Patterns with bounded treewidth [internal report]

Internal Report 1109
Department of Computer Science
Loughborough University

Patterns with Bounded Treewidth

Daniel Reidenbach
Markus L. Schmid

June 2012

Patterns with Bounded Treewidth*

Daniel Reidenbach

Markus L. Schmid†

Department of Computer Science, Loughborough University,
Loughborough, Leicestershire, LE11 3TU, United Kingdom
{D.Reidenbach, M.Schmid}@lboro.ac.uk

Abstract

A pattern is a string consisting of variables and terminal symbols, and its language is the set of all words that can be obtained by substituting arbitrary words for the variables. The membership problem for pattern languages, i. e., deciding on whether or not a given word is in the pattern language of a given pattern is NP-complete. We show that any parameter of patterns that is an upper bound for the treewidth of appropriate encodings of patterns as relational structures, if restricted, allows the membership problem for pattern languages to be solved in polynomial time. Furthermore, we identify new such parameters.

1 Introduction

A *pattern* α is a finite string that consists of *variables* and *terminal symbols* (taken from a fixed alphabet Σ), and its language is the set of all words that can be derived from α when substituting arbitrary words over Σ for the variables. For example, the language L generated by the pattern $\alpha := x_1\mathtt{a}x_2\mathtt{b}x_1$ (where x_1, x_2 are variables and $\mathtt{a}, \mathtt{b}$ are terminal symbols) consists of all words with an arbitrary prefix u, followed by the letter $\mathtt{a}$, an arbitrary word v, the letter $\mathtt{b}$ and a suffix that equals the prefix u. Thus, $w_1 := \mathtt{aaabbaa}$ is contained in L, whereas $w_2 := \mathtt{baaba}$ is not.

Patterns provide a compact and natural way to describe formal languages. In their original definition given by Angluin [3] variables can only be substituted by non-empty words; hence, the term *nonerasing pattern languages* (or, for short, *NE-pattern languages*) is used. *Extended* or *erasing pattern languages* (or, for short, *E-pattern languages*) where variables can also be substituted by the empty word have been introduced by Shinohara [23]. The original motivation of pattern

*A preliminary version of this work was presented at LATA 2012

†Corresponding author

languages (cf. Angluin [3]) is derived from *inductive inference*, i. e., the task of inferring a pattern from any given sequence of all words in its pattern language, for which numerous results can be found in the literature (see, e. g., Angluin [3], Shinohara [23], Lange and Wiehagen [13], Rossmanith and Zeugmann [22], Reidenbach [16, 17] and, for a survey, Ng and Shinohara [15]). On the other hand, due to their simple definition, pattern languages have connections to many areas of theoretical computer science, e. g., formal language theory, learning theory, combinatorics on words and pattern matching, and their general properties have been investigated in various contexts (for a survey, see, e. g., Mateescu and A. Salomaa [14]).

With respect to practical applications, their membership problem, i. e., the problem of deciding, for a given word w and a pattern α, whether or not the variables of α can be substituted in such a way that w is obtained, is probably the most important aspect of pattern languages. This is mainly due to the connection to so-called *extended regular expressions with backreferences* (*REGEX* for short) (see, e. g., Câmpeanu et al. [7]). REGEX can roughly be considered as classical regular expressions that are equipped with the possibility to define backreferences, i. e., to require factors to be repeated at several defined positions in the word; hence, backreferences correspond to the variables in patterns. While backreferences dramatically increase the expressive power of classical regular expressions, they are also responsible for the membership problem of this language class to become NP-complete. This is particularly worth mentioning as today's text editors and programming languages (such as Perl, Python, Java, etc.) all provide so-called *REGEX engines* that compute the solution to the membership problem for any language given by a REGEX and an arbitrary string. Hence, despite its theoretical intractability, algorithms that perform the match test for REGEX are a practical reality. While pattern languages merely describe a proper subset of REGEX languages, they cover what is computationally hard, i. e., the concept of backreferences. Hence, investigating the membership problem for pattern languages helps to improve algorithms solving the match test for extended regular expressions with backreferences.

The membership problem for pattern languages, that was first shown to be NP-complete by Angluin [3] in 1980, can also be considered as some kind of pattern matching task, since we have to decide whether or not a given word satisfies a given pattern. In fact, this pattern matching aspect of pattern languages, independently from Angluin's work, has recently been rediscovered in the pattern matching community in terms of so-called *parameterised pattern matching*, where a text is not searched for all occurrences of a specific factor, but for all occurrences of factors that satisfy a given pattern with parameters (i. e., variables). In the original version of parameterised pattern matching introduced by Baker [4], variables in the pattern can only be substituted by single symbols and, furthermore, the substitution must be injective, i. e., different variables cannot be substituted by the same symbol. Amir et al. [1] then generalised this problem by dropping the injectivity condition and Amir and Nor [2] added the possibility of substituting variables by words instead of single symbols and they also allowed "don't care" symbols to be used in addition to variables. In 2009,

Clifford et al. [8] considered parameterised pattern matching as introduced by Amir and Nor, but without "don't care" symbols, which led to patterns as introduced by Angluin. In [2], motivations for the membership problem of pattern languages can be found from such diverse areas as software engineering, image searching, DNA analysis, poetry and music analysis, or author validation.

Our main research task is to identify parameters of patterns that, if restricted to a constant, allow a polynomial time membership problem. The benefit of finding such parameters is twofold. Firstly, we can learn what properties of a pattern are actually responsible for the complexity of the membership problem, i. e., we achieve a refined complexity analysis of this problem. Secondly, restricting these parameters is likely to lead to improved algorithms for the membership problem of pattern languages. The first such parameter that comes to mind is the number of different variables in a pattern. Its restriction constitutes a trivial way to obtain a polynomial time membership problem, since the brute force algorithm that simply enumerates all possibilities to substitute the variables by terminal words in order to check whether the input word can be obtained is exponential in the number of variables (for a detailed complexity analysis see Ibarra et al. [12]). Nevertheless, the number of variables is a central parameter of patterns and important results about the learnability of pattern languages (see Angluin [3] and Reischuk and Zeugmann [21]) as well as recent results about the inclusion problem of pattern languages (see Bremer and Freydenberger [6]) are concerned with patterns with a restricted number of variables. Moreover, Stephan et al. [25] investigate the parameterised complexity of the membership problem for pattern languages and they show that it is fixed parameter intractable, if parameterised by the number of variables. The membership problem for pattern languages given by patterns with only one occurrence per variable (introduced by Shinohara [24]) is solvable in polynomial time, simply because these patterns describe regular languages; hence, they are called *regular* patterns. If the patterns are unrestricted, then the membership problem can still be solved in polynomial time provided that the length of the input word is bounded (see Geilke and Zilles [11]).

The arguably first nontrivial restriction of patterns that allow a polynomial time membership problem are Shinohara's *non cross* patterns [24], i. e., patterns where between any two occurrences of the same variable x no other variable different from x occurs. However, this result does not provide a structural parameter of patterns that can be considered to contribute to the complexity of the membership problem. Recently, in [19], an automata based approach has been used in order to extend Shinohara's result to an infinite hierarchy of classes of pattern languages with a polynomial time membership problem. The idea in [19] is to restrict a rather subtle parameter, namely the *distance* several occurrences of any variable x may have in a pattern (i. e., the maximum number of different variables separating any two consecutive occurrences of x). This parameter is called the *variable distance* vd of a pattern α, and in [19] it is demonstrated that the membership problem is solvable in time $\mathrm{O}(|\alpha|^3 \times |w|^{(\mathrm{vd}(\alpha)+4)})$, so it is exponential only in the variable distance.

In this work, we approach the problem of identifying such parameters in

a novel and quite general way. More precisely, we encode patterns and words as relational structures and, thus, reduce the membership problem to the homomorphism problem for relational structures. Our main result is that any parameter of patterns that is an upper bound for the treewidth of the corresponding relational structures, if restricted to a constant, allows the membership problem to be solved in polynomial time. In this new framework, we can restate the known results about the complexity of the membership problem mentioned above, as well as identifying new and, compared to the old results, rather large classes of patterns with a polynomial time membership problem. Therefore, we provide a convenient way to study the membership problem for pattern languages, which, as shall be pointed out by our results, has still potential for further improvements.

