geq 2) \wedge \sigma(abb) \geq 1 \wedge lgeq(4).$$

The semantics of QNF sentences is obtained from a string as in the definition below.

Definition 7.1.2 (QNF Semantics)**.** *Let* Σ *be a vocabulary,* $w,u,v_1,...,v_k \in \Sigma^*$*, and* $n \in \mathbb{N}$*. We define when a string* w *satisfies QNF sentence* φ*, written* $w \models \varphi$*, as:*

$$\begin{array}{lll} w \models pref(u) & iff & pref_{|u|}(w) = u; \\ w \models suff(u) & iff & suff_{|u|}(w) = u; \\ w \models lgeq(n) & iff & |w| \geq n; \\ w \models \gamma(u) \geq n & iff & \gamma(u,w) \geq n; \\ w \models \sigma(u) \geq n & iff & \sigma(u,w) \geq n; \\ w \models \mathscr{N}(u,\{v_1,...,v_k\}) & iff & \{v_1,...,v_k\} \subseteq \mathscr{N}(u,w); \\ w \models \neg\varphi & iff & w \not\models \varphi; \\ w \models \varphi_1 \wedge \varphi_2 & iff & w \models \varphi_1 \text{ and } w \models \varphi_2; \\ w \models \varphi_1 \vee \varphi_2 & iff & w \models \varphi_1 \text{ or } w \models \varphi_2. \end{array}$$

Based on this semantics, we can talk about the truth value of a QNF sentence such as $\varphi = (pref(ab) \vee \neg suff(bb)) \wedge \sigma(c) \geq 2$ in a given string w. For instance, $abcaacbb \models \varphi$. Then, as in the case of first-order formulas over strings, QNF sentences also define formal languages. We use the notation $L(\varphi)$ to represent the language defined by the QNF sentence φ. The following result is another advantage of QNF sentences over general first-order sentences.

Lemma 7.1.1. *Given a string* w *and a QNF sentence* φ*, one can check if* $w \models \varphi$ *in polynomial time.*

Proof. We need to show that, given a string w and an atomic QNF sentence φ, one 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 $\varphi = lgeq(n)$. It is possible to check in linear time in the

length of w whether $|w| \geq n$. Next, let $\varphi = pref(u)$. Clearly, it is also possible to check in linear time in the length of w whether $\text{pref}_{|u|}(w) = u$. The case in which $\varphi = suff(u)$ is analogous.

Now, let $\varphi = \gamma(u) \geq n$. At each position of w, it is necessary to check if u occurs starting on that position. This takes quadratic time in the length of w. Let $\varphi = \sigma(u) \geq n$. First, it is necessary to compute a $(|u|+1)$-segmentation of the set of starting positions of the occurrences of u in w. This is accomplished by checking if u occurs at each position. The processed positions are added to a given segment as long as their distance from the least position of the segment is less than or equal to $|u|+1$. A new segment is started when a position that does not meet such condition is found. Thus, this also takes quadratic time in $|w|$.

Finally, let $\varphi = \mathscr{N}(u, \{v_1, ..., v_k\})$. The procedure is similar to the one for $\varphi = \sigma(u) \geq n$. After computing the segments of a $(|u|+1)$-segmentation of $\Gamma(u,w)$, we compute $\mathscr{N}(u,w)$. It takes linear time time to compute $\mathscr{N}(u,w)$ from the segments. Therefore, since it takes quadratic time to compute the segmentations, it also takes quadratic time to determine $\mathscr{N}(u,w)$. Next, we check whether $\{v_1, ..., v_k\} \subseteq \mathscr{N}(u,w)$. Observe that $|v_i| \leq |w|$, for $i \in \{1, ..., k\}$, otherwise it is certain that $w \not\models \varphi$. As $\{v_1, ..., v_k\}$ and $\mathscr{N}(u,w)$ are multisets, for each $\beta \in \{v_1, ..., v_k\}$, we examine whether the number of occurrences of β in $\mathscr{N}(u,w)$ is at least the number of occurrences of β in $\{v_1, ..., v_k\}$. This task takes quadratic time. Therefore, the task to check whether $w \models \mathscr{N}(u, \{v_1, ..., v_k\})$ has complexity $O(|w|)^2)$. Therefore, if φ is a Boolean combination of atomic QNF sentences, then the overall complexity is $O((|\varphi| + |w|)^3)$. □

Now, we define the notion of QNF rank of QNF sentences. In the following definition, observe that the QNF rank of a QNF sentence is equal to the quantifier rank of the equivalent first-order sentence. The goal of this definition is to use the results of the Ehrenfeucht–Fraïssé game on strings with QNF sentences.

Definition 7.1.3 (QNF Rank)**.** *Let φ be a QNF sentence. The QNF rank of φ, written $qnfr(\varphi)$, is defined as*

$$qnfr(\varphi) := \begin{cases} \lceil log(n+2) \rceil, & \text{if } \varphi = lgeq(n) \\ \lceil log(|w|+2) \rceil, & \text{if } \varphi \in \{pref(w), suff(w)\} \\ \lceil log(|w|+1) \rceil + n - 1, & \text{if } \varphi \in \{\sigma(w) \geq n, \gamma(w) \geq n\} \\ \lceil log(|w|+1) \rceil + k, & \text{if } \varphi = \mathscr{N}(w, \{v_1, ..., v_k\}) \\ max\{qnfr(\psi) \mid \psi \in A(\varphi)\}, & \text{otherwise.} \end{cases}$$

For example, $qnfr(pref(ab) \vee \neg\gamma(ba) \geq 2) = max\{\lceil log(4) \rceil, \lceil log(3) \rceil + 1\} = 3$. Now, we show a result connecting the QNF rank of QNF sentences and the number of rounds in Ehrenfeu-

cht–Fraïssé games. First, we show that, over strings, for each QNF sentence, there exists an equivalent first-order sentence.

Lemma 7.1.2. *Let φ be a formula in QNF such that $qnfr(\varphi) \leq r$. Then, φ is equivalent to first-order sentence over strings ψ such that $qr(\psi) \leq r$.*

Proof. First, observe that atomic QNF sentences are equivalent to the following abbreviations defined in Subsection 6.2.4.

$$
\begin{aligned}
&lgeq(n) \text{ is equivalent to } \varphi_{\geq n},\\
&pref(a_1...a_k) \text{ is equivalent to } \varphi_{pref_k=a_1...a_k},\\
&suff(a_1...a_k) \text{ is equivalent to } \varphi_{suff_k=a_1...a_k},\\
&\mathscr{N}(u,\{v_1,...,v_k\}) \text{ is equivalent to } \varphi_{\mathscr{N}(u,\{v_1,...,v_k\})},\\
&\gamma(u) \geq n \text{ is equivalent to } \varphi_{\gamma(u)\geq n},\\
&\sigma(u) \geq n \text{ is equivalent to } \varphi_{\sigma(u)\geq n}.
\end{aligned}
$$

Moreover, for each atomic QNF sentence φ above, the equivalent first-order sentence φ' is such that $qnfr(\varphi) = qr(\varphi')$. We have shown first-order sentences equivalent to each atomic QNF sentence. As first-order logic is closed under Boolean connectives, and by the definition of quantifier rank, the lemma holds. □

Theorem 7.1.1. *The Spoiler has a winning strategy in $\mathscr{G}_r(u,v)$ iff there exists a QNF sentence φ such that $qnfr(\varphi) \leq r$, $u \models \varphi$ and $v \not\models \varphi$.*

Proof. Suppose that there exists a QNF sentence φ such that $qnfr(\varphi) \leq r$, $u \models \varphi$ and $v \not\models \varphi$. By Theorem 7.1.2, there exists a first-order sentence φ' equivalent to φ such that $qr(\varphi') \leq r$. Then, $u \models \varphi'$ and $v \not\models \varphi'$. From Theorem 2.3.1, it follows that the Spoiler has a winning strategy in $\mathscr{G}_r(u,v)$.

Now, assume that the Spoiler has a winning strategy in $\mathscr{G}_r(u,v)$. Then, one condition of Theorem 3.3.1 does not hold. First, assume that $|u| \neq |v|$ and, without loss of generality, $|u| < 2^r - 2$. Suppose that $|u| < |v|$. Then, $\varphi = \neg lgeq(|u|+1)$. Clearly, $qnfr(\varphi) = |u|+3$. Since $|u| \leq 2^r - 3$, then $qnfr(\varphi) \leq r$. Now, suppose that $|v| < |u|$. Then, we set $\varphi = lgeq(|v|+1)$. Since $|v| < |u| < 2^r - 2$, it follows that $qnfr(\varphi) \leq r$. In either case, $u \models \varphi$ and $v \not\models \varphi$.

Now, suppose that $pref_{2^r-2}(u) \neq pref_{2^r-2}(v)$. Clearly, $u \models \varphi$ and $v \not\models \varphi$, for $\varphi = pref(pref_{2^r-2}(u))$. Moreover, $qnfr(pref(pref_{2^r-2}(u))) \leq r$. The case such that $suff_{2^r-2}(u) \neq suff_{2^r-2}(v)$ is analogous.

Without loss of generality, assume that $\sigma(\alpha,u)+q_\alpha \leq r$ for some α such that $|\alpha| = 2^{q_\alpha}-1$ and $\sigma(\alpha,u) < \sigma(\alpha,v)$. Therefore, we set $\varphi = \neg(\sigma(\alpha) \geq \sigma(\alpha,u)+1)$. As $\sigma(\alpha,u)+q_\alpha \leq r$, then $qnfr(\varphi) \leq r$. Now, suppose that $\sigma(\alpha,u) > \sigma(\alpha,v)$. In this case, we set $\varphi = \sigma(\alpha) \geq \sigma(\alpha,v)+1$. Since $\sigma(\alpha,u) > \sigma(\alpha,v)$ and $\sigma(\alpha,u)+q_\alpha \leq r$, it follows that $\sigma(\alpha,v)+q_\alpha \leq r$. Therefore, $qnfr(\varphi) \leq r$. Clearly, in either case, $u \models \varphi$ and $v \not\models \varphi$.

Finally, assume that $\sigma(\alpha,u)+q_\alpha \leq r$, $\sigma(\alpha,u) = \sigma(\alpha,v)$, and $\gamma(\alpha,u) \neq \gamma(\alpha,v)$. We consider two cases. First, assume that $\gamma(\alpha,u)+q_\alpha, \gamma(\alpha,v)+q_\alpha \leq r$. In this case, we set $\varphi = \neg(\gamma(\alpha) \geq \gamma(\alpha,u)+1)$ or $\varphi = \gamma(\alpha) \geq \gamma(\alpha,v)+1$ analogously to the paragraph above. Otherwise, first assume that $\gamma(\alpha,u) > \gamma(\alpha,v)$. We set $\varphi = \mathscr{N}(\alpha,\mathscr{N}(\alpha,u))$. Clearly, $u \models \varphi$. Moreover, $v \not\models \varphi$ since $\sigma(\alpha,u) = \sigma(\alpha,v)$ and $\gamma(\alpha,u) > \gamma(\alpha,v)$. In the second case assume that $\gamma(\alpha,u) < \gamma(\alpha,v)$. Then, we set $\varphi = \neg\mathscr{N}(\alpha,\mathscr{N}(\alpha,v))$. Clearly, $v \models \neg\varphi$ and then $v \not\models \varphi$. As $\sigma(\alpha,u) = \sigma(\alpha,v)$ and $\gamma(\alpha,u) < \gamma(\alpha,v)$, $u \not\models \neg\varphi$. Therefore, $u \models \varphi$. Moreover, in all cases, $qnfr(\varphi) \leq r$. □

Now we show that every sentence in first-order logic over strings is equivalent to a QNF sentence. The idea is analogous to the proof of Theorem 6.2.4. First, we define the following abbreviations.

$$
\begin{aligned}
&leq(n) := lgeq(n) \wedge \neg lgeq(n+1),\\
&\gamma(u) = n := \gamma(u) \geq n \wedge \neg(\gamma(u) \geq n+1),\\
&\sigma(u) = n := \sigma(u) \geq n \wedge \neg(\sigma(u) \geq n+1).
\end{aligned}
$$

Now, we define formulas in QNF sentences equivalent to Hintikka formulas. As in condition 4 of Theorem 3.3.1, we only use strings α such that $|\alpha| = 2^{q_\alpha}-1$, for $q_\alpha > 0$.

$$
\varphi_w^{r,length} := \begin{cases} leq(|w|), & \text{if } |w| < 2^r - 2 \\ lgeq(2^r-2), & \text{otherwise.} \end{cases}
$$

$$
\varphi_w^{r,pref} := pref(pref_{2^r-2}(w))
$$

$$
\varphi_w^{r,suff} := suff(suff_{2^r-2}(w))
$$

$$
\varphi_w^{r,\alpha} := \begin{cases} (\sigma(\alpha) = \sigma(\alpha,w)) \wedge (\gamma(\alpha) = \gamma(\alpha,w)), & \text{if } q_\alpha + \sigma(\alpha,w) \leq r \\ \sigma(\alpha) \geq r - q_\alpha + 1, & \text{otherwise.} \end{cases}
$$

$$
\varphi_w^{r,sub} := \bigwedge \{ \varphi_w^{r,\alpha} \mid |\alpha| = 2^q - 1, q > 0 \}.
$$

Lemma 7.1.3. $\models \varphi_w^r \leftrightarrow (\varphi_w^{r,length} \wedge \varphi_w^{r,pref} \wedge \varphi_w^{r,suff} \wedge \varphi_w^{r,sub})$.

The proof of the lemma above is the same as the proof of Theorem 6.2.16. Observe that $qr(\varphi_w^r) \leq qr(\varphi_w^{r,length} \wedge \varphi_w^{r,pref} \wedge \varphi_w^{r,suff} \wedge \varphi_w^{r,sub})$ since we do not use QNF formulas of the form $\mathscr{N}(u,\{v_1,...,v_k\})$. However, this detail is not important because, in the following lemma, we are only interested in the equivalence. Then, we show that every first-order sentence over strings is equivalent to a QNF sentence.

Lemma 7.1.4. *Let φ be first-order sentence over strings. Then, φ is equivalent to a QNF sentence.*

Proof. Let $qr(\varphi) = r$. From Theorem 2.3.2, it follows that $\models \varphi \leftrightarrow \varphi_{u_1}^r \vee ... \vee \varphi_{u_s}^r$. By Lemma 7.1.3, each $\varphi_{u_i}^r$ is equivalent to a QNF sentence. Thus, φ is also equivalent to a QNF sentence. □

By Lemma 7.1.2 and Lemma 7.1.4, every first-order sentence over strings is equivalent to a QNF sentence and vice versa. We use QNF sentences in this work because they represent properties of strings in a more succinct way. As first-order logic over strings defines the class of LTT languages, we conclude this section with the following corollary.

Corollary 7.1.1. *The class of languages defined by QNF sentences is exactly the class LTT.*

7.2 Synthesis of QDNF Sentences

Now, we formally define the problem we are interested in. We also show that this problem is hard, and we define an algorithm to solve it. First, a disjunctive normal form is defined for QNF sentences (QDNF) in the same way as classical propositional logic. Then, a literal is an atomic QNF sentence or the negation of an atomic QNF sentence. A literal is positive if it is an atomic QNF sentence. A literal is negative if it is the negation of an atomic QNF sentence. Moreover, a conjunctive clause is a conjunction of literals. Then, a QDNF sentence is a disjunction of conjunctive clauses. For example, $(pref(ab) \wedge \neg suff(ba)) \vee \neg\gamma(ab) \geq 1$ is a QNF sentence with 2 conjunctive clauses. Then, $pref(ab)$ is a positive literal and $\neg suff(ba)$ is a negative literal.

For a QNF sentence φ, we also define $A(\varphi)$ as the set of atomic sentences occurring in φ. For example, for $\varphi = (pref(ab) \vee \neg\gamma(ba) \geq 2) \wedge \sigma(abb) \geq 1, A(\varphi) = \{pref(ab), \gamma(ba) \geq 2, \sigma(abb) \geq 1\}$.

A sample of strings $S = (P,N)$ 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. A QNF sentence φ covers a positive string $u \in P$ if $u \models \varphi$. A QNF sentence φ covers a negative string $v \in N$ if $v \not\models \varphi$. A QNF 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.