2 Preliminaries

Let $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$. For an arbitrary alphabet A, a *string* (*over* A) is a finite sequence of symbols from A, and ε stands for the *empty string*. The notation A^+ denotes the set of all nonempty strings over A, and $A^* := A^+ \cup \{\varepsilon\}$. For the *concatenation* of two strings w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. We say that a string $v \in A^*$ is a *factor* of a string $w \in A^*$ if there are $u_1, u_2 \in A^*$ such that $w = u_1 \cdot v \cdot u_2$. If u_1 or u_2 is the empty string, then v is a *prefix* (or a *suffix*, respectively) of w. The notation $|K|$ stands for the size of a set K or the length of a string K. If we wish to refer to the symbol at a certain position j, $1 \leq j \leq n$, in a string $w = \mathtt{a}_1 \cdot \mathtt{a}_2 \cdot \cdots \cdot \mathtt{a}_n$, $\mathtt{a}_i \in A$, $1 \leq i \leq n$, then we use $w[j] := \mathtt{a}_j$ and if the length of a string is unknown, then we denote its last symbol by $w[-] := w[|w|]$. Furthermore, for each j, j', $1 \leq j < j' \leq |w|$, let $w[j, j'] := \mathtt{a}_j \cdot \mathtt{a}_{j+1} \cdot \cdots \cdot \mathtt{a}_{j'}$ and $w[j, -] := w[j, |w|]$.

Pattern Languages and Parameters of Patterns

For any alphabets A, B, a *morphism* is a function $h : A^* \to B^*$ that satisfies $h(vw) = h(v)h(w)$ for all $v, w \in A^*$; h is said to be *nonerasing* if and only if, for every $a \in A$, $h(a) \neq \varepsilon$. Let Σ be a finite alphabet of so-called *terminal symbols* and let X be a countably infinite set of *variables* with $\Sigma \cap X = \emptyset$. We normally assume $X := \{x_1, x_2, x_3, \ldots\}$. A *pattern* is a nonempty string over $\Sigma \cup X$ and a *word* is a string over Σ. For any pattern α, we refer to the set of variables in α as $\mathrm{var}(\alpha)$ and for any variable $x \in \mathrm{var}(\alpha)$, $|\alpha|_x$ denotes the number of occurrences of x in α.

A morphism $h : (\Sigma \cup X)^* \to \Sigma^*$ is called a *substitution* if $h(a) = a$ for every $a \in \Sigma$. We define the *E-pattern language* of a pattern α by $L_{\mathrm{E},\Sigma}(\alpha) := \{h(\alpha) \mid h : (\Sigma \cup X)^* \to \Sigma^* \text{ is a substitution}\}$. The *NE-pattern language* $L_{\mathrm{NE},\Sigma}(\alpha)$ of α is defined analogously, just with respect to nonerasing substitutions. Since in this work the impact of the choice of the alphabet Σ is negligible, we mostly denote pattern languages by $L_{\mathrm{E}}(\alpha)$ and $L_{\mathrm{NE}}(\alpha)$.

The problem of deciding, for a given pattern α and a given word $w \in \Sigma^*$,

whether $w \in L_{\mathrm{E}}(\alpha)$ (or $w \in L_{\mathrm{NE}}(\alpha)$) is called the *membership problem for* E-*pattern languages* (or NE-*pattern languages*, respectively). For every class of patterns $C \subseteq (\Sigma \cup X)^*$ and every $Z \in \{\mathrm{E}, \mathrm{NE}\}$, Z-PATMem(C) denotes the membership problem for Z-pattern languages where the patterns are restricted to the class C.

We now formally define the already mentioned variable distance of a pattern, which has been introduced in [19]. Informally speaking, the variable distance is the maximum number of different variables separating any two consecutive occurrences of a variable:

Definition 1. *The* variable distance *of a pattern* α *(*$\mathrm{vd}(\alpha)$*) is the smallest number* $k \geq 0$ *such that, for every* $x \in \mathrm{var}(\alpha)$*, every factorisation* $\alpha = \beta \cdot x \cdot \gamma \cdot x \cdot \delta$ *with* $\beta, \gamma, \delta \in (\Sigma \cup X)^*$ *and* $|\gamma|_x = 0$ *satisfies* $|\mathrm{var}(\gamma)| \leq k$*.*

For example, $\mathrm{vd}(x_1x_2x_3x_2\mathtt{a}x_3x_1x_4\mathtt{b}x_3x_5x_5x_4) = 2$. Obviously, $\mathrm{vd}(\alpha) \leq |\mathrm{var}(\alpha)| - 1$ for all patterns α.

The concept of the scope coincidence degree has already been introduced in [18]. However, here we shall define it in a slightly different (yet equivalent) way.

Definition 2. *Let* α *be a pattern. For every* $y \in \mathrm{var}(\alpha)$*, the* scope of y in α *is defined by* $\mathrm{sc}_\alpha(y) := \{i, i+1, \ldots, j\}$*, where* i *is the leftmost and* j *the rightmost position of* y *in* α*. The scopes of* $y_1, y_2, \ldots, y_k \in \mathrm{var}(\alpha)$ coincide in α *if and only if* $\bigcap_{1 \leq i \leq k} \mathrm{sc}_\alpha(y_i) \neq \emptyset$*. The* scope coincidence degree *of* α *(*$\mathrm{scd}(\alpha)$*) is the maximum number of variables in* α *such that their scopes coincide.*

Let $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and let the pattern $\beta \in (\Sigma \cup X)^*$ be given by $\beta := x_1\mathtt{b}x_2\mathtt{a}x_1x_3x_2\mathtt{ab}x_3x_4x_2x_4x_5\mathtt{bcb}x_1x_4x_5$. Then each set $\{x_1, x_2, x_3\}$, $\{x_1, x_2, x_4\}$ and $\{x_1, x_4, x_5\}$ contains variables the scopes of which coincide, but there does not exist a set of more than 3 variables with the same property. This directly implies $\mathrm{scd}(\beta) = 3$.

It is straightforward to see that the scope coincidence degree as well as the variable distance can be computed in time linear in the length of the pattern. The following lemma relates the variable distance and the scope coincidence degree.

Lemma 3. *Let* α *be a pattern. Then* $\mathrm{scd}(\alpha) \leq \mathrm{vd}(\alpha) + 1$*.*

Proof. Let $\mathrm{scd}(\alpha) = k$, which, by definition, implies that, for k distinct variables $y_1, y_2, \ldots, y_k \in \mathrm{var}(\alpha)$, $\bigcap_{1 \leq i \leq k} \mathrm{sc}_\alpha(y_i) \neq \emptyset$. Furthermore, this implies that there exists a p, $1 \leq p \leq k$, such that α can be factorised into $\alpha = \beta \cdot y_p \cdot \gamma$ with $(\{y_1, y_2, \ldots, y_k\}/\{y_p\}) \subseteq (\mathrm{var}(\beta) \cap \mathrm{var}(\gamma))$. Now let q, $1 \leq q \leq k$, $q \neq p$, be such that β can be factorised into $\beta = \beta' \cdot y_q \cdot \beta''$ with $(\{y_1, y_2, \ldots, y_k\}/\{y_p, y_q\}) \subseteq \mathrm{var}(\beta'')$ and $y_q \notin \beta''$. Since there is an occurrence of y_q in γ, γ can be factorised into $\gamma = \gamma' \cdot y_q \cdot \gamma''$ with $|\gamma'|_{y_q} = 0$. Hence, α contains the factor $y_q \cdot \beta'' \cdot y_p \cdot \gamma' \cdot y_q$, where $|\beta'' \cdot y_p \cdot \gamma'|_{y_q} = 0$ and $(\{y_1, y_2, \ldots, y_k\}/\{y_q\}) \subseteq \mathrm{var}(\beta'' \cdot y_p \cdot \gamma')$, which implies $\mathrm{vd}(\alpha) \geq k - 1 = \mathrm{scd}(\alpha) - 1$. □

On the other hand, the variable distance cannot be bounded in terms of the scope coincidence degree since, for example, all patterns of form $x_1 \cdot x_2 \cdot x_3 \cdot$

$\ldots \cdot x_k \cdot x_{k+1} \cdot x_1$, $k \in \mathbb{N}$, have a variable distance of k, but a constant scope coincidence degree of 2.

Relational Structures, Treewidth and Homomorphism Problem

For the sake of completeness, we give the following standard definitions very briefly. For a comprehensive description, the reader is referred to Chapters 4, 11 and 13 of Flum and Grohe [9].

A *(relational) vocabulary* τ is a finite set of relation symbols. Every relation symbol $R \in \tau$ has an *arity* $\mathrm{ar}(R) \geq 1$. A *τ-structure* (or simply *structure*), comprises a finite set A called the *universe* and, for every $R \in \tau$, an *interpretation* $R^{\mathcal{A}} \subseteq A^{\mathrm{ar}(R)}$. For example, every graph can be given as a relational structure over a vocabulary with one binary relation symbol representing the edges. Let $\mathcal{A}$ and $\mathcal{B}$ be structures of the same vocabulary τ with universes A and B, respectively. A *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a mapping $h : A \to B$ such that for all $R \in \tau$ and for all $a_1, a_2, \ldots, a_{\mathrm{ar}(R)} \in A$, $(a_1, a_2, \ldots, a_{\mathrm{ar}(R)}) \in R^{\mathcal{A}}$ implies $(h(a_1), h(a_2), \ldots, h(a_{\mathrm{ar}(R)})) \in R^{\mathcal{B}}$.

Next, we introduce the concepts of *tree decompositions* and *treewidth* of a graph (see Chapter 11 of Flum and Grohe [9]).

Definition 4. *A* tree decomposition *of a graph $\mathcal{G} := (V, E)$ is a pair $(\mathcal{T}, \{B_t \mid t \in T\})$, where $\mathcal{T} := (T, F)$ is a tree and the B_t, $t \in T$, are subsets of V such that the following is satisfied:*

1. *For every $v \in V$, the set $\{t \in T \mid v \in B_t\}$ is nonempty and connected in $\mathcal{T}$.*
2. *For every edge $\{u, v\} \in E$ there is a $t \in T$ such that $\{u, v\} \subseteq B_t$.*

The width *of the tree decomposition $(\mathcal{T}, \{B_t \mid t \in T\})$ is the number $\max\{|B_t| \mid t \in T\} - 1$. The* treewidth of $\mathcal{G}$ (denoted by $\mathrm{tw}(\mathcal{G})$) *is the minimum of the widths of the tree decompositions of $\mathcal{G}$.*

A tree decomposition, the underlying tree of which is a path, is also called a *path decomposition* and the *pathwidth* of a graph $\mathcal{G}$ (denoted by $\mathrm{pw}(\mathcal{G})$) is defined as the treewidth, just with respect to path decompositions. For the sake of convenience, we shall denote a path decomposition as a sequence $(B_1, B_2, \ldots, B_k)$ of sets of vertices without the component of the tree $\mathcal{T}$. Obviously, $\mathrm{tw}(\mathcal{G}) \leq \mathrm{pw}(\mathcal{G})$.

Tree decompositions for general τ-structures are defined in a similar way as for graphs, with the difference that the sets B_t contain now elements from the universe A of the structure instead of vertices. Furthermore, analogously as for tree decompositions of graphs, the sets $\{t \in T \mid a \in B_t\}$, $a \in A$, must be nonempty and connected in $\mathcal{T}$, but instead of requiring each edge to be represented in some B_t, we require that, for every relation symbol $R \in \tau$ and every tuple $(a_1, \ldots, a_{\mathrm{ar}(R)}) \in R^{\mathcal{A}}$ there is a $t \in T$ such that $a_1, \ldots, a_{\mathrm{ar}(R)} \in B_t$ (see Chapter 11 of Flum and Grohe [9] for a detailed definition). Path decompositions, the treewidth and the pathwidth of relational structures are

also defined in an analogous way as for graphs. Tree decompositions of relational structures can also be characterised in terms of classical graphs. To this end, we need the concept of the *Gaifman graph* of a τ-structure $\mathcal{A}$, which is the graph that has the universe A of $\mathcal{A}$ as vertices and two vertices are connected if and only if they occur together in some relation (see Chapter 11 of Flum and Grohe [9]).

Proposition 5. *A relational structure has the same tree decompositions as its Gaifman graph.*

The previous proposition particularly implies that the treewidth of a structure equals the treewidth of its Gaifman graph. Thus, the Gaifman graph provides a convenient means to handle tree decompositions and the treewidth of structures. We say that a class of structures C has bounded treewidth if and only if there exists a $k \in \mathbb{N}$ such that, for every $\mathcal{A} \in C$, $\mathrm{tw}(\mathcal{A}) \leq k$.

The *homomorphism problem HOM* is the problem to decide, for given structures $\mathcal{A}$ and $\mathcal{B}$, whether there exists a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. For any set of structures C, by $\mathrm{HOM}(C)$ we denote the homomorphism problem that is restricted in such a way that the left hand input structure is from C. If C is a class of structures with bounded treewidth, then $\mathrm{HOM}(C)$ can be solved in polynomial time. This is a classical result that has been first achieved in terms of *constraint satisfaction problems* by Freuder [10] (see also Chapter 13 of Flum and Grohe [9]).

Theorem 6 (Freuder [10])**.** *Let C be a set of structures with bounded treewidth. HOM(C) is solvable in polynomial time.*

We shall briefly sketch how a tree decomposition of a structure $\mathcal{A}$ can be used in order to decide on whether or not there exists a homomorphism from $\mathcal{A}$ to another structure $\mathcal{A}'$. The naive way of deciding on the existence of a homomorphism is to simply enumerate all possible mappings from A to A', the universes of structures $\mathcal{A}$ and $\mathcal{A}'$, respectively, and check whether or not one of them satisfy the homomorphism condition. However, with a tree decomposition $(\mathcal{T} := (T, F), \{B_t \mid t \in T\})$ of $\mathcal{A}$, for every $t \in T$, we can first compute all mappings from B_t to A' that satisfy the homomorphism condition with respect to the elements in B_t. Then, by inductively merging these partial mappings according to the tree structure $\mathcal{T}$, we can construct a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$ if one exists. The correctness of this last step is provided by the conditions stating that, for every $a \in A$, $\{t \in T \mid a \in B_t\}$ is nonempty and connected in $\mathcal{T}$ and, for every relation symbol $R \in \tau$ and every tuple $(a_1, \ldots, a_{\mathrm{ar}(R)}) \in R^{\mathcal{A}}$ there is a $t \in T$ such that $a_1, \ldots, a_{\mathrm{ar}(R)} \in B_t$. In this procedure, we do not need to enumerate complete mappings, but only mappings for a number of elements that is bounded by the width of the tree decomposition. Hence, the time complexity of this approach is exponential only in the treewidth.

3 Patterns and Words as Relational Structures

In this section, we introduce a way of representing patterns and terminal words as relational structures. Our overall goal is to reduce the membership problem for pattern languages to the homomorphism problem for relational structures.

Representing words as relational structures is a common technique when mathematical logic is applied to language theory (see, e. g., Thomas [26] for a survey). However, our representations of patterns and words by structures substantially differ from the standard technique, since our approach is tailored to the homomorphism problem of structures and, furthermore, we want to exploit the treewidth.