Definition 7.2.1 (QDNF Synthesis (QDNFS))**.** *Given a sample of strings S, a set* Φ *of atomic QNF sentences, and a positive integer m, the QDNFS problem consists in finding, if it exists, a QDNF sentence* φ *such that*

- $A(\varphi) \subseteq \Phi$,
- φ *has m conjunctive clauses,*
- φ *is consistent with S.*

The following example illustrates the QDNF Synthesis problem.

Example 7.2.1. *Let S be the sample in Table 1 repeated below for convenience. Let* $\Phi = \{suff(e), pref(s), \sigma(v) \geq 2, \gamma(p) \geq 1\}$ *be a set of atomic QNF sentences, and* $m = 1$. *A solution to the QDNFS problem is the formula below.*

$$pref(s) \wedge \neg \gamma(p) \geq 1.$$

Table 1 – A sample of strings.

String	Class
stviil	Positive
ktvive	Negative
stviie	Positive
stpiie	Negative

Source: Own elaboration

Now, we consider a relaxed version of the QDNFS problem without the constraints on the number of conjunctive clauses and set Φ. In the QDNFS problem, the set Φ of atomic QNF sentences is also given as input. Clearly, if Φ is not a suitable set, then, for all positive integer m, there does not exist φ consistent with S such that φ has m conjunctive clauses. For example, let S be a sample such that $P = \{abbbb\}$ and $N = \{abbba\}$, and let $\Phi = \{pref(ab)\}$. Then, for all positive integer m, there does not exist φ consistent with S such that $A(\varphi) \subseteq \Phi$ and φ has m conjunctive clauses.

7.2.1 Suitable Set of Formulas

Now, we show how to consider a suitable set Φ. Let S be a sample. We define $\text{Pref}(S) = \{x \mid xy \in P \cup N\}$ as the set of all prefixes of all strings in S. Analogously, we define $\text{Suff}(S)$ for suffixes. Moreover, we define $\text{Inf}(S) = \{y \mid xyz \in P \cup N\}$ as the set of all infixes of all strings in a sample S. Now, we define the following set, given a sample S and QNF rank r.

$$\begin{aligned}
\Phi_S^r := \; & \{lgeq(n) \mid n \leq max\{|w| \mid w \in P \cup N\}, n \leq 2^r - 2\} \cup \\
& \{pref(u) \mid u \in \text{Pref}(S), |u| \leq 2^r - 2\} \cup \\
& \{suff(u) \mid u \in \text{Suff}(S), |u| \leq 2^r - 2\} \cup \\
& \{\gamma(u) \geq n \mid u \in \text{Inf}(S), n \geq 1, \lceil log_2(|u|+1) \rceil + n - 1 \leq r\} \cup \\
& \{\sigma(u) \geq n \mid u \in \text{Inf}(S), n \geq 1, \lceil log_2(|u|+1) \rceil + n - 1 \leq r\} \cup \\
& \{\mathscr{N}(u, \mathscr{N}(u,w)) \mid u \in \text{Inf}(S), w \in P \cup N, \lceil log_2(|u|+1) \rceil + |\mathscr{N}(u,w)| \leq r\}.
\end{aligned}$$

The set Φ_S^r consists of atomic QNF sentences such that the QNF rank is at most r. Furthermore, Φ_S^r can be built in polynomial time.

Proposition 7.2.1. *Let $\varphi \in \Phi_S^r$. Then, $qnfr(\varphi) \leq r$.*

Proposition 7.2.2. *Let S be a sample and r be a positive integer. Then, $|\Phi_S^r|$ is polynomial in the size of S. Moreover, Φ_S^r can be built in polynomial time.*

Now, we show that $\Phi_S^{max\{MinRound(u,v) \mid u \in P, v \in N\}}$ is a suitable set of atomic sentences in the QDNFS problem. Therefore, we show that there exists φ consistent with S such that $A(\varphi) \subseteq \Phi_S^{max\{MinRound(u,v) \mid u \in P, v \in N\}}$. First, we need the following result.

Lemma 7.2.1. *Let $u \in P$ be a positive string, $v \in N$ be a negative string, and r be a positive integer such that $MinRound(u,v) \leq r$. Then, there exists $\psi_{u,v}^r \in \Phi_S^r$ and $\varphi_{u,v}^r$ such that $\varphi_{u,v}^r = \psi_{u,v}^r$ or $\varphi_{u,v}^r = \neg\psi_{u,v}^r$ and $u \models \varphi_{u,v}^r$ and $v \not\models \varphi_{u,v}^r$.*

Proof. Since $MinRound(u,v) \leq r$, first, suppose that $pref_{2^r-2}(u) \neq pref_{2^r-2}(v)$. Note that the atomic sentence $pref(pref_{2^r-2}(u)) \in \Phi_S^r$. Then, $\varphi_{u,v}^r = pref(pref_{2^r-2}(u))$. The case such that $suff_{2^r-2}(u) \neq suff_{2^r-2}(u)$ is analogous.

Now, without loss of generality, suppose that $|u| \neq |v|$ and $|u| < 2^r - 2$. There are two cases two consider. First, assume that $|u| < |v|$. Then, $\varphi_{u,v}^r = \neg lgeq(|u|+1)$. Note that $|u|+1 \leq 2^r - 2$ and $lgeq(|u|+1) \in \Phi_S^r$. Now, suppose that $|v| < |u|$. Then, $\varphi_{u,v}^r = lgeq(|u|)$ which is also in Φ_S^r since $|u| \leq 2^r - 2$.

Assume now that $\sigma(\alpha,u)+q_\alpha \leq r$ for some α such that $|\alpha| = 2^{q_\alpha}-1$ and $\sigma(\alpha,u) \neq \sigma(\alpha,v)$. First, suppose that $\sigma(\alpha,u) < \sigma(\alpha,v)$. Therefore, $\varphi^r_{u,v} = \neg(\sigma(\alpha) \geq \sigma(\alpha,u)+1)$. As $\sigma(\alpha,u)+q_\alpha \leq r$, then $(\sigma(\alpha) \geq \sigma(\alpha,u)+1) \in \Phi^r_S$. For the second case, assume that $\sigma(\alpha,u) > \sigma(\alpha,v)$. Then, we set $\varphi^r_{u,v} = \sigma(\alpha) \geq \sigma(\alpha,u)$ which clearly is in Φ^r_S.

Lastly, suppose that $\sigma(\alpha,u)+q_\alpha \leq r$ for some α such that $|\alpha| = 2^{q_\alpha}-1$, $\sigma(\alpha,u) = \sigma(\alpha,v)$, and $\gamma(\alpha,u) \neq \gamma(\alpha,v)$. First, assume that $\gamma(\alpha,u) > \gamma(\alpha,v)$. Then, $\varphi^r_{u,v} = \mathscr{N}(\alpha,\mathscr{N}(\alpha,u))$. Observe that $\varphi^r_{u,v} \in \Phi^r_S$ since $|\mathscr{N}(\alpha,u)| = \sigma(\alpha,u)$ and $\sigma(\alpha,u)+q_\alpha \leq r$. Clearly, $u \models \varphi^r_{u,v}$. Since $\sigma(\alpha,u) = \sigma(\alpha,v)$ and $\gamma(\alpha,u) > \gamma(\alpha,v)$, there exists $\beta \in \mathscr{N}(\alpha,u)$ such that the number of occurrences of β in $\mathscr{N}(\alpha,u)$ is greater than the number of occurrences of $\beta \in \mathscr{N}(\alpha,v)$. Therefore, $\mathscr{N}(\alpha,u) \not\subseteq \mathscr{N}(\alpha,v)$. Then, $v \not\models \varphi^r_{u,v}$. For the second case, assume that $\gamma(\alpha,u) < \gamma(\alpha,v)$. In this case, $\varphi^r_{u,v} = \neg\mathscr{N}(\alpha,v)$. The rest is analogous to the case where $\gamma(\alpha,u) > \gamma(\alpha,v)$. □

Now, we define a formula such that its atoms are among those in $\Phi_S^{max\{MinRound(u,v)|u\in P,v\in N\}}$, and it is consistent with S and its atoms are among those in $\Phi_S^{max\{MinRound(u,v)|u\in P,v\in N\}}$. In the following, sentences $\varphi_{u,v}^{max\{MinRound(u,v)|u\in P,v\in N\}}$ are defined as in Lemma 7.2.1. Clearly, for strings $u \in P$ and $v \in N$, it follows that $MinRound(u,v) \leq max\{MinRound(u,v) \mid u \in P,\ v \in N\}$. Given a sample S, we define the following formula.

$$\varphi_S = \bigvee_{u\in P} \bigwedge_{v\in N} \varphi_{u,v}^{max\{MinRound(u,v)|u\in P,v\in N\}}.$$

Theorem 7.2.1. *Let S be a sample. Then, φ_S is consistent with S.*

Proof. Let $u \in P$. Then, as $MinRound(u,v) \leq max\{MinRound(u,v) \mid u \in P, v \in N\}$, for all $v \in N$, by Lemma 7.2.1, it follows that

$$u \models \bigwedge_{v\in N} \varphi_{u,v}^{max\{MinRound(u,v)|u\in P,v\in N\}}.$$

Therefore, $u \models \varphi_S$. Now, let $v \in N$ and assume that $v \models \varphi_S$, i.e, $v \models \bigwedge_{v'\in N} \varphi_{u,v'}^{max\{MinRound(u,v')|u\in P,v'\in N\}}$, for some $u \in P$. Then,

$$v \models \varphi_{u,v}^{max\{MinRound(u,v')|u\in P,v'\in N\}}.$$

However, by Lemma 7.2.1, $v \not\models \varphi_{u,v}^{max\{MinRound(u,v')|u\in P,v'\in N\}}$. This is an absurd. Therefore, $v \not\models \varphi_S$. □

7.2.2 A SAT-Based Approach

Observe that φ_S above has $|P|$ disjunctive clauses. In the QDNFS problem, the number of conjunctive clauses is given. Now, we show how to solve the QDNFS problem by a translation to the SAT problem. This approach is based on the one for the BFS problem (KAMATH *et al.*, 1992; IGNATIEV *et al.*, 2018). Since our solution is based on SAT, we call our method QDNFSAT. The variables used in the propositional representation are $p_{j,\varphi}$ and $p'_{j,\varphi}$, for $j \in \{1,...,m\}$, $\varphi \in \Phi$, and $c_{j,u}$, for $j \in \{1,...,m\}$, $u \in P$. The propositional variable $p_{j,\varphi}$ is true iff φ is not included in conjunctive clause j. Analogously, $p'_{j,\varphi}$ is true if $\neg\varphi$ is not included in clause j. Finally, $c_{j,u}$ is true iff conjunctive clause j is true in $u \in P$. Now, we show the constraints. The first constraint ensures that, for each clause C_j and each $\varphi \in \Phi$, φ and $\neg\varphi$ are not both in C_j.

$$p_{j,\varphi} \vee p'_{j,\varphi} \qquad \text{for } j \in \{1,...,m\},\ \varphi \in \Phi. \tag{7.1}$$

For the next constraint, given $v \in N$ a negative string, we define $P_v = \{\varphi \in \Phi \mid v \models \varphi\}$ and $N_v = \{\varphi \in \Phi \mid v \not\models \varphi\}$. In the second constraint, each negative string does not satisfy any clause.

$$\bigvee_{\varphi \in P_v} \neg p'_{j,\varphi} \vee \bigvee_{\varphi \in N_v} p_{j,\varphi} \qquad \text{for } j \in \{1,...,m\}, v \in N. \tag{7.2}$$

For $j \in \{1,...,m\}$, $\varphi \in \Phi$, and $u \in P$, the following constraint describes when clause j does not hold in u.

$$\begin{cases} p'_{j,\varphi} \vee \neg c_{j,u}, & \text{if } u \models \varphi, \\ p_{j,\varphi} \vee \neg c_{j,u}, & \text{otherwise.} \end{cases} \tag{7.3}$$

Finally, the last constraint guarantees that each positive string must be satisfied by at least one conjunctive clause.

$$\bigvee_{j \in \{1,...,m\}} c_{j,u} \qquad \text{for } u \in P. \tag{7.4}$$

Let $\psi_{S,\Phi,m}$ be the conjunction of propositional formulas 7.1 to 7.4 where S is a sample, Φ is a set of atomic QNF sentences, and m is the number of conjunctive clauses. Then, $\psi_{S,\Phi,m}$ has $O(m \times |\Phi| + m \times |P \cup N|)$ variables and $O(m \times |\Phi| \times |P \cup N|)$ clauses. In addition, if $\psi_{S,\Phi,m}$ is satisfiable, one can obtain a QDNF formula $\varphi_{\mathscr{V}}$ consistent with S such that $A(\varphi_{\mathscr{V}}) \subseteq \Phi$ and $\varphi_{\mathscr{V}}$ has m conjunctive clauses directly from a model $\mathscr{V}$ of $\psi_{S,\Phi,m}$.

Definition 7.2.2. *Let $\mathscr{V}$ be a model of $\psi_{S,\Phi,m}$. We define $\varphi_{\mathscr{V}}$ by*

$$\bigvee_{j=1}^{m} \bigwedge (\{\varphi \mid \varphi \in \Phi, \mathscr{V} \not\models p_{j,\varphi}\} \cup \{\neg\varphi \mid \varphi \in \Phi, \mathscr{V} \not\models p'_{j,\varphi}\}).$$

The next result states that our approach QDNFSAT produces a solution to the QDNFS problem.

Theorem 7.2.2. *For a sample S, set Φ of atomic QNF sentences, and a positive integer m, a model $\mathscr{V}$ of $\psi_{S,\Phi,m}$ provides a QDNF formula $\varphi_{\mathscr{V}}$ that is a solution to the QDNFS problem.*

Proof. Clearly, $\varphi_{\mathscr{V}}$ is well defined because it consists of at most m conjunctive clauses such that each clause is a conjunction of atomic sentences in Φ and negation of atomic sentences in Φ. Furthermore, Formula 7.1 makes sure that at most one of $\neg p_{j,\varphi}$ and $\neg p'_{j,\varphi}$ is set to true. Then, it is not possible to occur φ and $\neg\varphi$ at the same clause. Otherwise, such a clause would be unsatisfiable.

Now, we show that $\varphi_{\mathscr{V}}$ is consistent with S. Let $u \in P$. Then, Formula 7.4 ensures that there exists $j \in \{1,...,n\}$ such that $\mathscr{V} \models c_{j,u}$. Formula 7.3 enforces that if $\mathscr{V} \models \neg p_{j,\varphi}$, then $u \models \varphi$ and if $\mathscr{V} \models \neg p'_{j,\varphi}$, then $u \models \neg\varphi$. Therefore, $u \models \varphi_{\mathscr{V}}$.

Now, let $v \in N$. Assume that $v \models \varphi_{\mathscr{V}}$. Then, v satisfies the jth clause for some $j \in \{1,...,m\}$. Then, by Formula 7.2, $\mathscr{V} \models \neg p'_{j,\varphi}$ for some $\varphi \in P_v$ or $\mathscr{V} \models \neg p_{j,\varphi}$ for some $\varphi \in N_v$. In either case, it follows an absurd because the jth clause does not hold in v. Then, $v \not\models \varphi_{\mathscr{V}}$. Therefore, $\varphi_{\mathscr{V}}$ is consistent with S. □

7.2.3 NP-completeness

Now, we show that the QDNFS problem is NP-complete. First, we show the complexity of deciding if a given formula is a solution to the QDNFS problem.

Lemma 7.2.2. *Given a sample S, a set Φ of atomic QNF sentences, a natural m, and a QDNF formula φ, one can check if φ is a solution of the QDNFS problem in polynomial time.*

Proof. It is straightforward to check whether φ has at most m conjunctive clauses and $A(\varphi) \subseteq \Phi$ in polynomial time. From Lemma 7.1.1, it follows directly that one can check whether φ is consistent with S in polynomial time. □

Theorem 7.2.3. *QDNFS is NP-complete.*

Proof. From Lemma 7.2.2, it follows that QDNFS is in NP. Now, we show a polynomial time reduction from BFS to QDNFS.