In order to encode patterns and terminal words, i. e., an instance of the membership problem for pattern languages, we use the relational vocabulary $\tau_\Sigma := \{E, S, L, R\} \cup \{D_b \mid b \in \Sigma\}$, where E, S are binary relations and L, R, D_b, $b \in \Sigma$, are unary relations. The vocabulary depends on Σ, the alphabet under consideration. In order to represent a pattern α by a τ_Σ-structure, we interpret the set of positions of α as the universe. The roles of S, L, R and D_b, $b \in \Sigma$, are straightforward: S relates adjacent positions, L and R denote the leftmost and rightmost position, respectively, and, for every $b \in \Sigma$, the relation D_b contains the positions in α where the terminal symbol b occurrs. For the encoding of the variables, we do not explicitly store their positions in the pattern, which seems impossible, since the number of different variables can be arbitrarily large and we can only use a finite number of relation symbols. Instead, we use the relation E in order to record pairs of positions where the same variable occurs and, furthermore, this is done in a "sparse" way. More precisely, the relation E relates *some* positions with the same variable, i. e., positions i, j with $\alpha[i] = \alpha[j]$, in such a way that the symmetric transitive closure of E contains *all* pairs (i, j) with $\alpha[i] = \alpha[j]$ and $\alpha[i] \in X$. This way of interpreting the relation E is crucial for our results.

We now state the formal definition and shall illustrate it afterwards.

Definition 7. *Let α be a pattern and let $\mathcal{A}_\alpha$ be a τ_Σ-structure. $\mathcal{A}_\alpha$ is an* α-structure *if it has universe $A_\alpha := \{1, 2, \ldots, |\alpha|\}$ and $S^{\mathcal{A}_\alpha} := \{(i, i+1) \mid 1 \leq i \leq |\alpha|-1\}$, $L^{\mathcal{A}_\alpha} := \{1\}$, $R^{\mathcal{A}_\alpha} := \{|\alpha|\}$, for every $b \in \Sigma$, $D_b^{\mathcal{A}_\alpha} := \{i \mid \alpha[i] = b\}$, and $E^{\mathcal{A}_\alpha}$ is such that, for all $i, j \in A_\alpha$,*

- *$(i, j) \in E^{\mathcal{A}_\alpha}$ implies $\alpha[i] = \alpha[j]$ and $i \neq j$,*
- *$\alpha[i] = \alpha[j]$ implies that (i, j) is in the symmetric transitive closure of $E^{\mathcal{A}_\alpha}$.*

Since τ_Σ contains only unary and binary relation symbols, it is straightforward to derive the Gaifman graph from an α-structure, which is simply a graph with two different kinds of edges due to $S^{\mathcal{A}_\alpha}$ and $E^{\mathcal{A}_\alpha}$. Hence, in the following, we shall switch between these two models at our convenience without explicitly mentioning it. In the previous definition, the universe as well as the interpretations for the relation symbols S, L, R and D_b, $b \in \Sigma$, are uniquely defined for a fixed pattern α, while there are several possibilities of defining

an interpretation of E. Intuitively, a valid interpretation of E is created by connecting different occurrences of the same variable by edges in such a way that all the occurrences of some variable describe a connected component. The simplest way of doing this is to add an edge between *any two* occurrences of the same variable, i. e., $E^{\mathcal{A}_\alpha} := \{(i,j) \mid \alpha[i] = \alpha[j]\}$. However, we shall see that for our results the interpretation of E is crucial and using the one just mentioned is not advisable. Another example of a valid interpretation of E is the following one. For every $x \in \mathrm{var}(\alpha)$, let l_x be the leftmost occurrence of x in α. Defining $E^{\mathcal{A}_\alpha} := \bigcup_{x \in \mathrm{var}(\alpha)} \{(l_x, i) \mid l_x < i \leq |\alpha|, \alpha[i] = x\}$ yields another possible α-structure.

Next, we define a canonical α-structure, i. e., the interpretation of E is such that every occurrence of a variable x at position i is connected to the next occurrence of x to the right of position i.

Definition 8. *Let α be a pattern. The* standard α-structure $(\mathcal{A}^s_\alpha)$ *is the α-structure where $E^{\mathcal{A}^s_\alpha} := \{(i,j) \mid 1 \leq i < j \leq |\alpha|, \exists\, x \in X$ such that $x = \alpha[i] = \alpha[j]$ and $\alpha[k] \neq x, i < k < j\}$.*

As an example, we consider the standard α-structure $\mathcal{A}^s_\alpha$ for the pattern $\alpha := x_1\mathtt{a}\mathtt{b}x_1\mathtt{b}x_2\mathtt{a}x_1x_2x_1$. The universe of $\mathcal{A}^s_\alpha$ is $A_\alpha = \{1, 2, \ldots, 10\}$ and the relations are interpreted in the following way. $S^{\mathcal{A}^s_\alpha} = \{(1,2), (2,3), \ldots, (9,10)\}$, $L^{\mathcal{A}^s_\alpha} = \{1\}$, $R^{\mathcal{A}^s_\alpha} = \{10\}$, $D_{\mathtt{a}}^{\mathcal{A}^s_\alpha} = \{2, 7\}$, $D_{\mathtt{b}}^{\mathcal{A}^s_\alpha} = \{3, 5\}$ and, finally, $E^{\mathcal{A}^s_\alpha} = \{(1,4), (4,8), (6,9), (8,10)\}$.

We now introduce our representation of words over the terminal alphabet Σ as τ_Σ-structures. We recall that it is our goal to represent the membership problem for pattern languages as homomorphism problem for relational structures. Hence, the way we represent terminal words by τ_Σ-structures must cater for this purpose. Furthermore, we have to distinguish between the E case and the NE case. We first introduce the NE case and shall afterwards point out how to extend the constructions for the E case. We choose the universe to be the set of all possible factors of w, where these factors are represented by their unique start and end positions in w; thus, two factors that are equal but occur at different positions in w are different elements of the universe. The interpretation of L contains all prefixes and the interpretation of R contains all suffixes of w. The interpretation of S, which for patterns contains pairs of adjacent variables, contains now pairs of adjacent (non-overlapping) factors of w. The relation E is interpreted such that it contains *all* pairs of factors that are equal and non-overlapping. Finally, for every $b \in \Sigma$, D_b contains all factors of length one that equal b. This is necessary for the possible terminal symbols in the pattern.

For the E case, the empty factors of w need to be represented as well. To this end, for every i, $0 \leq i \leq |w|$, we add an element i_ε to the universe denoting the empty factor between positions i and $i+1$ in w. The interpretations of S and R are extended to also contain the empty prefix and the empty suffix, respectively, and relation S is extended to relate non-empty factors to adjacent empty factors and, in addition, each empty factor is also related to itself by S. Next, we formally define this construction for the NE case and its extension to the E case.

Definition 9. *Let $w \in \Sigma^*$ be a terminal word. The* NE-w-structure $(\mathcal{A}_w)$ *with universe A_w is defined by*

- $A_w := \{(i,j) \mid 1 \leq i \leq j \leq |w|\}$,
- $E^{\mathcal{A}_w} := \{((i,j),(i',j')) \mid j < i' \text{ or } j' < i, w[i,j] = w[i',j']\}$,
- $S^{\mathcal{A}_w} := \{((i,j),(j+1,j')) \mid 1 \leq i \leq j, j+1 \leq j' \leq |w|\}$,
- $L^{\mathcal{A}_w} := \{(1,j) \mid 1 \leq j \leq |w|\}$,
- $R^{\mathcal{A}_w} := \{(i,|w|)| \mid 1 \leq i \leq |w|\}$ *and,*
- *for every $b \in \Sigma$, $D_b^{\mathcal{A}_w} := \{(i,i) \mid w[i] = b\}$.*

Let $\mathcal{A}_w$ be the NE-w-structure with universe A_w. We define the E-w-structure $(\mathcal{A}_w^\varepsilon)$ *with universe A_w^ε as follows:*

- $A_w^\varepsilon := A_w \cup \{i_\varepsilon \mid 0 \leq i \leq |w|\}$,
- $E^{\mathcal{A}_w^\varepsilon} := E^{\mathcal{A}_w} \cup \{(i_\varepsilon, j_\varepsilon) \mid 0 \leq i \leq |w|, 0 \leq j \leq |w|\}$,
- $S^{\mathcal{A}_w^\varepsilon} := S^{\mathcal{A}_w} \cup \{(i_\varepsilon, i_\varepsilon) \mid 0 \leq i \leq |w|\} \cup$
  $\{((i,j), j_\varepsilon)) \mid 1 \leq i \leq j \leq |w|\} \cup \{(i_\varepsilon, (i+1,j)) \mid 0 \leq i \leq j \leq |w|\}$,
- $L^{\mathcal{A}_w^\varepsilon} := L^{\mathcal{A}_w} \cup \{0_\varepsilon\}$,
- $R^{\mathcal{A}_w^\varepsilon} := R^{\mathcal{A}_w} \cup \{|w|_\varepsilon\}$ *and,*
- *for every $b \in \Sigma$, $D_b^{\mathcal{A}_w^\varepsilon} := D_b^{\mathcal{A}_w}$.*

We illustrate the above definition with a brief example. To this end, let $w := \mathtt{abab}$. According to Definition 9, the universe of the NE-w-structure $\mathcal{A}_w$ is the set of all factors of w, given by their start and end positions in w, i. e.,

$$A_w = \{(1,1),(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,3),(3,4),(4,4)\}\,.$$

For every i,j,k, $1 \leq i \leq j < k \leq 4$, the elements (i,j) and $(j+1,k)$ are in S relation under $\mathcal{A}_w$. Thus,

$$\begin{aligned} S^{\mathcal{A}_w} = \{&((1,1),(2,2)),((1,1),(2,3)),((1,1),(2,4)),((1,2),(3,3)),\\ &((1,2),(3,4)),((1,3),(4,4)),((2,2),(3,3)),((2,2),(3,4)),\\ &((2,3),(4,4)),((3,3),(4,4))\}\,. \end{aligned}$$

Every prefix of w is in L relation and every suffix of w is in R relation under $\mathcal{A}_w$. Hence, $L^{\mathcal{A}_w} = \{(1,1),(1,2),(1,3),(1,4)\}$ and $R^{\mathcal{A}_w} = \{(1,4),(2,4),(3,4),(4,4)\}$. Furthermore, $D_{\mathtt{a}}^{\mathcal{A}_w}$ and $D_{\mathtt{b}}^{\mathcal{A}_w}$ contain all factors that correspond to a single occurrence of $\mathtt{a}$ and $\mathtt{b}$, respectively, which implies $D_{\mathtt{a}}^{\mathcal{A}_w} = \{(1,1),(3,3)\}$, $D_{\mathtt{b}}^{\mathcal{A}_w} = \{(2,2),(4,4)\}$. Finally, two elements (i,j) and (i',j') are in E relation under $\mathcal{A}_w$ if and only if $w[i,j] = w[i',j']$; thus,

$$\begin{aligned} E^{\mathcal{A}_w} = \{&((1,1),(3,3)),((2,2),(4,4)),((1,2),(3,4)),\\ &((3,3),(1,1)),((4,4),(2,2)),((3,4),(1,2))\}\,. \end{aligned}$$

In the following lemma we state that the membership problem for pattern languages can be reduced to the homomorphism problem for relational structures. We shall informally explain this for the case of NE-pattern languages given by patterns that do not contain any terminal symbols. Let α be a pattern without terminal symbols and let w be a terminal word, let $\mathcal{A}_\alpha$ be an α-structure and let $\mathcal{A}_w$ be the NE-w-structure. If there exists a substitution h that maps α to w, then we can construct a homomorphism g from $\mathcal{A}_\alpha$ to $\mathcal{A}_w$ by mapping the positions of α to the factors of w according to the substitution h. If two positions in α are adjacent, then so are their images under h in w and the same holds for equal variables in α; hence, g is a valid homomorphism. If, on the other hand, there exists a homomorphism g from $\mathcal{A}_\alpha$ to $\mathcal{A}_w$, then the elements of the universe of $\mathcal{A}_\alpha$, i. e., positions of α, are mapped to factors of w such that a factorisation of w is described. This is enforced by the relations S, L and R. Furthermore, this mapping from α to w induced by g is a substitution, since the symmetric transitive closure of $E^{\mathcal{A}_\alpha}$ contains all pairs (i,j) with $\alpha[i] = \alpha[j]$ and $\alpha[i] \in X$. For general patterns with terminal symbols and for the E case the idea is the same, but the situation is technically more complex.

Lemma 10. *Let α be a pattern, $w \in \Sigma^*$ and let $\mathcal{A}_\alpha$ be an α-structure. Then $w \in L_{\mathrm{NE}}(\alpha)$ (or $w \in L_{\mathrm{E}}(\alpha)$) if and only if there exists a homomorphism from $\mathcal{A}_\alpha$ to $\mathcal{A}_w$ (or from $\mathcal{A}_\alpha$ to $\mathcal{A}_w^\varepsilon$, respectively).*

Proof. We only prove the E-case, i. e., $w \in L_{\mathrm{E}}(\alpha)$ if and only if there exists a homomorphism from $\mathcal{A}_\alpha$ to $\mathcal{A}_w^\varepsilon$. The proof for the NE-case is easier and can be done analogously. We start with the *if* direction. To this end, we assume that there exists a homomorphism $g : A_\alpha \to A_w^\varepsilon$ from $\mathcal{A}_\alpha$ to $\mathcal{A}_w^\varepsilon$, i. e., for every $p, q \in A_\alpha$,

- if $(p,q) \in E^{\mathcal{A}_\alpha}$, then $(g(p), g(q)) \in E^{\mathcal{A}_w^\varepsilon}$,
- if $(p,q) \in S^{\mathcal{A}_\alpha}$, then $(g(p), g(q)) \in S^{\mathcal{A}_w^\varepsilon}$,
- for every $b \in \Sigma$, if $p \in D_b^{\mathcal{A}_\alpha}$, then $g(p) \in D_b^{\mathcal{A}_w^\varepsilon}$
- $g(1) \in L^{\mathcal{A}_w^\varepsilon}$ and
- $g(|\alpha|) \in R^{\mathcal{A}_w^\varepsilon}$.

For the sake of convenience, we partition the universe of $\mathcal{A}_w^\varepsilon$ into $A_w^\varepsilon = A_w^{\neg\varepsilon} \cup A_w^\varepsilon$, where $A_w^{\neg\varepsilon} := \{(i,j) \mid 1 \leq i \leq j \leq |w|\}$ and $A_w^\varepsilon := \{i_\varepsilon \mid 0 \leq i \leq |w|\}$. For every $p \in A_\alpha$, if $g(p) = (s,t) \in A_w^{\neg\varepsilon}$, then we define $h(\alpha[p]) := w[s,t]$ and if, on the other hand, $g(p) \in A_w^\varepsilon$, then we define $h(\alpha[p]) := \varepsilon$. We can observe that if $\alpha[p]$ is a terminal $b \in \Sigma$, then $p \in D_b^{\mathcal{A}_\alpha}$. Thus, $g(p) \in D_b^{\mathcal{A}_w^\varepsilon}$, which implies $g(p) = (s,s)$ with $w[s] = b$ and, therefore, $h(b) = b$. For every $p, q \in A_\alpha$ with $\alpha[p] = \alpha[q]$ and $\alpha[p] \in X$, (p,q) is in the symmetric transitive closure of $E^{\mathcal{A}_\alpha}$. We note that, by definition, $E^{\mathcal{A}_w^\varepsilon}$ equals its symmetric transitive closure. Hence, we can conclude that $(g(p), g(q)) \in E^{\mathcal{A}_w^\varepsilon}$, which implies that $w[s,t] = w[s',t']$, where $(s,t) := g(p)$ and $(s',t') := g(q)$. Since $h(\alpha[p]) = w[s,t]$ and $h(\alpha[q]) = w[s',t']$,

we may conclude $h(\alpha[p]) = h(\alpha[q])$. Consequently, h is a valid substitution and it remains to show $h(\alpha) = w$.