The reduction maps an instance (Ψ, E, m) of BFS into an instance (Φ, S, m) of QDNFS as follows. First, we assume an order $p_1, ..., p_r$ in Ψ. We associate each $p_i \in \Psi$ to symbols $p_{i0}, p_{i1} \in \Sigma$. We define a string $w_{\mathscr{V}} = w_{\mathscr{V}_1} ... w_{\mathscr{V}_r} \in S$, for each valuation in $\mathscr{V} \in E^+ \cup E^-$. Let $\mathscr{V} \in E^+ \cup E^-$ be a valuation and $p_i \in \Psi$ be a propositional variable. Then, if $\mathscr{V} \not\models p_i$, we set $w_{v_i} = p_{i0}$; if $\mathscr{V} \models p_i$, we set $w_{v_i} = p_{i1}$. The set Φ is defined as $\{\gamma(p_{i1}) \geq 1 \mid p_i \in \Phi\}$. Note that $\mathscr{V} \models p_i$ iff $w_{\mathscr{V}} \models \gamma(p_{i1}) \geq 1$.

Let (Ψ, E, m) be an instance of BFS such that ψ is consistent with E and ψ has m clauses. Let (Φ, S, m) be as in the reduction. We build a QDNF formula φ by replacing each p_i occurring in ψ by $\gamma(p_{i1}) \geq 1$. Therefore, $\mathscr{V} \models \psi$ iff $w_{\mathscr{V}} \models \varphi$. Thus, φ has m clauses and it is consistent with S.

Now, let (Ψ, E, m) be an instance that has no formula with m clauses and consistent with E, and (Φ, S, m) be as in the reduction. Suppose that there exists φ with m clauses and consistent with S. We can easily build a formula ψ by replacing each $\gamma(p_{i1}) \geq 1$ in φ by p_i. Therefore, ψ is consistent with E and it has m clauses. This is an absurd. □

A disadvantage of the method defined in this section is that it may return a QNF sentence with a large number of literals per clause. In order to overcome this problem, we suggest a method which also considers the maximum number of literals per clause in the next section.

7.3 Synthesis of l-QDNF Sentences

In this section, since we consider the number of literals per conjunctive clause, we say that a QDNF formula with at most l literals per clause is a l-QDNF formula. For example, $(pref(ab) \wedge \neg suff(ba)) \vee \neg\gamma(ab) \geq 1$ is a 2-QDNF sentence.

Then, besides the number of conjunctive clauses, we can also control the maximum number of literals per clause. Sentences with few clauses and with few literals per clause seem more natural to be understood by humans than sentences with few clauses and a large number of literals per clause. Then, our method defined in this section is able to obtain a compact set of rules from the input sample. In the following, we formally define the problem.

Definition 7.3.1 (l-QDNF Synthesis (l-QDNFS))**.** *Given a sample of strings S, a set* Φ *of atomic QNF formulas, two positive integers m and l, the l-QDNFS problem consists in finding, if it exists, a QDNF formula* φ *such that*

- $A(\varphi) \subseteq \Phi$,
- *φ has m conjunctive clauses,*
- *each conjunctive clause in φ has at most l literals,*
- *φ is consistent with S.*

In the following, we give an example of instance and solution to the above problem.

Example 7.3.1. *Let S be the sample in Table 3 repeated below for convenience. Let Φ below be the set of atomic QNF sentences, $m = 2$, and $l = 2$.*

$$\Phi = \{pref(\grave{H}\grave{L}), \gamma(\grave{L}) \geq 1, suff(\acute{L}), suff(\acute{H}), \sigma(H) \geq 1, \gamma(L\grave{L}) \geq 1\}$$

A solution to the l-QDNFS problem is the formula φ below.

$$\varphi = (\neg\sigma(H) \geq 1 \wedge suff(\acute{L})) \vee (suff(\acute{H}) \wedge \neg\gamma(L\grave{L}) \geq 1).$$

Table 3 – Sample of stress patterns in Cambodian.

String	Class
$\grave{H}\grave{L}\grave{H}\acute{L}$	Positive
$\grave{H}LH\acute{L}$	Negative
$\grave{H}L\grave{H}L\acute{H}$	Positive
$\grave{H}L\grave{L}\grave{H}\acute{H}$	Negative
$\grave{H}L\grave{H}L$	Negative

Source: Own elaboration

Again, it is easy to solve a relaxed version of the l-QDNFS problem without the constraints on the number of conjunctive clauses, the maximum number of literals per clause, and set Φ. We can easily build a QDNF formula consistent with the input sample S using only the atomic QNF sentences in $\Phi_S^{max\{MinRound(u,v)|u\in P, v\in N\}}$. In what follows, we show our method to solve the l-QDNFS problem.

7.3.1 A SAT Encoding for l-QDNF Synthesis

Now, we show how to solve the synthesis problem by encoding this problem into the satisfiability problem of propositional logic. Since our solution is based on SAT, we call our method l-QDNFSAT (l-QDNF Synthesis using SAT). Formally, given S, Φ, m, l, we construct a propositional formula $\psi_{S,\Phi}^{m,l}$ of size polynomial such that $\psi_{S,\Phi}^{m,l,K}$ is satisfiable iff there exists

an l-QDNF formula φ consistent with S with m conjunctive clauses, and $A(\varphi) \subseteq \Phi$. Also, it is possible to obtain such a l-QDNF formula φ from a model of $\psi^{m,l}_{S,\Phi}$.

We define $\psi^{m,l}_{S,\Phi}$ to encode the structure of an l-QDNF formula with m conjunctive clauses. In order to encode this structure, we use the following propositional variables: $x_{i,j,t}$, for $i \in \{1,...,m\}$, $j \in \{1,...,l\}$, $t \in \Phi \cup \{*\}$, and $p_{i,j}$, for $i \in \{1,...,m\}$, $j \in \{1,...,l\}$. The meaning of a variable of the form $x_{i,j,t}$, for $t \in \Phi$, is that if it is set to true, then the associated atomic to the jth literal in clause i is t. Therefore, if $p_{i,j}$ is set to true, then atomic t occurs as a positive literal in clause i. Otherwise, it occurs as a negative literal. On the other hand, if $x_{i,j,*}$ is set to true, then the jth literal is skipped in clause i. Therefore, we ignore $p_{i,j}$ in this case.

To make the variables to encode a structure of an l-QDNF formula, we impose the following constraints expressing that at most one atomic QNF formula is associated to literal j in clause i, for $i \in \{1,...,m\}$ and $j \in \{1,...,l\}$. Formulas 7.5-7.6 enforce this property by allowing to set only one $x_{i,j,t}$ to true for literal j in clause i. Formula 7.7 guarantees that each clause has at least one literal.

$$\bigwedge_{i \in \{1,...,m\}} \bigwedge_{j \in \{1,...,l\}} \bigvee_{t \in \Phi \cup \{*\}} x_{i,j,t}. \tag{7.5}$$

$$\bigwedge_{i \in \{1,...,m\}} \bigwedge_{j \in \{1,...,l\}} \bigwedge_{t,t' \in \Phi \cup \{*\}, t \neq t'} \neg x_{i,j,t} \vee \neg x_{i,j,t'}. \tag{7.6}$$

$$\bigwedge_{i \in \{1,...,m\}} \bigvee_{j \in \{1,...,l\}} \neg x_{i,j,*}. \tag{7.7}$$

Now, we consider the constraints related to the sample S. We add variables to track the semantics of literals and clauses for each string in the sample. This idea is based on an encoding to the problem of learning linear temporal logic formulas presented in (NEIDER; GAVRAN, 2018). Then, we use the following types of propositional variables: $y^w_{i,j}$, for $i \in \{1,...,m\}$, $j \in \{1,...,l\}$, $w \in P \cup N$, and z^w_i, for $i \in \{1,...,m\}$, $w \in P \cup N$. Intuitively, the meaning is that if $y^w_{i,j}$ is set to true, then w satisfies literal j in clause i. In addition, if z^w_i is set to true, then clause i holds in string w.

First, we define the constraints to enforce the meaning of variables of the form $y^w_{i,j}$. Clearly, the semantics of a literal in a string w depends on the atomic QNF formula associated to literal j in clause i. Then, let $d^{w,t}_{i,j} = y^w_{i,j}$ if $w \models t$ and $d^{w,t}_{i,j} = \neg y^w_{i,j}$ if $w \not\models t$, for $t \in \Phi$. Also, the semantics relies upon the propositional variable $p_{i,j}$. Then, Formula 7.8 ensures that $y^w_{i,j}$ is set to

true iff w satisfies the jth literal in clause i. This is ensured by using the semantics of atomic QNF formulas and variables $p_{i,j}$. Formula 7.9 guarantees that the semantics of conjunctive clauses is not interfered by skipped literals. Finally, Formula 7.10 implements the semantics of conjunctive clauses. Clearly, z_i^w must be set to true if all literals in clause i are satisfied by w.

$$\bigwedge_{i\in\{1,\ldots,m\}}\bigwedge_{j\in\{1,\ldots,l\}}\bigwedge_{t\in\Phi}\bigwedge_{w\in P\cup N} x_{i,j,t}\rightarrow(p_{i,j}\leftrightarrow d_{i,j}^{w,t}). \tag{7.8}$$

$$\bigwedge_{i\in\{1,\ldots,m\}}\bigwedge_{j\in\{1,\ldots,l\}}\bigwedge_{w\in P\cup N} x_{i,j,*}\rightarrow y_{i,j}^{w}. \tag{7.9}$$

$$\bigwedge_{i\in\{1,\ldots,m\}}\bigwedge_{w\in P\cup N} z_i^w\leftrightarrow\bigwedge_{j=1}^{l} y_{i,j}^{w}. \tag{7.10}$$

In this noiseless case, the following constraints impose that a QNF formula must be consistent with the sample S.

$$\bigwedge_{u\in P}\bigvee_{i=1}^{m} z_i^u. \tag{7.11}$$

$$\bigwedge_{v\in N}\bigwedge_{i=1}^{m}\neg z_i^v. \tag{7.12}$$

Formulas 7.5-7.12 can easily be translated to CNF. Let $\psi_{S,\Phi}^{m,l}$ be the conjunction of Formulas 7.5-7.12 in CNF. Then, $\psi_{S,\Phi}^{m,l}$ has $O(m\times l\times|\Phi|+m\times l\times|P\cup N|)$ Boolean variables, and it consists of $O(m\times l\times|\Phi|^2+m\times l\times|\Phi|\times|P\cup N|)$ clauses.

Now, we show how to obtain a QDNF formula from a solution of SAT. One can obtain a l-QDNF formula $\varphi_{\mathscr{V}}$ from a model $\mathscr{V}$ of $\psi_{S,\Phi}^{m,l}$, if $\psi_{S,\Phi}^{m,l}$ is satisfiable. Clearly, we only need the truth values of variables $x_{i,j,t}$ and $p_{i,j}$.

Definition 7.3.2. *Let $\mathscr{V}$ be a model of $\psi_{S,\Phi}^{m,l}$. The l-QDNF formula is defined by*

$$\varphi_{\mathscr{V}}:=\textstyle\bigvee_{i=1}^{m} C_i\text{, such that each } C_i \text{ is defined by}$$

$$C_i:=\begin{array}{l}\bigwedge\{t\mid\mathscr{V}\models x_{i,j,t},\mathscr{V}\models p_{i,j},t\neq *,j\in\{1,\ldots,l\}\}\cup\\ \bigwedge\{\neg t\mid\mathscr{V}\models x_{i,j,t},\mathscr{V}\not\models p_{i,j},t\neq *,j\in\{1,\ldots,l\}\}.\end{array}$$

We finish this subsection by showing that our approach l-QDNFSAT produces a solution to the l-QDNFS problem.

Theorem 7.3.1. *For a sample S, a set of atomic QNF formulas Φ, positive integers m, l, a model $\mathscr{V}$ of $\psi^{m,l}_{S,\Phi}$ provides a l-QDNF formula $\varphi_{\mathscr{V}}$ that is a solution to the l-QDNFS problem.*

Proof. Clearly, $\varphi_{\mathscr{V}}$ is well defined because it consists of m conjunctive clauses such that each clause is a conjunction of at most l literals. Formula 7.7 ensures that each clause has at least one literal. Moreover, since each atomic is in Φ, $A(\varphi_{\mathscr{V}}) \subseteq \Phi$.

Let $u \in P$. By Formula 7.11, z^u_i is true for some $i \in \{1,...,m\}$. Then, from Formula 7.10, it follows that $y^u_{i,j}$ is true for all $j \in \{1,...,l\}$. Let j be any element in $\{1,...,l\}$. Formulas 7.5-7.7 force that $x_{i,j,t}$ is true for some atomic QNF sentence $t \in \Phi$ or $x_{i,j,*}$ is true. We just need to consider the first case. If $d^{u,t}_{i,j} = y^u_{i,j}$ in Formula 7.8, then $p_{i,j}$ must be true. Then, u holds in literal t in clause C_i. If $d^{u,t}_{i,j} = \neg y^u_{i,j}$ in Formula 7.8, then $p_{i,j}$ must be false. Then, u holds in literal $\neg t$ in clause C_i. Since j is arbitrary, then $u \models C_i$. Therefore, $u \models \varphi_{\mathscr{V}}$.
Now, let $v \in N$. From Formula 7.12, it follows that z^v_i is false for all $i \in \{1,...,m\}$. Let i be an arbitrary element in $\{1,...,m\}$. Then, by Formula 7.10, $y^v_{i,j}$ is false for some $j \in \{1,...,l\}$. Then, $x_{i,j,*}$ must be false by Formula 7.9. Therefore, $x_{i,j,t}$ is true for some $t \in \Phi$. There are two cases. First, assume that $v \models t$. Then, Formula 7.8 imposes that $p_{i,j}$ must be false. Then, v does not hold in the jth literal in C_i. Now, suppose that $v \not\models t$. Then, $p_{i,j}$ must be true. Again, v does not hold in the jth literal in C_i. As the clauses are conjunctive $v \not\models C_i$. Since i is arbitrary, $v \not\models \varphi_{\mathscr{V}}$. Therefore, $\varphi_{\mathscr{V}}$ is consistent with S. □

7.3.2 NP-completeness of l-QDNFS

Since the l-QDNFS problem is an extension of the QDNFS problem, we show our result on the hardness of the l-QDNFS problem.

Theorem 7.3.2. *l-QDNFS is an NP-complete problem.*

Proof. Since the size $|S|$ of a sample S is the sum of the lengths of all strings it includes, the construction of $\psi^{m,l}_{S,\Phi}$ takes polynomial time. Then, this construction is a polynomial time reduction from l-QDNFS to SAT. Since SAT is an NP problem, it follows that l-QDNFS is in NP.

Now, we show a polynomial time reduction from QDNFS to l-QDNFS. The reduction maps an instance (S', Φ', m') of QDNFS into an instance (S, Φ, m, l) of l-QDNFS as follows. We set $S = S'$, $\Phi = \Phi'$, $m = m'$, $l = |\Phi|$.

Let (S', Φ', m') be an instance of QDNFS such that φ' is a QDNF formula consistent with S' such that it has m' conjunctive clauses and $A(\varphi') \subseteq \Phi'$. Since l-QDNFS is an extension of

QDNFS, then φ' also has at most $|\Phi|$ literals per clause, otherwise such a clause would certainly be unsatisfiable.

Now, let (S', Φ', m') be an instance of QDNFS such that it does not have solution. We show by contradiction that l-QDNFS has no solution either. Assume that there exists an l-QDNF formula φ consistent with S such that it has m conjunctive clauses and $A(\varphi) \subseteq \Phi$. In other words, φ has at most $|\Phi|$ literals per clause. Then, φ is a QDNF formula consistent with S such that it has m clauses and $A(\varphi) \subseteq \Phi'$. This is an absurd with the assumption, and so the claim holds. Therefore, l-QDNFS is NP-complete. □

7.4 Examples

In this section, we give examples of solutions obtained by DFASAT (HEULE; VERWER, 2010) for DFA synthesis and our methods QDNFSAT and l-QDNFSAT for synthesis of QNF formulas. We use samples from the domain of phonology.

In phonology, stress is a relative emphasis given to a certain syllable in a word. Usually, there are two syllable weights: light (L) and heavy (H). We use acute accent to denote primary stress level and grave accent to denote secondary stress level. Then, $\acute{H}$ represents a heavy syllable with primary stress and $\grave{L}$ denotes a light syllable with secondary stress. Furthermore, $L\grave{H}\acute{L}$ represents a word consisting of an unstressed light syllable, followed by a heavy syllable with secondary stress, followed by a light syllable with primary stress. For details, see (STROTHER-GARCIA *et al.*, 2017).