For every $p \in A_\alpha$, $p < |\alpha|$, $(p, p+1) \in S^{\mathcal{A}_\alpha}$ and, thus, $(g(p), g(p+1)) \in S^{\mathcal{A}_w^\varepsilon}$. By definition of $S^{\mathcal{A}_w^\varepsilon}$, this implies that either

1. $g(p) = (s,t) \in A_w^{\neg\varepsilon}$ and $g(p+1) = (t+1, t') \in A_w^{\neg\varepsilon}$,
2. $g(p) = s_\varepsilon \in A_w^\varepsilon$ and $g(p+1) = (s+1, t') \in A_w^{\neg\varepsilon}$,
3. $g(p) = (s,t) \in A_w^{\neg\varepsilon}$ and $g(p+1) = t_\varepsilon \in A_w^\varepsilon$ or
4. $g(p) = s_\varepsilon \in A_w^\varepsilon$ and $g(p+1) = s_\varepsilon \in A_w^\varepsilon$.

By the definition of h above, we can conclude that, for every $p, q \in A_\alpha$, $p < q$, with $g(p) = (s,t) \in A_w^{\neg\varepsilon}$ and $g(q) = (s', t') \in A_w^{\neg\varepsilon}$, $h(\alpha[p,q]) = w[s,t']$. Now let $l, r \in A_\alpha$ such that $g(l), g(r) \in A_w^{\neg\varepsilon}$ and, for every i with $1 \leq i < l$ and $r < i \leq |w|$, $g(i) \in A_w^\varepsilon$. This particularly means that $g(l) = (1,t)$ and $g(r) = (s', |w|)$. Since $1 \in L^{\mathcal{A}_\alpha}$ and $|\alpha| \in R^{\mathcal{A}_\alpha}$, we can conclude that $g(i) = 0_\varepsilon$, $1 \leq i < l$, and $g(i) = |w|_\varepsilon$, $r < i \leq |\alpha|$. Consequently, $h(\alpha[1, l-1]) = \varepsilon$, $h(\alpha[l,r]) = w$ and $h(\alpha[r+1, |\alpha|]) = \varepsilon$; hence, $h(\alpha) = w$.

For the *only if* direction, we assume that there exists a substitution h with $h(\alpha) = w$. We define a mapping $g : A_\alpha \to A_w^\varepsilon$ in the following way. For every $p \in A_\alpha$, if $h(\alpha[p]) \neq \varepsilon$, then we define $g(p) := (|h(\alpha[1, p-1])| + 1, |h(\alpha[1,p])|) \in A_w^{\neg\varepsilon}$ and, if $h(\alpha[p]) = \varepsilon$, then we define $g(p) := |h(\alpha[1, p-1])|_\varepsilon \in A_w^\varepsilon$. It remains to show that g is a homomorphism from $\mathcal{A}_\alpha$ to $\mathcal{A}_w^\varepsilon$. For every $b \in \Sigma$, if $p \in D_b^{\mathcal{A}_\alpha}$, then $\alpha[p] = h(\alpha[p]) = b$; thus, $g(p) = (s,s)$, where $s := |h(\alpha[1,p])|$, and, since $h(\alpha) = w$, $w[s] = b$, which implies $g(p) \in D_b^{\mathcal{A}_w^\varepsilon}$. Obviously, either $g(1) = (1, |h(\alpha[1])|)$ or $g(1) = 0_\varepsilon$, and therefore $g(1) \in L^{\mathcal{A}_w^\varepsilon}$. Similarly, either $g(|\alpha|) = (|h(\alpha[1, |\alpha| - 1])| + 1, |h(\alpha[1, |\alpha|])|)$ or $g(|\alpha|) = |\alpha|_\varepsilon$, which implies $g(|\alpha|) \in R^{\mathcal{A}_w^\varepsilon}$. For every $p, q \in A_\alpha$, if $(p,q) \in E^{\mathcal{A}_\alpha}$, then $\alpha[p] = \alpha[q]$ and, since $h(\alpha[p]) = h(\alpha[q])$, either $g(p) = (s,t)$ and $g(q) = (s', t')$ with $w[s,t] = w[s', t']$ or $g(p) = s_\varepsilon$ and $g(q) = s'_\varepsilon$. In both cases we can conclude that $(g(p), g(q)) \in E^{\mathcal{A}_w^\varepsilon}$. Let $p \in A_\alpha$, $p < |\alpha|$. We recall that $(p, p+1) \in S^{\mathcal{A}_\alpha}$ and observe four possible cases:

- If $g(p), g(p+1) \in A_w^{\neg\varepsilon}$, then $g(p) = (s,t)$ and $g(p+1) = (t+1, t')$.
- If $g(p) \in A_w^\varepsilon$ and $g(p+1) \in A_w^{\neg\varepsilon}$, then $g(p) = s_\varepsilon$ and $g(p+1) = (s+1, t')$.
- If $g(p) \in A_w^{\neg\varepsilon}$ and $g(p+1) \in A_w^\varepsilon$, then $g(p) = (s,t)$ and $g(p+1) = t_\varepsilon$.
- If $g(p), g(p+1) \in A_w^{\neg\varepsilon}$, then $g(p) = s_\varepsilon$ and $g(p+1) = s_\varepsilon$.

For all of these cases, $(g(p), g(p+1)) \in S^{\mathcal{A}_w^\varepsilon}$ is implied. This shows that g is an homomorphism from $\mathcal{A}_\alpha$ to $\mathcal{A}_w^\varepsilon$, which concludes the proof of the lemma. □

The above lemma shows that the membership problem for pattern languages is reducible to the homomorphism problem for relational structures and, thus, it can be solved by first transforming the pattern and the word into an α-structure

and the NE-w-structure or E-w-structure and then deciding the homomorphism problem for these structures.

In the following, we say that a set of patterns P has bounded treewidth if and only if there exists a polynomial time computable mapping g that maps every $\alpha \in P$ to an α-structure, such that $\{g(\alpha) \mid \alpha \in P\}$ has bounded treewidth. From Theorem 6 and Lemma 10 we can conclude the following result.

Corollary 11. *Let $P \subseteq (X \cup \Sigma)^+$ be a set of patterns with bounded treewidth. Then* NE-PATMem(P) *and* E-PATMem(P) *are decidable in polynomial time.*

Proof. We assume that P has a bounded treewidth of $k \in \mathbb{N}$. Let $\alpha \in P$ and let $w \in \Sigma^*$. Obviously, w can be converted into the E-w-structure $\mathcal{A}_w^\varepsilon$ or into the NE-w-structure $\mathcal{A}_w$ in time $\mathrm{O}(|w|^4)$. Furthermore, by assumption, an α-structure $\mathcal{A}_\alpha$ that satisfies $\mathrm{tw}(\mathcal{A}_\alpha) \leq k$ can be computed in polynomial time. From Theorem 6, it follows that we can check whether or not there exists a homomorphism from $\mathcal{A}_\alpha$ to $\mathcal{A}_w^\varepsilon$ (or from $\mathcal{A}_\alpha$ to $\mathcal{A}_w$, respectively) in polynomial time. Now with Lemma 10, we can conclude the statement of the corollary. □

Due to Corollary 11, the task of identifying classes of patterns for which the membership problem is decidable in polynomial time can now be seen from a different angle, i. e., as the problem of finding classes of patterns with bounded treewidth. The fact that we can easily rephrase known results about the complexity of the membership problem for pattern languages in terms of standard α-structures with a bounded treewidth, pointed out by the following proposition, indicates that this point of view is natural and fits with our current knowledge of the membership problem for pattern languages.

Proposition 12. *For every $k \in \mathbb{N}$, the sets of patterns $\{\alpha \mid \alpha$ is regular$\}$, $\{\alpha \mid \alpha$ is non-cross$\}$, $\{\alpha \mid |\mathrm{var}(\alpha)| \leq k\}$ and $\{\alpha \mid \mathrm{vd}(\alpha) \leq k\}$ have all bounded treewidth.*

Proof. If α is regular, then $\mathcal{A}_\alpha^s$ is a path and, thus, $\mathrm{tw}(\mathcal{A}_\alpha^s) = 1$. If α is non-cross, then it is straightforward to construct a path decomposition of $\mathcal{A}_\alpha^s$ with a width of at most 2. We can note that, By Lemma 3, $\mathrm{scd}(\alpha) \leq \mathrm{vd}(\alpha) + 1$ and, obviously, $\mathrm{vd}(\alpha) + 1 \leq |\mathrm{var}(\alpha)|$. In Section 4, Lemma 15, it shall be shown that $\mathrm{tw}(\mathcal{A}_\alpha^s) \leq \mathrm{scd}(\alpha) + 1$, which implies that $\{\alpha \mid |\mathrm{var}(\alpha)| \leq k\}$ and $\{\alpha \mid \mathrm{vd}(\alpha) \leq k\}$ have bounded treewidth. □

We conclude that our encoding of the membership problem for pattern languages in terms of the homomorphism problem for relational structure is natural and the hardness of the membership problem seems to be covered by the treewidth of the α-structures.

In the next section, we show that the numerical parameter of the scope coincidence degree of a pattern α is an upper bound for the treewidth of the standard α-structure; thus, restricting it yields classes of patterns with a polynomial time solvable membership problem. Moreover, in Section 5, we identify a large class of patterns with a bounded treewidth of 2, but an unbounded scope coincidence degree.

4 Patterns with Restricted Scope Coincidence Degree

In order to show that, for every $k \in \mathbb{N}$, the set $\{\alpha \mid \mathrm{scd}(\alpha) \leq k\}$ has bounded treewidth we define, for any pattern α, a path decomposition of its standard α-structure.

Definition 13. *Let α be a pattern and let $V := \{v_1, v_2, \ldots, v_{|\alpha|}\}$ be the set of vertices of the Gaifman graph of its standard α-structure, where, for every i, $1 \leq i \leq |\alpha|$, v_i corresponds to $\alpha[i]$. We inductively construct a sequence P_α of subsets of V in the following way.*

1. *Add $\{v_1\}$ to P_α, add $\{v_1, v_2\}$ to P_α, define $B := \{v_1, v_2\}$ and $i := 3$.*
2. *Define $B := B \cup \{v_i\}$ and, if $\alpha[i-2]$ is a terminal symbol or the rightmost occurrence of a variable in α, then define $B := B \setminus \{v_{i-2}\}$.*
3. *Add B to P_α.*
4. *If $\alpha[i] = x \in X$, but $\alpha[i]$ is not the leftmost occurrence of x, then define $B := B \setminus \{v_j\}$, where $j < i$, $\alpha[j] = x$ and, for every j', $j < j' < i$, $\alpha[j'] \neq x$.*
5. *Define $i := i + 1$ and if $i \leq |\alpha|$, then go to step 2.*

Intuitively, the sequence $P_\alpha := (B_1, B_2, \ldots, B_k)$ is constructed in the following way. The first two sets are $\{v_1\}$ and $\{v_1, v_2\}$, respectively, and every following set is obtained from the previous one by adding the next vertex v_i and removing v_{i-2} if it corresponds to a terminal or the rightmost occurrence of a variable. Furthermore, if v_i corresponds to a variable that is not the leftmost occurrence of that variable, then the previous occurrence of this variable is still in our set and can now be removed. This ensures that for every edge $\{v_i, v_j\}$ of the Gaifman graph of the standard α-structure, there exists an l, $1 \leq l \leq k$, such that $\{v_i, v_j\} \subseteq B_l$. Furthermore, it can be easily verified that, for every vertex v of the Gaifman graph of the standard α-structure, there exist i, j, $1 \leq i < j \leq k$, such that $v \in \bigcap_{l=i}^{j} B_l$ and $v \notin ((\bigcup_{l=1}^{i-1} B_l) \cup (\bigcup_{l=j+1}^{k} B_l))$. Since, for every i, $1 \leq i \leq k$, exactly one element B_i is added to P_α in the construction of Definition 13, we can conclude that $k = |\alpha|$. We can further note that, for every i, $2 \leq i \leq k$, B_i contains exactly one new vertex that is not already contained in B_{i-1}, i. e., $|B_i \setminus B_{i-1}| = 1$. Next, we shall illustrate Definition 13 by a short example. Let $\beta := x_1 \mathtt{a} x_2 x_1 \mathtt{ab} x_2 x_3 x_3$. Then $P_\alpha = (\{v_1\}, \{v_1, v_2\}, \{v_1, v_2, v_3\}, \{v_1, v_3, v_4\}, \{v_3, v_4, v_5\}, \{v_3, v_5, v_6\}, \{v_3, v_6, v_7\}, \{v_7, v_8\}, \{v_8, v_9\})$.

The above considerations imply the following:

Proposition 14. *Let α be a pattern. Then $P_\alpha := (B_1, B_2, \ldots, B_k)$ is a path decomposition of the Gaifman graph of its standard α-structure. Moreover, $k = |\alpha|$ and, for every i, $2 \leq i \leq |\alpha|$, $|B_i \setminus B_{i-1}| = 1$.*

We call P_α the *standard path decomposition of* α and we shall now show that the width of the standard path decomposition is bounded by the scope coincidence degree of the corresponding pattern.

Lemma 15. *Let α be a pattern. Then the standard path decomposition of α has width at most* $\mathrm{scd}(\alpha) + 1$.

Proof. Let $P_\alpha := (B_1, B_2, \ldots, B_{|\alpha|})$ be the standard path decomposition of α. We assume to the contrary that P_α has a width of at least $\mathrm{scd}(\alpha) + 2$, which implies that there exists a q, $1 \leq q \leq |\alpha|$, such that $|B_q| = m \geq \mathrm{scd}(\alpha) + 3$. Let $B_q := \{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$, where the vertices of B_q are in ascending order with respect to their indices. By definition of the standard path decomposition of α, for every j, $1 \leq j \leq m-2$, v_{i_j} corresponds to an occurrence of a distinct variable y_j in α. Furthermore, for every j, $1 \leq j \leq m-2$, there must exist an occurrence of y_j to the left and to the right of position q in α. This is due to the fact that if there is no occurrence of y_j to the left of q, then no vertex that corresponds to an occurrence of variable y_j is contained in B_q, and if there is no occurrence of y_j to the right of q, then vertex v_{i_j} would have been removed in step 2 of the procedure described in Definition 13. This directly implies that the scopes of variables $y_1, y_2, \ldots, y_{m-2}$ coincide and, since $m \geq \mathrm{scd}(\alpha)+3$, there are at least $\mathrm{scd}(\alpha) + 1$ variables in α, the scopes of which coincide, which is a contradiction. □

By the previous lemma, we can conclude that, for every pattern α, the treewidth of the standard α-structure is bounded by the scope coincidence degree of α. Hence, for every $k \in \mathbb{N}$, the class of patterns $\{\alpha \mid \mathrm{scd}(\alpha) \leq k\}$ has bounded treewidth, and with Corollary 11 we can conclude that E-PATMem($\{\alpha \mid \mathrm{scd}(\alpha) \leq k\}$) and NE-PATMem($\{\alpha \mid \mathrm{scd}(\alpha) \leq k\}$) are solvable in polynomial time. However, we are interested in more detailed analyses of the time complexity of these problems. To this end, we give an algorithm that solves the homomorphism problem for the standard α-structure and a w-structure by using the standard path decomposition of α and analyse its time complexity. This algorithm follows the obvious way of using tree decompositions, which has already been briefly outlined at the end of Section 2. Hence, the main effort is to determine its runtime.

Theorem 16. *Let $k \in \mathbb{N}$ and $Z \in \{\mathrm{E}, \mathrm{NE}\}$. The problem* Z-PATMem$(\{\alpha \mid \mathrm{scd}(\alpha) \leq k\})$ *is solvable in time* $\mathrm{O}(|\alpha| \times |w|^{2(k+3)} \times (k+2)^2)$.