We define samples of stress patterns in the human languages Alawa and Cambodian. We define these samples using constraints presented in (LAMBERT; ROGERS, 2019) which define stress patterns in these natural languages.

7.4.1 Stress Patterns in Alawa

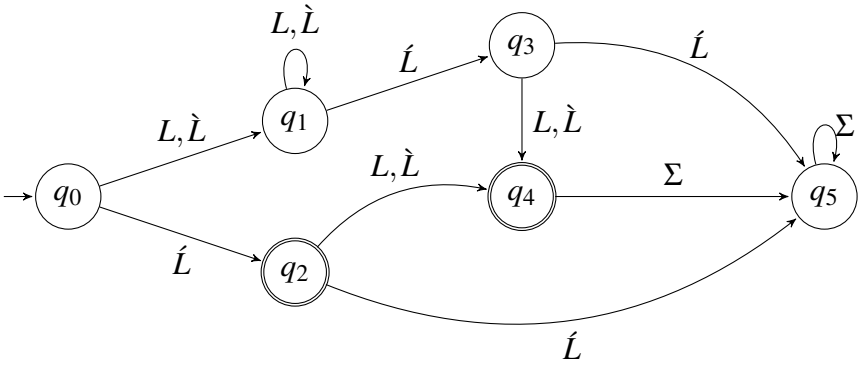
First, we define a sample of stress patterns in the human language Alawa which is an indigenous Australian language. In this language, there is only one syllable weight and two level of stress (LAMBERT; ROGERS, 2019). Then, we use the alphabet $\Sigma_{alawa} = \{L, \grave{L}, \acute{L}\}$. In Alawa, monosyllables are always stressed. In addition, in all other words, primary stress falls on the penultimate syllable. Also, the following universal constraint is essential: every word has exactly one syllable that receives primary stress.

We adapted the constraints in (LAMBERT; ROGERS, 2019) to the language of QDNF formulas. The following QDNF formula represents the constraints of the stress patterns in Alawa.

$$\begin{aligned}\varphi_{alawa} = \ & (suff(\acute{L}L) \wedge \gamma(\acute{L}) \geq 1 \wedge \neg\gamma(\acute{L}) \geq 2) \vee \\ & (suff(\acute{L}\grave{L}) \wedge \gamma(\acute{L}) \geq 1 \wedge \neg\gamma(\acute{L}) \geq 2) \vee \\ & (pref(\acute{L}) \wedge \neg lgeq(2)).\end{aligned}$$

The DFA A_{alawa} in Figure 33 also represents the stress patterns in Alawa. We generate a sample of stress patterns in Alawa according to the following procedure. First, we generate all strings w over the alphabet Σ_{alawa} such that $|w| \leq 3$. For each generated string, we classify this string in P or N according to A_{alawa}. For example, $L\acute{L}L \in P$ and $\acute{L}\acute{L}L \in N$. This sample is composed by 7 positive strings and 32 negative strings.

Figure 33 – DFA A_{alawa}

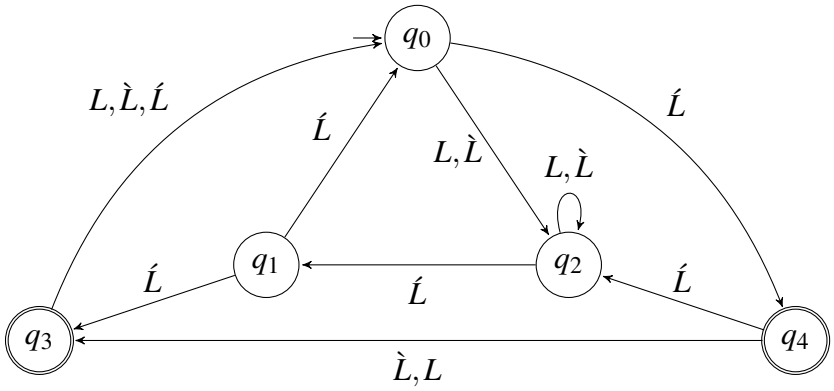


Source: Own elaboration

Now, we describe how we use the three methods in this example. For our method QDNFSAT, we use $s = r + m$ as the main parameter. Then, we start with $r = 1$ and $m = 1$, i.e., $s = 2$. We successively increase s by one until a QDNF formula is found. Note that, in general, for a value of s, there is many values for r, m. For l-QDNFSAT, we follow a similar procedure. We use $s = r + m + l$ such that the procedure starts with $s = 3$, and we successively increase s by one until a QDNF formula is found. For DFASAT, we start with the number of states $n = 1$. Then, we increase n by one until a DFA is found.

DFASAT returned the following DFA in Figure 34. This DFA has 5 states, while A_{alawa} consists of 6 states. In addition, the DFA in Figure 34 accepts strings with more than one occurrence of $\acute{L}$. For example, it accepts $\acute{L}L\acute{L}\acute{L}$.

Figure 34 – DFA returned by DFASAT



Source: Own elaboration

Our method QDNFSAT returned the QDNF formula below. It consists of 18 literals and only one clause. Also observe that the formula below does not hold in $LL\acute{L}L$ since $LL\acute{L}L \not\models \neg pref(LL)$. However, $LL\acute{L}L$ is a stress pattern in Alawa. This is probably due to the large number of literals in the formula returned by QDNFSAT. Clauses with a large number of literals constrains too much the number of strings classified as positive since the string must be true in all literals.

$$\begin{aligned}
&\gamma(\acute{L}) < 2 \wedge \sigma(\acute{L}) \geq 1 \wedge \sigma(\acute{L}\grave{L}L) < 1 \wedge \\
&\neg suff(\grave{L}\acute{L}) \wedge \neg suff(\grave{L}\grave{L}) \wedge \neg pref(LL) \wedge \\
&\neg suff(\acute{L}\acute{L}) \wedge \neg suff(L\acute{L}) \wedge \sigma(LLL) < 1 \wedge \\
&\gamma(LL\acute{L}) < 1 \wedge \gamma(\grave{L}L\grave{L}) < 1 \wedge \sigma(\acute{L}L\grave{L}) \wedge \\
&\gamma(\acute{L}LL) < 1 \wedge \gamma(\acute{L}\acute{L}L) < 1 \wedge \gamma(LL\grave{L}) \wedge \\
&\sigma(LL\grave{L}) < 1 \wedge \gamma(L\grave{L}\grave{L}) < 1 \wedge \gamma(\acute{L}\grave{L}\grave{L}).
\end{aligned}$$

l-QDNFSAT returned the following QDNF formula. It consists of 6 literals, 3 clauses, and at most 2 literals per clause. Moreover, it is easy to see that this formula is equivalent to φ_{alawa}. For example, the formula below holds in $LL\acute{L}L$. Also, it does not hold in $\acute{L}L\acute{L}\acute{L}$. Furthermore, observe that $\gamma(\acute{L}) \geq 1$ is not necessary in the first two clauses. For example, $suff(\acute{L}L)$ forces at least one occurrence of $\acute{L}$ since $suff(\acute{L}L) \models \gamma(\acute{L}) \geq 1$. Then, this formula

considers the universal constraint that primary stress falls on exactly one syllable.

$$(suff(\acute{L}L) \wedge \neg\gamma(\acute{L}) \geq 2) \vee$$
$$(suff(\acute{L}\grave{L}) \wedge \neg\gamma(\acute{L}) \geq 2) \vee$$
$$(\sigma(\acute{L}) \geq 1 \wedge \neg lgeq(2)).$$

7.4.2 Stress Patterns in Cambodian

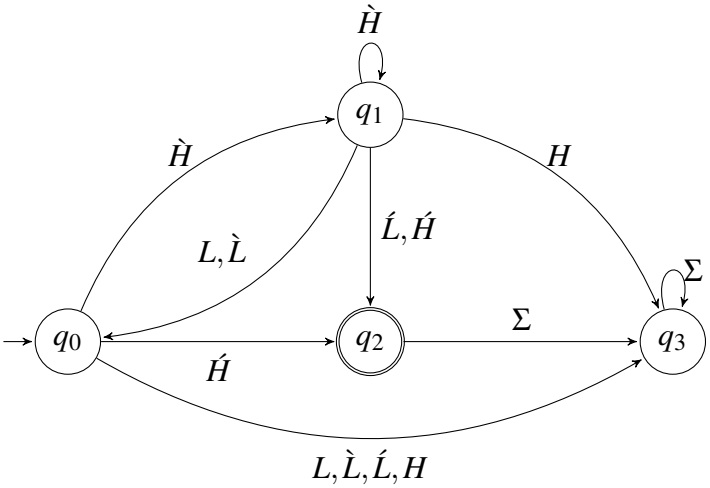
Now, we turn to the human language Cambodian which is the official language of Cambodia. The stress patterns in Cambodian satisfy the following constraints: in words of all sizes, primary stress falls on the final syllable and secondary stress falls on all heavy syllables. Moreover, light syllables occur only immediately following heavy syllables and light monosyllables do not occur. Also, every word has exactly one syllable that receives primary stress.

Then, we consider an alphabet $\Sigma_{cambodian} = \{L, H, \acute{L}, \acute{H}, \grave{L}, \grave{H}\}$. We are interested in formal languages over Σ^* which represent stress patterns in Cambodian. We adapted the constraints in (LAMBERT; ROGERS, 2019) to the language of QDNF formulas. Then, we obtained the formula $\varphi_{cambodian}$ below.

$$\begin{aligned}
\varphi_{cambodian} = \ & (suff(\acute{L}) \vee suff(\acute{H})) \wedge \\
& \gamma(H) < 1 \wedge \\
& (pref(\grave{H}) \vee pref(\acute{H})) \wedge \\
& \gamma(LL) < 1 \wedge \gamma(L\grave{L}) < 1 \wedge \gamma(L\acute{L}) < 1 \wedge \\
& \gamma(\grave{L}L) < 1 \wedge \gamma(\grave{L}\grave{L}) < 1 \wedge \gamma(\grave{L}\acute{L}) < 1 \wedge \\
& \gamma(\acute{L}L) < 1 \wedge \gamma(\acute{L}\grave{L}) < 1 \wedge \gamma(\acute{L}\acute{L}) < 1 \wedge \\
& \neg(\gamma(\acute{L}) \geq 1 \wedge \gamma(\acute{H}) \geq 1) \wedge \\
& (\gamma(\acute{L}) = 1 \vee \gamma(\acute{H}) = 1).
\end{aligned}$$

The DFA $A_{cambodian}$ in Figure 35 below recognizes exactly the stress patterns in Cambodian. We use the sample procedure defined in Subsection 7.4.1 in order to generate a sample of stress patterns in Cambodian. Then, the sample consists of all strings w over the alphabet $\Sigma_{cambodian}$ such that $|w| \leq 3$. We classify each string as positive or negative in accord with $A_{cambodian}$. Then, this sample has 7 positive strings and 251 negative ones. For example, $\grave{H}L\acute{H} \in P$ and $\grave{H}L\acute{L} \in N$.

Figure 35 – DFA $A_{cambodian}$



Source: Own elaboration

We ran the three method as in Subsection 7.4.1. DFASAT returned a DFA isomorphic to $A_{cambodian}$. QDNFSAT returned a QDNF formula with 208 literals only one clause. Then, we do not provide this QDNF formula here. However, the literal $\neg\gamma(\grave{L}\grave{H}) \geq 1$ is present in the clause. Therefore, this QDNF formulas does not hold in $\grave{H}\grave{L}\grave{H}\acute{L}$, for example. Observe that $\grave{H}\grave{L}\grave{H}\acute{L}$ is a stress pattern in Cambodian. Again, a likely explanation for this phenomenon is that a string must hold in all literals in order to be classified as positive.

Lastly, l-QDNFSAT returned the QDNF formula below. Clearly, l-QDNFSAT returned a QDNF formula smaller than the one provided by QDNFSAT. The formula below consists of 5 clauses and 10 literals with at most 2 literals per clause. However, it holds in $L\grave{H}L\acute{H}$. Observe that his string is not a stress pattern in Cambodian since there exists an L which does not occur immediately following a heavy syllable.

$$
\begin{aligned}
&(suff(\acute{H}) \wedge \sigma(\grave{H}L\grave{H}) \geq 1) \vee \\
&(pref(\grave{H}) \wedge suff(\grave{H}\acute{L})) \vee \\
&(suff(\grave{H}\acute{H}) \wedge pref(\grave{H})) \vee \\
&(\neg suff(H) \wedge \sigma(\grave{H}\grave{L}\acute{H}) \geq 1) \vee \\
&(\sigma(\acute{H}) \geq 1 \wedge \neg lgeq(2)).
\end{aligned}
$$

8 QDNF SYNTHESIS FROM NOISY SAMPLES

In this chapter, we show how to apply our techniques to the synthesis of sentences from noisy data. In this case, not all examples should necessarily be covered. Then, we define a cost function for QNF sentences on samples. We also introduce the notion of indistinguishability graph using the Ehrenfeucht–Fraïssé game. Using this notion of indistinguishability graph, we show how to determine a suitable set of atomic QNF sentences for the synthesis problem from noisy samples.

Since, in the noisy case, a QNF sentence may not classify correctly a string in the sample, our first approach consists of covering the majority of the examples. In this first noise tolerant extension of our framework, we consider the synthesis of QNDF and l-QDNF sentences. In both cases, we solve the synthesis problem using a MaxSAT encoding. The main results of our first approach were submitted to (ROCHA *et al.*, 2020).

We also present an approach in which the maximum number of misclassified strings is also given as part of the input. Then, the advantage of this approach is that it can verify whether there exists a QNF sentence with cost bounded by the input. Again, we regard the synthesis of QNDF and l-QDNF sentences. In this different scenario, we directly adapt the SAT-based approach to learn DFA from noisy samples in (ULYANTSEV *et al.*, 2015). The primary results of this second approach were accepted for publication in (ROCHA; MARTINS, 2019).

8.1 Cost Function and Indistinguishability Graph

In our extension to handle noisy samples, a sentence may not cover an example. Therefore, we use a cost associated with a sentence for not covering examples. Formally, the cost function of a sentence φ on S is the fraction of all examples that are not covered by φ.

$$cost(\varphi, S) := \frac{|\{u \in P \mid u \not\models \varphi\}| + |\{v \in N \mid v \models \varphi\}|}{|P| + |N|}.$$

One should note that P and N are not necessarily disjoint, that is, we may have $P \cap N \neq \emptyset$. Then, we do not use $|P \cup N|$ as a denominator in the definition of $cost(\varphi, S)$.

Observe that, in the cost function we are considering, all examples have the same weight. We may choose another cost function when the sample is unbalanced. For example, if the number of positive examples is far greater than the number of negative examples, we may choose a higher weight for the negative examples.

Now, we show that the set Φ_S^r is a suitable set, i.e., there exists a QNF sentence φ such that $A(\varphi) \subseteq \Phi_S^r$, and for all φ' such that $qnfr(\varphi') \leq r$, then $cost(\varphi, S) \leq cost(\varphi', S)$. Then, there is no sentence with QNF rank at most r that is better than φ. First, we define the indistinguishability graph for a sample S and a QNF rank r.

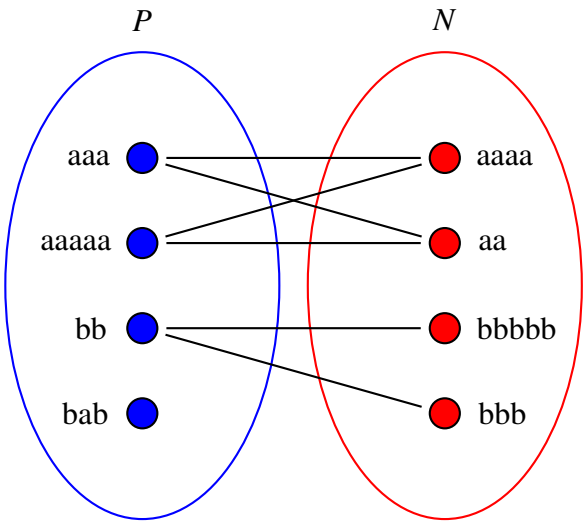
Definition 8.1.1 (Indistinguishability Graph)**.** *Let S be a sample and r be a QNF rank. The indistinguishability graph for S and r is a undirected bipartite graph $G(S,r) := (P,N,E)$ such that*

$$E = \{(u,v) \mid u \in P,\ v \in N,\ \textit{and}\ MinRound(u,v) > r\}.$$

The edges of $G(S,r)$ consist of pairs (u,v) such that the minimum QNF rank of a sentence distinguishing u from v is greater than r. A sentence of QNF rank at most r can not distinguish u from v for $(u,v) \in E$. Then, for all φ such that $qnfr(\varphi) \leq r$, $u \models \varphi$ and $v \models \varphi$ or $u \not\models \varphi$ and $v \not\models \varphi$.

Example 8.1.1. *Let $S = (P,N)$ such that $P = \{abbb, aabb, bbab, bbaa\}$, $N = \{aaab, abab, babb\}$, and $r = 1$. Then, $G(S,r)$ is represented in Figure 36. For the edge $(aaa, aa) \in E$ and $\varphi = \gamma(b) \geq 1$, $aaa \not\models \varphi$ and $aa \not\models \varphi$.*

Figure 36 – Indistinguishability graph of Example 8.1.1



Source: Own elaboration

From Example 8.1.1, one should note that the set of edges induces a partition on $P \cup N$ such that each partition is the set of vertices of a complete bipartite graph. It is straightforward to compute

the complete bipartite subgraphs in linear time in the numbers of vertices of the graph. The idea is to compute the connected components of the graph. Each connected component is a complete bipartite subgraph of the original graph. In what follows, we show this claim.

Proposition 8.1.1. *Let $G(S,r)$ be a indistinguishability graph for some S and r. Assume that $w_1, w_1, w_2, ..., w_n$ is a path in $G(S,r)$. Then, $MinRound(w_1, w_n) > r$.*

Proof. Clearly, when $n = 1$, $MinRound(w_1, w_n) > r$. By induction hypothesis, suppose that it holds for $w_1, w_1, w_2, ..., w_{k-1}$. Now, let $w_1, w_1, w_2, ..., w_k$ be a path. Then, it follows that $(w_{k-1}, w_k) \in E$. By definition, $MinRound((w_{k-1}, w_k) > r$. By induction hypothesis, it follows that $MinRound((w_1, w_{k-1}) > r$. By contradiction, assume that there exists a QNF sentence φ such that $qnfr(\varphi) \leq r$, $w_1 \models \varphi$ and $w_k \not\models \varphi$. Since $MinRound((w_1, w_{k-1}) > r$, then $w_{k-1} \models \varphi$. Again, as $MinRound((w_{k-1}, w_k) > r$, then $w_k \models \varphi$. This is an absurd. □

Proposition 8.1.2. *Let $G(S,r)$ be a indistinguishability graph for some sample S and QNF rank r. Let $G' = (V', E')$ be a connected component of $G(S,r)$. Then, $G' = (V', E')$ is a complete bipartite graph.*

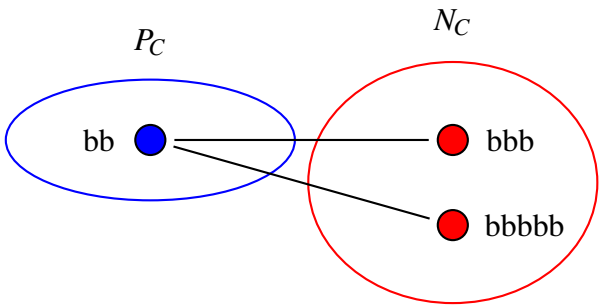
Proof. Let $G' = (V', E')$ be a connected component of $G(S,r)$. Clearly, $G' = (V', E')$ is bipartite by the definition of $G(S,r)$. Then, we can see G' as a bipartite $G' = (P_C, N_C, E')$ such that, $V' = P_C \cup N_C$, $P_C \cap N_C = \emptyset$, $P_C \subseteq P$ and $N_C \subseteq N$. By contradiction, assume that $G' = (P_C, N_C, E')$ is not a complete bipartite graph. Then, there exists a pair $(u, v) \notin E'$ such that $u \in P_C$ and $v \in N_C$. It follows that there exists a QNF sentence φ such that qnfr$(\varphi) \leq r$, $u \models \varphi$ and $v \not\models \varphi$. As G' is connected, there exists a path $u = u_1, v_1, u_2, ..., u_n, v_n = v$ in G'. Clearly, by Proposition 8.1.1, $MinRound(u, v) > r$. Then, by definition, $(u, v) \in E$, and then, $(u, v) \in E'$. This is an absurd. Therefore, G' is a complete bipartite graph. □

Let $G(S,r)$ be a indistinguishability graph. A cluster $C = (P_C, N_C, E_C)$ of G is a complete bipartite subgraph of G such that $(P_C \cup N_C, E_C)$ is a connect component of G, $P_C \subseteq P$, $N_C \subseteq N$.

Example 8.1.2. *Let $G(S,r)$ be the indistinguishability graph of Example 8.1.1. The graph C in Figure 37 is a cluster of $G(S,r)$.*

It is straightforward that a positive example and a negative example in the same cluster of $G(S,r)$ are indistinguishable by QNF sentences of QNF rank at most r. This also holds for any pair of examples in the same cluster as in the following result.

Figure 37 – A cluster of the indistinguishability graph of Example 8.1.1



Source: Own elaboration

Lemma 8.1.1. *Let C be a cluster of $G(S,r)$ for some S and r. For all $w_1, w_2 \in P_C \cup N_C$, and sentence φ of QNF rank at most r, $w_1 \models \varphi$ and $w_2 \models \varphi$ or $w_1 \not\models \varphi$ and $w_2 \not\models \varphi$.*

Proof. The case such that $w_1 \in P_C$ and $w_2 \in N_C$ follows directly from the definition of $G(S,r)$. Now, suppose that $w_1, w_2 \in P_C$. The case such that $w_1, w_2 \in N_C$ is analogous. Then, it must exist $v \in N_C$ such that $(w_1, v), (w_2, v) \in E_C$. Let φ be a sentence of QNF rank at most r. Then, φ is true in w_1 and v or it is false in w_1 and v. Without loss of generality, assume that φ is true in w_1 and v. As $(w_2, v) \in E_C$, φ must also be true in w_2. Therefore, the lemma holds. □

Furthermore, for a cluster C of $G(S,r)$, and a formula φ such that $\text{qnfr}(\varphi) \leq r$, φ is true in all vertices of $P_C \cup N_C$ or it is false in all vertices of $P_C \cup N_C$. For example, for $\varphi = \neg\text{pref}(b)$, φ is false in all vertices of $P_C \cup N_C$ in Example 8.1.2. This result is formalized in the following lemma. It follows directly from Lemma 8.1.1.

Lemma 8.1.2. *Let C be a cluster of $G(S,r)$ for some S and r. Let φ be a QNF sentence such that $qnfr(\varphi) \leq r$. Then, either for all $w \in P_C \cup N_C$, $w \models \varphi$ or for all $w \in P_C \cup N_C$, $w \not\models \varphi$.*

Therefore, for a cluster C of $G(S,r)$ such that $|P_C| > 0$ and $|N_C| > 0$, any formula of QNF rank at most r can not cover all vertices of the cluster. In this case, a solution to the synthesis problem must cover the examples of the biggest set between P_C and N_C. For example, a solution must cover the examples in P_C if $|P_C| > |N_C|$. Let $\mathscr{C}(S,r) = \{C_1, ..., C_n\}$ be the set of clusters of $G(S,r)$. Let $\mathscr{P} = \{C \in \mathscr{C} \mid |P_C| \geq |N_C|\}$ be the set of all clusters such that the number of positive examples is not less than the number of negative examples. We define $\mathscr{N}$ in an analogous way. Then, the lower bound to the cost of a sentence φ of QNF rank at most r on a sample S is

$$\frac{sum\{|N_C| \mid C \in \mathscr{P}\} + sum\{|P_C| \mid C \in \mathscr{N}\}}{|P| + |N|}.$$

For example, the lower bound to the setting in Example 8.1.1 is $\frac{2+1}{8}$. Now, we show a formula of QNF rank at most r such that its atoms are among the ones in Φ^r_S, and its cost on S is the lower bound. For a cluster C, we define an arbitrary vertex of C as w_C. Let S be a sample and r a QNF rank, then

$$\varphi^r_S := \bigvee_{C_P \in \mathscr{P}} \bigwedge_{C_N \in \mathscr{N}} \varphi^r_{u_{C_P}, v_{C_N}}$$

such that $\varphi^r_{u_{C_P}, v_{C_N}}$ is as in Lemma 7.2.1. Clearly, $qnfr(\varphi^r_S) \leq r$. In what follows, we give an example.

Example 8.1.3. *Let S and r be as in Example 8.1.1. Let $\varphi^r_S = \neg\gamma(b) \geq 1 \vee \gamma(a) \geq 1$. The cost of φ^r_S on S is $cost(\varphi^r_S, S) = \frac{3}{8}$. Then, φ^r_S is a solution to the instance of QDNFS-Noise in Example 8.1.1.*

Theorem 8.1.1. *Let S be a sample and r be a QNF rank. Then, for all φ such that $qnfr(\varphi) \leq r$, it follows that $cost(\varphi^r_S, S) \leq cost(\varphi, S)$.*

Proof. We compute the cost of φ^r_S. Let $C_P \in \mathscr{P}$. Clearly, it follows that $u_{C_P} \models \bigwedge_{C_N \in \mathscr{N}} \varphi^r_{u_{C_P}, v_{C_N}}$. Let $u \in P_{C_P} \cup N_{C_P}$ be an element of C_P. Then, by Lemma 8.1.2, $u \models \bigwedge_{C_N \in \mathscr{N}} \varphi^r_{u_{C_P}, v_{C_N}}$. Therefore, $u \models \varphi^r_S$ and φ^r_S does not cover $|N_{C_P}|$ elements of C_P.

Now, let $C_N \in \mathscr{N}$. Let $v \in P_{C_N} \cup N_{C_N}$ be an element of C_N. By contradiction, suppose that $v \models \bigwedge_{C \in \mathscr{N}} \varphi^r_{u_{C_P}, v_C}$ for some $C_P \in \mathscr{P}$. Then, $v \models \varphi^r_{u_{C_P}, v_{C_N}}$. Then, $v_{C_N} \models \varphi^r_{u_{C_P}, v_{C_N}}$. This is an absurd. It follows that, $v \not\models \varphi^r_S$. Therefore, φ^r_S does not cover $|P_{C_N}|$ elements of C_N. Finally, it follows that

$$cost(\varphi^r_S, S) = \frac{sum\{|N_C| \mid C \in \mathscr{P}\} + sum\{|P_C| \mid C \in \mathscr{N}\}}{|P| + |N|}.$$

Therefore, for all φ such that $qnfr(\varphi) \leq r$, $cost(\varphi^r_S, S) \leq cost(\varphi, S)$. □

8.2 A MaxSAT Approach

In this section, we present our first framework to handle noisy samples. The goal of this first framework is to find a QDNF sentence that minimizes the cost function concerning a given sample.

Here, we consider that the QNF rank is also given. First, we consider the synthesis of QDNF sentences in which the number of clauses is given as input. In what follows, we formally define the problem.

Definition 8.2.1 (min-QDNFS)**.** *Given a sample of strings S, a QNF rank r, a positive integer m, and a set* Φ *of atomic QNF sentences such that for all* $\mu \in \Phi$, $qnfr(\mu) \leq r$, *the min-QDNFS problem consists in finding a QDNF sentence* φ *such that*

- $A(\varphi) \subseteq \Phi$*;*
- φ *has m conjunctive clauses;*
- *For all* φ' *such that* $A(\varphi') \subseteq \Phi$, $cost(\varphi, S) \leq cost(\varphi', S)$.

Now, we show how to solve the min-QDNFS problem by a translation to MaxSAT. We call our approach QDNFMaxSAT since it is based on MaxSAT. We use Formula 7.1 and Formula 7.3, defined in Subsection 7.2.2, as hard clauses. Furthermore, we use Formula 7.2 and Formula 7.4, also defined in Subsection 7.2.2, as soft clauses. Formula 7.2 and Formula 7.4 are repeated below for convenience.

$$\bigvee_{\varphi \in P_v} \neg p'_{j,\varphi} \vee \bigvee_{\varphi \in N_v} p_{j,\varphi} \qquad \text{for } j \in \{1,...,m\}, v \in N. \tag{7.2}$$

$$\bigvee_{j \in \{1,...,m\}} c_{j,u} \qquad \text{for } u \in P. \tag{7.4}$$

Using Formula 7.2 as soft clauses allows a negative string to be true in the solution. Analogously, if we use Formula 7.4 as soft clauses, then a positive string may do not satisfy the solution. As a MaxSAT algorithm finds a valuation that maximizes the number of satisfied soft clauses, the number of strings covered by a solution is maximum.

A MaxSAT encoding provides means to minimize the number of literals as well. This can be achieved by including soft clauses in order to force all literals associated with atomic QNF formulas of Φ to be unused. This is achieved by using the following clauses.

$$q_{j,\varphi} \qquad \text{for } j \in \{1,...,m\},\ \varphi \in \Phi,\ q_{j,\varphi} \in \{p_{j,\varphi}, p'_{j,\varphi}\}. \tag{8.1}$$

Let $\Psi^{soft}_{S,\Phi^r,m}$ be the set of clauses in Formula 7.2, Formula 7.4, and $\Psi^{hard}_{S,\Phi^r,m}$ be the set of clauses in Formula 7.1 and Formula 7.3. A valuation $\mathscr{V}$ that is a solution to MaxSAT with instance $(\psi^{hard}_{S,\Phi^r,m}, \psi^{soft}_{S,\Phi^r,m})$ allows us to extract a solution $\varphi_{\mathscr{V}}$ to the min-QDNFS problem with instance S, Φ^r, m. The formula $\varphi_{\mathscr{V}}$ is as in Definition 7.2.2.

The next result states that our approach QDNFMaxSAT produces a solution to the min-QDNFS problem.

Theorem 8.2.1. *For a sample S, set Φ of atomic QNF formulas of QNF rank at most r, and a positive integer m, a model $\mathscr{V}$ of $\psi_{S,\Phi,m}$ provides a QDNF formula $\varphi_{\mathscr{V}}$ that is a solution to the min-QDNFS problem.*

Clearly, this MaxSAT encoding can be adapted to the case in which the maximum number of literals per clause is also given as input. Then, the goal is to find a l-QDNF sentence that minimizes the cost function.

Definition 8.2.2 (min-l-QDNFS)**.** *Given a sample of strings S, a QNF rank r, positive integers m, l, and a set Φ of atomic QNF sentences such that for all $\mu \in \Phi$, $qnfr(\mu) \leq r$, the min-l-QDNFS problem consists in finding a QDNF sentence φ such that*

- $A(\varphi) \subseteq \Phi$*;*
- *φ has m conjunctive clauses;*
- *Each conjunctive clause in φ consists of at most l literals;*
- *For all φ' such that $A(\varphi') \subseteq \Phi$, $cost(\varphi, S) \leq cost(\varphi', S)$.*

In order to solve the min-l-QDNFS problem using a MaxSAT encoding, we use Formula 7.11 and Formula 7.12 as soft clauses. These formulas are defined in Subsection 7.3.1 and repeated below for convenience. Let $\Psi^{soft}_{S,\Phi^r,m,l}$ be the set of clauses in Formula 7.11, Formula 7.12.

$$\bigvee_{i=1}^{m} z_i^u \qquad \text{for } u \in P. \tag{7.11}$$

$$\neg z_i^v \qquad \text{for } i \in \{1,...,m\},\ v \in N. \tag{7.12}$$

We also use Formulas 7.5-7.10 as hard clauses. Let $\Psi^{hard}_{S,\Phi^r,m,l}$ be the set of clauses in Formulas 7.5-7.10. A valuation $\mathscr{V}$ which is a solution to MaxSAT with instance $(\psi^{hard}_{S,\Phi^r,m,l}, \psi^{soft}_{S,\Phi^r,m,l})$ provides a solution $\varphi_{\mathscr{V}}$ to the min-l-QDNFS problem with instance S, Φ^r, m, l. The formula $\varphi_{\mathscr{V}}$ is as in Definition 7.3.2. We call our method l-QDNFMaxSAT.

Again, one can define a MaxSAT encoding that minimizes the number of literals as well. The following soft clauses impose that a valuation must satisfy as many $x_{i,j,*}$ as possible, i.e., maximizes the the number of literals that must be skipped.

$$x_{i,j,*} \qquad \text{for } i \in \{1,...,m\},\ j \in \{1,...,l\}. \tag{8.2}$$

The next result states that our approach l-QDNFMaxSAT produces a solution to the min-l-QDNFS problem.

Theorem 8.2.2. *For a sample S, set Φ of atomic QNF formulas of QNF rank at most r, and positive integers m, l, a model $\mathscr{V}$ of $\psi_{S,\Phi,m}$ provides a QDNF formula $\varphi_{\mathscr{V}}$ that is a solution to the min-QDNFS problem.*

We finish this section by showing that our framework for handling noisy data defined in this section is an extension of that for samples without noise. This extension may be used in real settings where examples are not guaranteed to be covered. Then, we show that a solution to the case with noise is also a solution to the noiseless case. We give this result for the synthesis of QDNF and l-QDNF sentences.

Theorem 8.2.3. *Let S be a sample, r be a QNF rank, m be a positive integer, and Φ be a set of atomic QNF sentences such that for all $\mu \in \Phi$, $qnfr(\mu) \leq r$. If there exists a QDNF formula φ consistent with S such that $A(\varphi) \subseteq \Phi$ and φ has m conjunctive clauses, then a solution to the min-QDNFS problem is also a solution to the QDNFS problem.*

Proof. Let μ' be a solution to the min-QDNFS problem. As φ is consistent with S, then $cost(\varphi, S) = 0$. Therefore, $cost(\mu', S) = 0$ because for all φ' such that $A(\varphi') \subseteq \Phi$, it holds that $cost(\mu', S) \leq cost(\varphi', S)$. Then, μ' is consistent with S and it is a solution to the QDNFS problem. □

The proof of the following theorem for the synthesis of l-QDNF sentences is directly adapted from the proof above.

Theorem 8.2.4. *Let S be a sample, r be a QNF rank, m and l be positive integers, and Φ be a set of atomic QNF sentences such that for all $\mu \in \Phi$, $qnfr(\mu) \leq r$. If there exists a l-QDNF formula φ consistent with S such that $A(\varphi) \subseteq \Phi$ and φ has m conjunctive clauses, then a solution to the min-l-QDNFS problem is also a solution to the l-QDNFS problem.*

Therefore, our methods QDNFMaxSAT and l-QDNFMaxSAT for data with noise may also be used to handle noiseless data. Then, each method is a unified approach for noise-free and noisy data.

In the following section, we give our second approach to the synthesis of QDNF sentences from noisy samples.

8.3 The SAT-Based Approach

In this section, we present our second approach to the synthesis problem from noisy samples. In this second approach, the maximum number of strings that can me misclassified is given as input as well. Then, the objective is finding a QDNF sentence which obey this requirement. This approach is directly adapted from the one in (ULYANTSEV *et al.*, 2015) for synthesis of DFA. In what follows, we formally define the problem for general QDNF sentences.

Definition 8.3.1 (Synthesis of QDNF Formulas from Noisy Samples (QDNFSN))**.** *Given a sample of strings S, a set Φ of atomic QNF sentences, a positive integer m, and a natural number K, the QDNFSN problem consists in finding, if it exists, a QDNF formula φ such that*

- $A(\varphi) \subseteq \Phi$*;*
- φ *has m conjunctive clauses;*
- $cost(\varphi, S) \leq K$.

In what follows, we give an example of instance and solution to the above problem. Let S be the sample in Table 5. One should note that P and N in S are not disjoint since $\grave{H}LH\grave{L}$ is positive and negative.

Table 5 – Sample of stress patterns in Cambodian.

String	Class
$\grave{H}\grave{L}H\acute{L}$	Positive
$\grave{H}LH\acute{L}$	Negative
$\grave{H}L\grave{H}L\acute{H}$	Positive
$\grave{H}L\grave{L}\grave{H}\acute{H}$	Negative
$\grave{H}L\grave{H}L$	Positive
$\grave{H}L\grave{H}L$	Negative

Source: Own elaboration

Example 8.3.1. *Let $\Phi = \{pref(\grave{H}\grave{L}), \gamma(\grave{L}) \geq 1, suff(\acute{L}), suff(\acute{H}), \sigma(H) \geq 1, \gamma(L\grave{L}) \geq 1\}$ be the set of atomic QNF sentences, $m = 2$, and $K = 2$. Let S be the sample in Table 5. A solution to the QDNFSN problem is the formula φ below. The reader should observe that $cost(\varphi, S) \leq 2$.*

$$\varphi = (\neg\sigma(H) \geq 1 \wedge suff(\acute{L})) \vee (suff(\acute{H}) \wedge \neg\gamma(L\grave{L}) \geq 1).$$

Now, in order to solve the QDNFSN problem, we adapt the approach defined in (ULYANTSEV *et al.*, 2015). Then, we define a SAT-based method QDNFSAT-Noise for solving this problem.

For the noiseless case in Subsection 7.2.2, we consider the constraints in Formula 7.2 and Formula 7.4. Now, we adapt these constraints to deal with noisy samples.

As in Subsection 4.3.2, for each string w in the sample, we use a variable e_w such that it is set to true if the class of string w can (but does not have to) be incorrect. Then, Formulas 8.3-8.4 below express that if a string w is in the correct class, then a QDNF formula must cover w.

$$\neg e_u \rightarrow \bigvee_{j \in \{1,\ldots,m\}} c_{j,u} \qquad \text{for } u \in P. \tag{8.3}$$

$$\neg e_v \rightarrow (\bigvee_{\varphi \in P_v} \neg p'_{j,\varphi} \vee \bigvee_{\varphi \in N_v} p_{j,\varphi}) \qquad \text{for } j \in \{1,\ldots,m\},\ v \in N. \tag{8.4}$$

In order to limit the number of strings that can be considered in the wrong class, we also use the constraints in Formulas 4.7-4.11 defined in Subsection 4.3.2.

Formulas 8.3-8.4 can easily be translated to CNF. Let $\psi_{S,\Phi,m,K}$ be the conjunction of Formulas 8.3-8.4, Formulas 4.7-4.11, Formula 7.1, and Formula 7.3 in CNF. If $\psi_{S,\Phi,m,K}$ is satisfiable, then we can derive a QDNF formula $\varphi_{\mathscr{V}}$ from a model $\mathscr{V}$ of $\psi_{S,\Phi,m,K}$.

Theorem 8.3.1. *For a sample S, set Φ of atomic QNF sentences, a positive integer m, and a natural number K, a model $\mathscr{V}$ of $\psi_{S,\Phi,m,K}$ determines a QDNF formula $\varphi_{\mathscr{V}}$ that is a solution to the QDNFSN problem.*

Clearly, the QDNFSN problem is an extension of the QDNFS problem in the following sense: one can use our method QDNFSAT-Noise with $K = 0$ in order to solve the QDNFS problem. It follows that QDNFSAT-Noise also handles noiseless samples.

Theorem 8.3.2. *Let S be a sample, m be a positive integer, and Φ be a set of atomic QNF sentences. There exists a QDNF formula φ consistent with S such that $A(\varphi) \subseteq \Phi$ and φ has m conjunctive clauses if and only if there exists a solution to the QDNFSN problem with $K = 0$.*

We use the above result in order to prove the following theorem.

Theorem 8.3.3. *QDNFSN is NP-complete.*

Proof. The construction of Formula $\psi_{S,\Phi,m,K}$ takes polynomial time in the input size of QDNFSN. Then, this construction is a polynomial time reduction from QDNFSN to SAT. Since SAT is an NP problem, it follows that SQFS is in NP.

By Theorem 8.3.2, there is a polynomial time reduction from QDNFS to QDNFSN. Since QDNFS is NP-complete, it follows that QDNFSN is also an NP-complete problem. □

Now, we present the SAT-based approach to handle noisy samples for the case of l-QDNF sentences. Then, the maximum number of literals per clause is also given as input. We formally define the problem below.

Definition 8.3.2 (Synthesis of l-QDNF Formulas from Noisy Samples (l-QDNFSN)). *Given a sample of strings S, a set Φ of atomic QSF formulas, two positive integers m and l, and a natural number K, the l-QDNFSN problem consists in finding, if it exists, a QDNF sentence φ such that*

- $A(\varphi) \subseteq \Phi$
- *φ has m conjunctive clauses;*
- *φ is an l-QDNF sentence;*
- $cost(\varphi, S) \leq K$.

Now, we show how to solve this problem by adapting the approach in (ULYANTSEV *et al.*, 2015). Since our solution is based on SAT, we call our method l-QDNFSAT-Noise (Synthesis of l-QDNF sentences from noisy samples using SAT). We adapt the constraints in Formula 7.11 and Formula 7.12 as in the following formulas.

$$\neg e_u \rightarrow (\bigvee_{i=1}^{m} z_i^u) \qquad \text{for } u \in P. \tag{8.5}$$

$$\neg e_v \rightarrow \neg z_i^v \qquad \text{for } i \in \{1,...,m\},\ v \in N. \tag{8.6}$$

Clearly, Formulas 8.5-8.6 can be easily translated to conjunctive normal form. Let $\psi_{S,\Phi,m,l,K}$ be the conjunction of Formulas 8.5-8.6, Formulas 4.7-4.11, Formula 7.5-7.10 in CNF. The proofs of the following results are analogous to the proofs of Theorems 8.3.1-8.3.3.

Theorem 8.3.4. *For a sample S, set Φ of atomic QNF sentences, positive integers m and l, and a natural number K, a QDNF formula $\varphi_{\mathscr{V}}$ that is a solution to the l-QDNFSN problem can be derived from a model $\mathscr{V}$ of $\psi_{S,\Phi,m,l,K}$.*

Theorem 8.3.5. *Let S be a sample, m and l be positive integers, and* Φ *be a set of atomic QNF sentences. There exists a l-QDNF formula* φ *consistent with S such that* $A(\varphi) \subseteq \Phi$ *and* φ *has m conjunctive clauses if and only if there exists a solution to the l-QDNFSN problem with* $K = 0$.

Theorem 8.3.6. *l-QDNFSN is NP-complete.*

9 CONCLUSIONS AND FUTURE WORK

This work consists in defining approaches to obtain first-order sentences from positive and negative structures over a fixed class. In the general case, for a fixed class of structures, given a sample of structures in the class, the task is to find a first-order sentence that holds in all positive structures and does not hold in any negative structure. We considered the following classes of structures: monadic structures, equivalence structures, disjoint unions of linear order, and strings.

Our first approach considers first-order formulas over standard vocabularies. In this case, the goal is to find a first-order sentence such that the quantifier rank is minimum. We introduced an algorithm that returns, in polynomial time, a first-order sentence of minimum quantifier rank that is consistent with a given sample of structures. Our work is motivated by the algorithm defined in (KAISER, 2012) that runs in exponential time for these classes of structures since it works for arbitrary classes of structures. Therefore, our approach is an improvement over the work in (KAISER, 2012), for the particular problem on monadic structures, equivalence structures, disjoint unions of linear orders, and strings.

In general, our first approach consists in using results on Ehrenfeucht–Fraïssé games for the classes of structures we are considering. The algorithm defined in (KAISER, 2012) uses Hintikka formulas which have size exponential in the size of a given structure. Then, we introduced the distinguishability sentences which are defined based on the conditions characterizing winning strategies for the Spoiler on MS, ES, DULO, and strings. Given two structures $\mathscr{A}, \mathscr{B}$ and a natural number r, the distinguishability sentences hold in $\mathscr{A}$, do not hold in $\mathscr{B}$, and have quantifier rank at most r.

The first step of our algorithm is to find, for each pair $\mathscr{A} \in P$ and $\mathscr{B} \in N$, the minimum number of rounds r such that the Spoiler has a winning strategy in an EF game on $\mathscr{A}$ and $\mathscr{B}$. Let $\{r_1, ..., r_k\}$ be the set of minimum rounds computed in the first step. Then, the maximum value among $r_1, ..., r_k$ is the minimum quantifier rank of a first-order sentence consistent with the input sample of structures. Then, we use this quantifier rank to build distinguishability sentences for pair of structures $\mathscr{A}$ and $\mathscr{B}$ such that $\mathscr{A}$ is a positive structure and $\mathscr{B}$ is a negative structure.

In order to show that our algorithm runs in polynomial time, we showed that the size of distinguishability sentences is polynomial in the size of the input sample of structures. Furthermore, given $\mathscr{A}, \mathscr{B}$ and r, the number of distinguishability formulas is also polynomial

in the size of $\mathscr{A}$ and $\mathscr{B}$. Finally, we also show that any first-order sentence is equivalent to a Boolean combination of distinguishability sentences. This result suggests that our approach is likely to find any first-order sentence, up to logical equivalence, given a suitable sample of structures.

In order to summarize, our contributions and comparisons for the first approach are in the following items:

- We showed our result on a characterization of the winning strategies for Ehrenfeucht–Fraïssé games on disjoint unions of linear orders. Furthermore, for this class of structures, it takes polynomial time to find the minimum number such that the Spoiler has a winning strategy. We considered disjoint unions of linear orders because one may model states of the elementary blocks world by using them (COOK; LIU, 2003).
- We described the synthesis of minimum quantifier rank first-order sentences on a fixed class of structures. We considered the following classes: monadic structures, equivalence structures, disjoint unions of linear orders, and strings.
- For the classes of structures we are considering in this work, we defined distinguishability sentences. Given two structures $\mathscr{A}$, $\mathscr{B}$ and a quantifier rank r, a distinguishability sentence hold in exactly one of the given structures, and it has quantifier rank bounded by r. We also define distinguishability sentences in a way such that they have polynomial size. This result is essential in ensuring that our algorithm runs in polynomial time in the size of the sample. Furthermore, we also showed that any first-order formula is equivalent to a Boolean combination of distinguishability sentences.
- We designed an algorithm for the synthesis problem on the classes of structures we are considering. We used results on Ehrenfeucht–Fraïssé games to find the minimum quantifier rank. Our algorithm returns a Boolean combination of distinguishability sentences in order to find a first-order formula consistent with the sample. Our algorithm takes polynomial time in the size of the sample since it takes polynomial time to compute the minimum quantifier rank and distinguishability sentences have size polynomial.
- The algorithm for the synthesis problem on arbitrary structures defined in (KAISER, 2012) returns a Boolean combination of Hintikka formulas. Hintikka formulas have size exponential while distinguishability sentences have polynomial size. Besides, for arbitrary structures, the problem of determining whether the Spoiler has a winning strategy for a given number of rounds is a PSPACE-complete problem. Then, our algorithm runs in

polynomial time while the one in (KAISER, 2012) takes exponential time for the classes we are dealing with in this work. Then, our approach is an improvement, with respect to computational complexity, over the one in the literature for monadic structures, equivalence structures, disjoint unions of linear orders, and strings. This fact illustrates the importance of considering a fixed class of structures.

However, our first approach has disadvantages. The main first disadvantage is that general first-order sentences may be hard to read. The second main disadvantage is that the problem of deciding whether a sentence holds in a structure is PSPACE-complete even in the classes we are considering in this work. This motivated our second approach to the synthesis of first-order sentences. Then, we defined a quantifier-free disjunctive normal form QDNF for each class of structures we are dealing with. This normal form is easier to read than general first-order sentences. Furthermore, we showed that determining whether a formula in QDNF is true in a structure takes polynomial time.

We also showed that for every first-order sentence, there exists an equivalent formula in QDNF and vice versa. We also present the synthesis problem of finding a formula in QDNF that is consistent with a given sample of structures. We analyze this problem, and we show that it is NP-complete. We also present our method QDNFSAT based on a SAT encoding for solving this synthesis problem.

QDNFSAT may return QDNF sentences with a large number of literals per clause. Then, we also introduce the synthesis problem for l-QDNF sentences, i.e., QDNF sentences such that each conjunctive clause has at most l literals. Therefore, we design our SAT-base approach l-QDNFSAT to solve the problem in which the maximum number of literals per clause is also given as input. We also show that this new problem is NP-complete.

In this work, we also introduce an extended version of the synthesis problem in order to handle noisy samples of structures. We present two frameworks to the synthesis problem from noisy samples. In our first framework, the goal is to find a QDNF sentence that maximizes the number of structures correctly covered. We showed our approach QDNFMaxSAT to solve the extended version by a translation to MaxSAT. In the second framework, the maximum number of structures misclassified K is given as input. Then, the objective is finding an l-QDNF sentence such that it classifies incorrectly at most K structures. In this case, we propose a SAT-based approach l-QDNFSAT-Noise to solve this version of the synthesis problem. Therefore, QDNFMaxSAT and l-QDNFSAT-Noise are unified approaches that extract knowledge from

noiseless and noisy samples of classified structures.

We think that our algorithm can be useful when it is desirable to compute sentences defining a class of structures from positive and negative structures. For example, as we have outlined in the introduction, disjoint unions of linear orders may represent states of the elementary blocks world domain. Therefore, for disjoint unions of linear orders, our algorithm can be used to define initial and final states of this problem. In the elementary blocks world domain, given initial and final states, the goal is to find a sequence of actions capable of transforming the initial state into a final state.

Our results for strings are important to grammatical inference. In grammatical inference, one crucial problem is to find a model for a formal language from a sample of strings. For strings, first-order logic defines exactly the class of locally threshold testable languages (LTT) (THOMAS, 1982). Then, first-order logic over strings is a model for LTT languages. The class of LTT languages is a subregular class, i.e., it is included in the class of regular languages. Several recent works on grammatical inference consider models for subregular classes of languages (GARCIA; RUIZ, 2004; HEINZ, 2009; HEINZ, 2010; HEINZ *et al.*, 2012; HEINZ; ROGERS, 2013; STROTHER-GARCIA *et al.*, 2017; CHANDLEE *et al.*, 2019). As far as we know, there are no results available in the literature considering the class of LTT languages as a whole. Then, our approaches for strings may be seen as contributions for grammatical inference.

Below, we summarize our contributions to the synthesis of QNDF sentences.

- We defined a quantifier-free disjunctive normal form QDNF for first-order logic over strings. For the other classes of structures we are dealing with in this work, the definition is analogous. Sentences in this normal form are defined over a non-standard vocabulary such that atomic sentences are an abbreviation of first-order sentences over the standard vocabulary. This normal form is easier to read than general first-order sentences.
- We showed that, given a string w and a QDNF sentence φ, determining whether φ holds in w, i.e., $w \models \varphi$, takes polynomial time.
- We proved that, for strings, the class of languages defined by QDNF sentences is exactly the class of languages defined by general first-order sentences, i.e., locally threshold testable languages.
- We defined two synthesis problem for QDNF sentences. In the first one, given a positive integer m, the goal is to find a QNDF sentence with m conjunctive clauses and consistent with the input sample. In the second synthesis problem, given two positive integers m and

l, the problem is determining if exists a QDNF sentence with m conjunctive clauses, at most l literals per clause and consistent with the input sample.

- We designed methods to solve each of the above problems. Our methods are based on encoding these problems into the Boolean satisfiability problem. Then, one can use a modern SAT-solver to find a solution. We also proved that both problems are NP-complete.
- We presented two scenarios to handle noisy samples of strings. In the first scenario, the task is to return a QDNF sentence such that the number of misclassified strings in the input samples is minimum. In the second scenario, the problem is testing whether there exists an l-QDNF sentence such that it classifies incorrectly at most a given number of strings in the input sample.
- We described a Max-SAT method to solve the first scenario and a SAT-based approach to solve the second scenario. We also proved that the problem in the second scenario is NP-complete. The advantage of the SAT-based method is that it allows to determine exactly the existence or non-existence of a solution of the synthesis problem from noisy samples.

The results in this work motivate some directions for future research. For example, since our work is close to the area of grammatical inference, our results can be used in applications of this area such as classification of biological sequences (WIECZOREK; UNOLD, 2014) and finding patterns in phonology (STROTHER-GARCIA *et al.*, 2017; LAMBERT; ROGERS, 2019). In these applications, understanding the characteristics of the samples is essential. Therefore, our approaches QDNFSAT, l-QDNFSAT, QDNFMaxSAT, and l-QDNFSAT-Noise can be used to extract knowledge from samples of strings. Besides, l-QDNFSAT-Noise also considers the maximum number of literals per clause, and it also can minimize the total number of literals in solutions. This is useful because it provides small formulas that are easier to read. Interpretability is essential in fields where human experts need to be able to infer new knowledge from the model provided by the method.

Our methods for synthesis of QDNF sentences rely on a set of atomic QDNF formulas Φ^r_S. However, this set is potentially huge and could be a source of intractability. Then, a direction for future research is to define approaches to select a small relevant subset of Φ^r_S. One criterion for selecting such a subset is using only the distinguishability formulas instead of all Φ^r_S. Other criteria can be found in the literature of sequence classification (CHUZHANOVA *et al.*, 1998; LESH *et al.*, 1999; AGGARWAL, 2002; JI *et al.*, 2007; XING *et al.*, 2010).

Also, we plan to explore natural parameterizations of the synthesis problem, such as the number of clauses and sample size. Then, we can classify such parameterizations with the tools provided by parameterized complexity (FLUM; GROHE, 2006) such as fixed-parameter tractability and W-hierarchy. Using these tools, it is possible to determine what aspects of a problem cause it to become hard. For example, the DFA synthesis problem is NP-complete even when the number of states is 2 (FERNAU *et al.*, 2015; FERNAU, 2019). This means that the DFA synthesis problem with the number of states as the parameter does not belong to the W-hierarchy. On the other hand, this problem is fixed-parameter tractable (FPT) when parameterized with the sample size, i.e., the sum of the lengths of all strings it includes (FERNAU *et al.*, 2015). Therefore, parameterized complexity may be a useful tool to theoretically compare the synthesis of QDNF sentences and DFA synthesis concerning natural parameters.

In (STROTHER-GARCIA *et al.*, 2017), positions in strings may belong to more than one unary relation. Using this non-standard vocabulary, one can define languages with smaller formulas as seen in Section 4.4. Interesting future work is to modify our methods so that they also become robust concerning this non-standard vocabulary for strings. This representation of strings is very common in phonology (STROTHER-GARCIA, 2018; VU *et al.*, 2018; LAMBERT; ROGERS, 2019; CHANDLEE *et al.*, 2019).

As future work, we intend to investigate algorithms for the synthesis problem on other classes of structures. For example, equivalence structures with colors, embedded equivalence structures, and trees with level predicates (KHOUSSAINOV; LIU, 2009). For these classes of structures, the problem of determining if the Spoiler has a winning strategy is solved in exponential time. Besides, we intend to consider the class of strings with a limited order relation (MARIA *et al.*, 2009) and strings with a built-in linear order.

In the case of first-order logic over strings with a built-in linear order, it is necessary to show necessary and sufficient conditions for a winning strategy in Ehrenfeucht–Fraïssé games on these structures. First-order logic over strings with a built-in linear order is compelling because it captures star-free languages (THOMAS, 1982). An algorithm for the synthesis problem over strings can be used to derive a recognizer for a star-free language from a sample of strings. This result is significant because strings may be used to model text data, traces of program executions, DNA sequences, and sequences of symbolic data in general. Also, the class of star-free languages includes the class of locally threshold testable language. Moreover, first-order logic over strings with a built-in linear order is more expressive than the logic we considered in this work.

We also plan to extend our approach to other logics such as monadic second-order logic. For example, the Ajtain-Fagin game (AJTAI; FAGIN, 1990) is an extension of Ehrenfeucht–Fraïssé games such that the players are able to choose sets of elements in each round. Then, Ajtain-Fagin games may be seen as Ehrenfeucht–Fraïssé games for monadic second-order logic. Regular languages are exactly the languages definable in monadic second-order logic over strings (BÜCHI, 1960; LADNER, 1977). An algorithm which returns monadic second-order sentences consistent with a given sample of strings can be used in the problem of finding a model of a regular language consistent with a given sample of strings (HEULE; VERWER, 2010; ZAKIRZYANOV *et al.*, 2017; ZAKIRZYANOV *et al.*, 2019).

Another direction of future research is to combine our exact methods with traditional algorithms of machine learning, such as algorithms for learning decision trees. This idea is presented in (NEIDER; GAVRAN, 2018) for learning linear temporal logic formulas. Classical algorithms for learning decision trees do not ensure the minimality of the solution. However, they usually build a small decision tree, and they have a good trade-off between input size and solution time. Then, this may result in methods that return small QDNF sentences and scale to samples with a large number of strings.

We also intend to analyze classical logical properties for QNF sentences such as satisfiability and entailment. In phonology, it is essential to check whether a property follows from phonotactic constraints represented as logical formulas (LAMBERT; ROGERS, 2019). Then, we plan to define methods for these logical problems and investigate their complexity. A starting point is the literature of satisfiability modulo theories (SMT) (GANZINGER *et al.*, 2004; NIEUWENHUIS; OLIVERAS, 2005; NIEUWENHUIS *et al.*, 2005; NIEUWENHUIS *et al.*, 2006) since the main technique for the SMT problem relies on solvers for classical propositional logic.

Earlier works on SAT-based rule learning (IGNATIEV *et al.*, 2018; MALIOUTOV; MEEL, 2018; GRIENENBERGER; RITZERT, 2019) consider only the number of clauses as input. Since we introduced a SAT encoding that considers the maximum number of literals per clause, an exciting direction of future work consists of using our encoding for rule learning.

Lastly, we plan to investigate the synthesis of first-order queries. A first-order query is a mapping $Q_t : \mathscr{C}_1 \to \mathscr{C}_2$ from a class of structures $\mathscr{C}_1$ to another class of structures $\mathscr{C}_2$ defined by a tuple of first-order formulas t (IMMERMAN, 1999). In this case, a sample is a binary relation $S \subseteq \mathscr{C}_1 \times \mathscr{C}_2$. Then, the problem is to find a tuple of first-order formulas t such that $S \subseteq \{(\mathscr{A}, Q_t(\mathscr{A})) \mid$

$\mathscr{A} \in \mathscr{C}_1\}$ (JORDAN; KAISER, 2016). This methodology may be used for learning programs as logical queries (JORDAN; KAISER, 2013c), and finding reductions automatically (CROUCH *et al.*, 2010; JORDAN; KAISER, 2013b; JORDAN; KAISER, 2013a). Furthermore, we plan to investigate this problem when $\mathscr{C}_1$ and $\mathscr{C}_2$ are fixed classes of structures. For example, for strings, this approach may be used to find models of phonological processes (CHANDLEE, 2014; CHANDLEE *et al.*, 2014; CHANDLEE; HEINZ, 2018; HAO; ANDERSSON, 2019; CHANDLEE; JARDINE, 2019).

REFERENCES

ABITEBOUL, S.; HULL, R.; VIANU, V. **Foundations of Databases: The Logical Level**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0201537710.

AGGARWAL, C. C. On effective classification of strings with wavelets. In: **Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining**. [*S. l.*]: ACM, 2002. p. 163–172.

AJTAI, M.; FAGIN, R. Reachability is harder for directed than for undirected finite graphs. **The Journal of Symbolic Logic**, Cambridge Univ Press, v. 55, n. 01, p. 113–150, 1990.

ARORA, S.; FAGIN, R. On winning strategies in Ehrenfeucht-Fraïssé games. **Theoretical Computer Science**, Elsevier, v. 174, n. 1-2, p. 97–121, 1997.

AVCU, E.; SHIBATA, C.; HEINZ, J. Subregular complexity and deep learning. In: **Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML)**. [*S. l.*: *s. n.*], 2017. p. 20–33.

BARAHONA, P.; HÖLLDOBLER, S.; NGUYEN, V.-H. Efficient sat-encoding of linear csp constraints. In: **International Symposium on Artificial Intelligence and Mathematics**. [*S. l.*: *s. n.*], 2014.

BIERE, A.; BIERE, A.; HEULE, M.; MAAREN, H. van; WALSH, T. **Handbook of Satisfiability**. [*S. l.*]: IOS Press, 2009. v. 185. (Frontiers in Artificial Intelligence and Applications, v. 185). ISBN 1586039296, 9781586039295.

BÜCHI, R. J. Weak second-order arithmetic and finite automata. **Zeitschrift für Mathematische Logik und Grundlagen der Mathematik**, Ann Arbor, Michigan, v. 6, n. 1-6, p. 66–92, 1960.

BUGALHO, M.; OLIVEIRA, A. L. Inference of regular languages using state merging algorithms with search. **Pattern Recognition**, Elsevier, v. 38, n. 9, p. 1457–1467, 2005.

CHANDLEE, J. **Strictly Local Phonological Processes**. Tese (Doutorado) — University of Delaware, 2014.

CHANDLEE, J.; EYRAUD, R.; HEINZ, J. Learning strictly local subsequential functions. **Transactions of the Association for Computational Linguistics**, MIT Press, v. 2, p. 491–504, 2014.

CHANDLEE, J.; EYRAUD, R.; HEINZ, J.; JARDINE, A.; RAWSKI, J. Learning with partially ordered representations. In: **16th Meeting on the Mathematics of Language**. [*S. l.*: *s. n.*], 2019.

CHANDLEE, J.; HEINZ, J. Strict locality and phonological maps. **Linguistic Inquiry**, MIT Press, v. 49, n. 1, p. 23–60, 2018.

CHANDLEE, J.; JARDINE, A. Quantifier-free least fixed point functions for phonology. In: **Proceedings of the 16th Meeting on the Mathematics of Language**. [*S. l.*: *s. n.*], 2019. p. 50–62.

CHUZHANOVA, N. A.; JONES, A. J.; MARGETTS, S. Feature selection for genetic sequence classification. **Bioinformatics (Oxford, England)**, v. 14, n. 2, p. 139–143, 1998.

COOK, S. A. The complexity of theorem-proving procedures. In: **Symposium on Theory of Computing**. [*S. l.*: *s. n.*], 1971. p. 151–158.

COOK, S. A.; LIU, Y. A complete axiomatization for blocks world. **Journal of Logic and Computation**, v. 13, n. 4, p. 581–594, 2003.

CROUCH, M.; IMMERMAN, N.; MOSS, J. E. B. Finding reductions automatically. In: **Fields of Logic and Computation**. [*S. l.*]: Springer, 2010. p. 181–200.

EBBINGHAUS, H.; FLUM, J. **Finite Model Theory**. [*S. l.*]: Springer, 1995. (Perspectives in Mathematical Logic). ISBN 978-3-540-60149-4.

EBBINGHAUS, H.-D.; FLUM, J.; THOMAS, W. **Mathematical Logic**. 2nd. ed. [*S. l.*]: Springer, 1994. I-X, 1-289 p. (Undergraduate texts in mathematics). ISBN 978-3-540-94258-0.

EHRENFEUCHT, A. An application of games to the completeness problem for formalized theories. **Fundamenta Mathematicae**, v. 49, n. 2, p. 129–141, 1961.

FAGIN, R.; STOCKMEYER, L. J.; VARDI, M. Y. On monadic NP vs monadic co-NP. **Information and Computation**, Elsevier, v. 120, n. 1, p. 78–92, 1995.

FELICI, G.; TRUEMPER, K. Lsquare system for mining logic data. In: **Encyclopedia of Data Warehousing and Mining**. [*S. l.*]: IGI Global, 2005. p. 693–697.

FERNAU, H. Modern aspects of complexity within formal languages. In: **International Conference on Language and Automata Theory and Applications**. [*S. l.*]: Springer, 2019. p. 3–30.

FERNAU, H.; HEGGERNES, P.; VILLANGER, Y. A multi-parameter analysis of hard problems on deterministic finite automata. **Journal of Computer and System Sciences**, Elsevier, v. 81, n. 4, p. 747–765, 2015.

FLUM, J.; GROHE, M. **Parameterized Complexity Theory**. [*S. l.*]: Springer Science & Business Media, 2006.

GANZINGER, H.; HAGEN, G.; NIEUWENHUIS, R.; OLIVERAS, A.; TINELLI, C. DPLL(T): Fast decision procedures. In: **International Conference on Computer Aided Verification**. [*S. l.*]: Springer, 2004. p. 175–188.

GARCÍA, P.; LÓPEZ, D.; PARGA, M. V. de. Polynomial characteristic sets for dfa identification. **Theoretical Computer Science**, Elsevier, v. 448, p. 41–46, 2012.

GARCIA, P.; RUIZ, J. Learning k-testable and k-piecewise testable languages from positive data. **Grammars**, v. 7, p. 125–140, 2004.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: Theory and Practice**. [*S. l.*]: Elsevier, 2004.

GHOSH, B.; MEEL, K. S. IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules. In: **Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society**. [S.l.: s.n.], 2019. p. 203–210.

GOLD, E. M. Complexity of automaton identification from given data. **Information and Control**, Elsevier, v. 37, n. 3, p. 302–320, 1978.

GRÄDEL, E.; KOLAITIS, P. G.; LIBKIN, L.; MARX, M.; SPENCER, J.; VARDI, M. Y.; VENEMA, Y.; WEINSTEIN, S. **Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series)**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN 3540004289.

GRIENENBERGER, E.; RITZERT, M. Learning definable hypotheses on trees. In: **22nd International Conference on Database Theory (ICDT 2019)**. [*S. l.*: *s. n.*], 2019.

GROHE, M.; LÖDING, C.; RITZERT, M. Learning MSO-definable hypotheses on strings. In: **International Conference on Algorithmic Learning Theory**. [*S. l.*: *s. n.*], 2017. p. 434–451.

GROHE, M.; RITZERT, M. Learning first-order definable concepts over structures of small degree. In: **2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)**. [*S. l.*]: IEEE, 2017. p. 1–12.

GRÄDEL, E. Why are modal logics so robustly decidable? **Bulletin EATCS**, v. 68, p. 90–103, 1999.

GUPTA, N.; NAU, D. S. On the complexity of blocks-world planning. **Artificial Intelligence**, v. 56, n. 2-3, p. 223–254, 1992.

HAO, Y.; ANDERSSON, S. Unbounded stress in subregular phonology. In: **Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology**. [*S. l.*: *s. n.*], 2019. p. 135–143.

HEINZ, J. Learning quantity insensitive stress systems via local inference. In: **Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology**. [*S. l.*]: Association for Computational Linguistics, 2006. p. 21–30.

HEINZ, J. **Inductive Learning of Phonotactic Patterns**. Tese (Doutorado) — University of California, Los Angeles, 2007.

HEINZ, J. On the role of locality in learning stress patterns. **Phonology**, Cambridge University Press, v. 26, n. 2, p. 303–351, 2009.

HEINZ, J. String extension learning. In: **Proceedings of the Annual Meeting of the Association for Computational Linguistics**. [*S. l.*: *s. n.*], 2010. p. 897–906.

HEINZ, J.; KASPRZIK, A.; KÖTZING, T. Learning in the limit with lattice-structured hypothesis spaces. **Theoretical Computer Science**, Elsevier, v. 457, p. 111–127, 2012.

HEINZ, J.; ROGERS, J. Learning subregular classes of languages with factored deterministic automata. In: **Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)**. [*S. l.*: *s. n.*], 2013. p. 64–71.

HEINZ, J.; SEMPERE, J. M. **Topics in Grammatical Inference**. [*S. l.*]: Springer, 2016. v. 465.

HEULE, M. J.; VERWER, S. Exact DFA identification using SAT solvers. In: **International Colloquium on Grammatical Inference**. [*S. l.*: *s. n.*], 2010. p. 66–79.

HIGUERA, C. D. L. A bibliographical study of grammatical inference. **Pattern Recognition**, Elsevier, v. 38, n. 9, p. 1332–1348, 2005.

HIGUERA, C. De la. **Grammatical Inference: Learning Automata and Grammars**. [*S. l.*]: Cambridge University Press, 2010.

HINTIKKA, J. **Distributive Normal Forms in the Calculus of Predicates**. [*S. l.*]: Edidit Societas Philosophica, 1953. (Acta philosophica Fennica).

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introduction to Automata Theory, Languages, and Computation**. 3. ed. [*S. l.*]: Addison-Wesley, 2006. ISBN 0321455363.

HUTH, M.; RYAN, M. **Logic in Computer Science: Modelling and Reasoning about Systems**. [*S. l.*]: Cambridge University Press, 2004.

IGNATIEV, A.; PEREIRA, F.; NARODYTSKA, N.; MARQUES-SILVA, J. A SAT-based approach to learn explainable decision sets. In: **International Joint Conference on Automated Reasoning**. [*S. l.*: *s. n.*], 2018. p. 627–645.

IMMERMAN, N. **Descriptive complexity**. [*S. l.*]: Springer, 1999.

JI, X.; BAILEY, J.; DONG, G. Mining minimal distinguishing subsequence patterns with gap constraints. **Knowledge and Information Systems**, Springer, v. 11, n. 3, p. 259–286, 2007.

JORDAN, C.; KAISER, Ł. Benchmarks from reduction finding. In: **International Workshop on Quantified Boolean Formulas 2013**. Helsinki, Finland: [*S. n.*], 2013. p. 40–43.

JORDAN, C.; KAISER, Ł. Experiments with reduction finding. In: **International Conference on Theory and Applications of Satisfiability Testing**. [*S. l.*]: Springer, 2013. p. 192–207.

JORDAN, C.; KAISER, Ł. Learning programs as logical queries. In: **Workshop on Learning Theory and Complexity (LTC)**. [*S. l.*: *s. n.*], 2013.

JORDAN, C.; KAISER, Ł. **Machine Learning with Guarantees using Descriptive Complexity and SMT Solvers**. 2016.

KAISER, Ł. Learning games from videos guided by descriptive complexity. In: **26th AAAI Conference on Artificial Intelligence**. [*S. l.*: *s. n.*], 2012.

KAMATH, A. P.; KARMARKAR, N. K.; RAMAKRISHNAN, K.; RESENDE, M. G. A continuous approach to inductive inference. **Mathematical Programming**, Springer, v. 57, n. 1-3, p. 215–238, 1992.

KEISLER, H. J.; LOTFALLAH, W. B. Shrinking games and local formulas. **Annals of Pure and Applied Logic**, Elsevier, v. 128, n. 1-3, p. 215–225, 2004.

KHOUSSAINOV, B.; LIU, J. On complexity of ehrenfeucht–fraïssé games. **Annals of Pure and Applied Logic**, v. 161, n. 3, p. 404 – 415, 2009. ISSN 0168-0072.

KNIJNENBURG, T. A.; KLAU, G. W.; IORIO, F.; GARNETT, M. J.; MCDERMOTT, U.; SHMULEVICH, I.; WESSELS, L. F. Logic models to predict continuous outputs based on binary inputs with an application to personalized cancer therapy. **Scientific reports**, Nature Publishing Group, [*S. l.*], v. 6, n. 1, p. 1–14, 2016.

LADNER, R. E. Application of model theoretic games to discrete linear orders and finite automata. **Information and Control**, v. 33, n. 4, p. 281 – 303, 1977. ISSN 0019-9958.

LAMBERT, D.; ROGERS, J. A logical and computational methodology for exploring systems of phonotactic constraints. **Proceedings of the Society for Computation in Linguistics**, v. 2, n. 1, p. 247–256, 2019.

LEJEUNE, M.; LOZIN, V.; LOZINA, I.; RAGAB, A.; YACOUT, S. **Recent Advances in the Theory and Practice of Logical Analysis of Data**. [*S. l.*]: Elsevier, 2018. European Journal of Operational Research.

LESH, N.; ZAKI, M. J.; OGIHARA, M. Mining features for sequence classification. In: **Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [*S. l.*: *s. n.*], 1999. v. 99, p. 342–346.

LIBKIN, L. **Elements Of Finite Model Theory**. [*S. l.*]: Springer, 2004. ISBN 3540212027.

LUCAS, S. M.; REYNOLDS, T. J. Learning DFA: evolution versus evidence driven state merging. In: **Congress on Evolutionary Computation**. [*S. l.*: *s. n.*], 2003. p. 351–358.

MALIOUTOV, D.; MEEL, K. S. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In: **International Conference on Principles and Practice of Constraint Programming**. [*S. l.*]: Springer, 2018. p. 312–327.

MARIA, E. D.; MONTANARI, A.; VITACOLONNA, N. Games on strings with a limited order relation. In: **International Symposium on Logical Foundations of Computer Science**. [*S. l.*: *s. n.*], 2009. p. 164–179.

MONTANARI, A.; POLICRITI, A.; VITACOLONNA, N. An algorithmic account of Ehrenfeucht games on labeled successor structures. In: **International Conference on Logic for Programming Artificial Intelligence and Reasoning**. [*S. l.*]: Springer, 2005. p. 139–153.

MUGGLETON, S. Inductive logic programming. **New Generation Computing**, Springer, v. 8, n. 4, p. 295–318, 1991.

MUGGLETON, S.; RAEDT, L. D. Inductive logic programming: Theory and methods. **The Journal of Logic Programming**, Elsevier, v. 19, p. 629–679, 1994.

NEIDER, D.; GAVRAN, I. Learning linear temporal properties. In: **Formal Methods in Computer-Aided Design**. [*S. l.*: *s. n.*], 2018. p. 1–10.

NIEUWENHUIS, R.; OLIVERAS, A. DPLL(T) with exhaustive theory propagation and its application to difference logic. In: **International Conference on Computer Aided Verification**. [*S. l.*]: Springer, 2005. p. 321–334.

NIEUWENHUIS, R.; OLIVERAS, A.; TINELLI, C. Abstract DPLL and abstract DPLL modulo theories. In: **International Conference on Logic for Programming Artificial Intelligence and Reasoning**. [*S. l.*]: Springer, 2005. p. 36–50.

NIEUWENHUIS, R.; OLIVERAS, A.; TINELLI, C. Solving SAT and SAT modulo theories: from an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). **Journal of the ACM (JACM)**, ACM, v. 53, n. 6, p. 937–977, 2006.

ONCINA, J.; GARCíA, P. Inferring regular languages in polynomial updated time. In: **Pattern Recognition and Image Analysis**. [*S. l.*: *s. n.*], 1992. p. 49–61.

PEZZOLI, E. Computational complexity of Ehrenfeucht-Fraïssé games on finite structures. In: **International Workshop on Computer Science Logic**. [*S. l.*]: Springer, 1998. p. 159–170.

RAEDT, L. D. **Logical and Relational Learning**. [*S. l.*]: Springer, 2008.

RAWAL, C.; TANNER, H. G.; HEINZ, J. (Sub)regular robotic languages. In: **Mediterranean Conference on Control & Automation**. [*S. l.*: *s. n.*], 2011. p. 321–326.

ROCHA, T. A.; MARTINS, A. T. Synthesis of quantifier-free first-order sentences from noisy samples of strings. In: **2019 8th Brazilian Conference on Intelligent Systems (BRACIS)**. [*S. l.*]: IEEE, 2019. p. 12–17.

ROCHA, T. A.; MARTINS, A. T.; FERREIRA, F. M. On finding a first-order sentence consistent with a sample of strings. **Electronic Proceedings in Theoretical Computer Science**, Open Publishing Association, v. 277, p. 220–234, sep 2018.

ROCHA, T. A.; MARTINS, A. T.; FERREIRA, F. M. Synthesis of a DNF formula from a sample of strings. In: **7th Brazilian Conference on Intelligent Systems**. [*S. l.*]: IEEE, 2018. p. 534–539.

ROCHA, T. A.; MARTINS, A. T.; FERREIRA, F. M. On distinguishing sets of structures by first-order sentences of minimal quantifier rank. **Electronic Notes in Theoretical Computer Science**, Elsevier, v. 344, p. 189–208, 2019.

ROCHA, T. A.; MARTINS, A. T.; FERREIRA, F. M. Synthesis of a DNF formula from a sample of strings using Ehrenfeucht–Fraïssé games. **Theoretical Computer Science**, Elsevier, v. 805, p. 109–126, 2020.

ROGERS, J.; HEINZ, J.; FERO, M.; HURST, J.; LAMBERT, D.; WIBEL, S. Cognitive and sub-regular complexity. In: **Formal Grammar**. [*S. l.*]: Springer Berlin Heidelberg, 2013. p. 90–108. ISBN 978-3-642-39998-5.

ROZENBERG, G.; SALOMAA, A. **Handbook of Formal Languages**. Berlin, Heidelberg: Springer, 1997. v. 3.

SCHWENTICK, T. On winning Ehrenfeucht games and monadic NP. **Annals of Pure and Applied Logic**, Elsevier, v. 79, n. 1, p. 61–92, 1996.

STANISLAWSKI, J.; KOTULSKA, M.; UNOLD, O. Machine learning methods can replace 3d profile method in classification of amyloidogenic hexapeptides. **BMC Bioinformatics**, BioMed Central, v. 14, n. 1, p. 21, 2013.

STOCKMEYER, L. J. **The Complexity of Decision Problems in Automata Theory and Logic**. Tese (Doutorado) — Massachusetts Institute of Technology, 1974.

STROTHER-GARCIA, K. Imdlawn tashlhiyt berber syllabification is quantifier-free. **Proceedings of the Society for Computation in Linguistics (SCiL) 2018**, p. 145–153, 2018.

STROTHER-GARCIA, K.; HEINZ, J.; HWANGBO, H. J. Using model theory for grammatical inference: a case study from phonology. In: **International Conference on Grammatical Inference**. [*S. l.*: *s. n.*], 2017. p. 66–78.

THOMAS, W. Classifying regular events in symbolic logic. **Journal of Computer and System Sciences**, Elsevier, v. 25, n. 3, p. 360–376, 1982.

TRIANTAPHYLLOU, E. **Data Mining and Knowledge Discovery via Logic-based Methods**. [*S. l.*]: Springer, 2010. v. 43.

ULYANTSEV, V.; ZAKIRZYANOV, I.; SHALYTO, A. BFS-based symmetry breaking predicates for DFA identification. In: **9th International Conference on Language and Automata Theory and Applications**. [*S. l.*: *s. n.*], 2015. p. 611–622.

UMANS, C.; VILLA, T.; SANGIOVANNI-VINCENTELLI, A. L. Complexity of two-level logic minimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 25, n. 7, p. 1230–1246, 2006.

VARDI, M. Y. The complexity of relational query languages. In: **Proceedings of the 14th Annual ACM Symposium on Theory of Computing**. [*S. l.*]: ACM, 1982. p. 137–146.

VU, M. H.; ZEHFROOSH, A.; STROTHER-GARCIA, K.; SEBOK, M.; HEINZ, J.; TANNER, H. G. Statistical relational learning with unconventional string models. **Frontiers in Robotics and AI**, Frontiers, v. 5, p. 76, 2018.

WIECZOREK, W. **Grammatical Inference: Algorithms, Routines and Applications**. [*S. l.*]: Springer, 2016. v. 673.

WIECZOREK, W.; UNOLD, O. Induction of directed acyclic word graph in a bioinformatics task. In: **International Conference on Grammatical Inference**. [*S. l.*: *s. n.*], 2014. p. 207–217.

WIECZOREK, W.; UNOLD, O. Use of a novel grammatical inference approach in classification of amyloidogenic hexapeptides. **Computational and Mathematical Methods in Medicine**, Hindawi, v. 2016, 2016.

WIECZOREK, W.; UNOLD, O. GP-based grammatical inference for classification of amyloidogenic sequences. In: **International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics**. [*S. l.*]: Springer, 2017. p. 81–91.

WIECZOREK, W.; UNOLD, O.; STRĄK, Ł. Suffix classification trees. In: **International Conference on Grammatical Inference**. [*S. l.*: *s. n.*], 2019. p. 44–53.

XING, Z.; PEI, J.; KEOGH, E. A brief survey on sequence classification. **ACM Sigkdd Explorations Newsletter**, ACM, v. 12, n. 1, p. 40–48, 2010.

ZAKIRZYANOV, I.; MORGADO, A.; IGNATIEV, A.; ULYANTSEV, V.; MARQUES-SILVA, J. Efficient symmetry breaking for SAT-based minimum DFA inference. In: **13th International Conference on Language and Automata Theory and Applications**. [*S. l.*: *s. n.*], 2019. p. 159–173.

ZAKIRZYANOV, I.; SHALYTO, A.; ULYANTSEV, V. Finding all minimum-size DFA consistent with given examples: SAT-based approach. In: **International Conference on Software Engineering and Formal Methods**. [*S. l.*]: Springer, 2017. p. 117–131.

ZEITOUN, M.; ROOIJEN, L. van; PLACE, T. On separation by locally testable and locally threshold testable languages. **Logical Methods in Computer Science**, v. 10, n. 3, p. 24, 2014.