Proof. We only show that NE-PATMem($\{\alpha \mid \mathrm{scd}(\alpha) \leq k\}$) is solvable in time $\mathrm{O}(|\alpha| \times |w|^{2(k+3)} \times (k+2)^2)$, since the E case can be dealt with analogously. Let (α, w) be an instance of NE-PATMem($\{\alpha \mid \mathrm{scd}(\alpha) \leq k\}$). We decide on whether or not $w \in L_{\mathrm{NE}}(\alpha)$ by reduction to the homomorphism problem for relational structures. To this end, we first need to construct $\mathcal{A}_\alpha^s$ and $\mathcal{A}_w$, which can be done in time $\mathrm{O}(|w|^4 + |\alpha|)$. Let A_α and A_w be the universes of $\mathcal{A}_\alpha^s$ and $\mathcal{A}_w$, respectively, and let $P_\alpha := (B_1, B_2, \ldots, B_{|\alpha|})$ be the standard path decomposition of α. Before we give an algorithm deciding on whether or not there exists a homomorphism from $\mathcal{A}_\alpha^s$ to $\mathcal{A}_w$, we introduce some helpful notations.

Let h be a partial mapping from A_α to A_w. We say that h satisfies condition $(*)$ if and only if, for every $R \in \tau_\Sigma$ and for all $a_1, a_2, \ldots, a_{\mathrm{ar}(R)} \in A_\alpha$ for which h is defined, $(a_1, a_2, \ldots, a_{\mathrm{ar}(R)}) \in R^{\mathcal{A}^s_\alpha}$ implies $(h(a_1), h(a_2), \ldots, h(a_{\mathrm{ar}(R)})) \in R^{\mathcal{A}_w}$. Let $A := (a_1, a_2, \ldots, a_k)$ and $B := (b_1, b_2, \ldots, b_k)$ be arbitrary tuples of equal length. Then $A \mapsto B$ denotes the mapping that, for every $1 \leq i \leq k$, maps a_i to b_i. For any $C \subseteq A_\alpha$, $\mathrm{ord}(C)$ is a tuple containing the elements from C in increasing order (recall that $A_\alpha = \{1, 2, \ldots, |\alpha|\}$). Two partial mappings g and h from A_α to A_w are called *compatible* if and only if, for every $a \in A_\alpha$ for which both h and g are defined, $g(a) = h(a)$ is satisfied.

In the following, we shall describe an algorithm that decides on whether or not there exists a homomorphism from $\mathcal{A}^s_\alpha$ to $\mathcal{A}_w$. First, we compute a set H_1 of all tuples C of size $|B_1|$ containing elements from A_w such that the mapping $\mathrm{ord}(B_1) \mapsto C$ satisfies condition $(*)$. After that, for every i, $2 \leq i \leq |\alpha|$, we inductively compute a set H_i that is defined in the following way. For every tuple C of size $|B_i|$ containing elements from A_w, if the mapping $\mathrm{ord}(B_i) \mapsto C$ satisfies condition $(*)$ and the set H_{i-1} contains a tuple C' such that the mappings $\mathrm{ord}(B_i) \mapsto C$ and $B_{i-1} \mapsto C'$ are compatible, then we add C to H_i.

We now claim that there exists a homomorphism from $\mathcal{A}^s_\alpha$ to $\mathcal{A}_w$ if and only if $H_{|\alpha|}$ is nonempty. In order to prove this claim, we first assume that there exists a homomorphism from $\mathcal{A}^s_\alpha$ to $\mathcal{A}_w$. Now, for every i, $1 \leq i \leq |\alpha|$, let C_i be the tuple of elements from A_w, such that the mappings $\mathrm{ord}(B_i) \mapsto C_i$, $1 \leq i \leq |\alpha|$, if combined, form h. We note that this particularly implies that each two of the mappings $\mathrm{ord}(B_i) \mapsto C_i$, $1 \leq i \leq |\alpha|$, are compatible. Since h is a homomorphism from $\mathcal{A}^s_\alpha$ to $\mathcal{A}_w$, for every i, $1 \leq i \leq |\alpha|$, the mapping $\mathrm{ord}(B_i) \mapsto C_i$ satisfies condition $(*)$. This implies that $C_1 \in H_1$ holds and if, for some i, $1 \leq i \leq |\alpha| - 1$, $C_i \in H_i$ is satisfied, then, since the mappings $\mathrm{ord}(B_i) \mapsto C_i$ and $\mathrm{ord}(B_{i+1}) \mapsto C_{i+1}$ are compatible, $C_{i+1} \in H_{i+1}$ follows. By induction, this implies that $H_{|\alpha|}$ contains $C_{|\alpha|}$ and, thus, is nonempty.

Next, we assume that $H_{|\alpha|}$ is nonempty; thus, it contains some $C_{|\alpha|}$. By definition, this directly implies that, for every i, $1 \leq i \leq |\alpha| - 1$, H_i contains some element C_i and, without loss of generality, we can also conclude that, for every i, $1 \leq i \leq |\alpha| - 1$, the mappings $\mathrm{ord}(B_i) \mapsto C_i$ and $\mathrm{ord}(B_{i+1}) \mapsto C_{i+1}$ are compatible. Furthermore, since, for every $a \in A_\alpha$, there must exist at least one i, $1 \leq i \leq |\alpha|$, with $a \in B_i$ and, for all j, j', $1 \leq j < j' \leq |\alpha|$, $a \in (B_j \cap B_{j'})$ implies $a \in B_{j''}$, $j \leq j'' \leq j'$, we can conclude that each two of the mappings $\mathrm{ord}(B_i) \mapsto C_i$, $1 \leq i \leq |\alpha|$, are compatible and for every $a \in A_\alpha$ at least one of the mappings $\mathrm{ord}(B_i) \mapsto C_i$, $1 \leq i \leq |\alpha|$, is defined. This particularly implies that we can construct a total mapping h from A_α to A_w by combining all the mappings $\mathrm{ord}(B_i) \mapsto C_i$, $1 \leq i \leq |\alpha|$. Now let $a_1, a_2, \ldots, a_{\mathrm{ar}(R)}$ be arbitrary elements from A_α such that, for some $R \in \tau_\Sigma$, $(a_1, a_2, \ldots, a_{\mathrm{ar}(R)}) \in R^{\mathcal{A}^s_\alpha}$. Since there must exist an i, $1 \leq i \leq |\alpha|$, with $a_1, a_2, \ldots, a_{\mathrm{ar}(R)} \in B_i$ and since $C_i \in H_i$, i. e., $\mathrm{ord}(B_i) \mapsto C_i$ satisfies condition $(*)$, we can conclude that $(h(a_1), h(a_2), \ldots, h(a_{\mathrm{ar}(R)})) \in R^{\mathcal{A}_w}$, which implies that h is a homomorphism from $\mathcal{A}^s_\alpha$ to $\mathcal{A}_w$.

It remains to determine the runtime of the above algorithm. A central

element of that algorithm is to check whether or not, for some i, $1 \leq i \leq |\alpha|$, and some tuple C of size $|B_i|$ containing elements from A_w, the mapping $\mathrm{ord}(B_i) \mapsto C$ satisfies condition $(*)$. Since the arity of any relation symbol in τ_Σ is at most 2, this can be done in time $\mathrm{O}(|B_i|^2)$. The set H_1 can be computed by simply considering every tuple C of elements from A_w of size $|B_1|$ and checking whether $\mathrm{ord}(B_1) \mapsto C$ satisfies condition $(*)$. Thus, time $\mathrm{O}(|B_1|^2 \times |A_w|^{|B_1|})$ is sufficient for computing H_1 and it remains to compute H_i, for every i, $2 \leq i \leq |\alpha|$. We recall that in order to compute such an H_i, we need to collect all tuples C of size $|B_i|$ containing elements from A_w such that the mapping $\mathrm{ord}(B_i) \mapsto C$ satisfies condition $(*)$ and the set H_{i-1} contains a tuple C' such that the mappings $\mathrm{ord}(B_i) \mapsto C$ and $\mathrm{ord}(B_{i-1}) \mapsto C'$ are compatible. However, this can be done without having to enumerate all possible tuples C of size $|B_i|$ and then check for each such tuple whether or not H_{i-1} contains a tuple C' such that the mappings $\mathrm{ord}(B_i) \mapsto C$ and $\mathrm{ord}(B_{i-1}) \mapsto C'$ are compatible. This is due to the fact that, by Proposition 14, $|B_i \setminus B_{i-1}| = 1$, thus, all elements but one of the tuple C are already determined by the condition that there needs to be a $C' \in H_{i-1}$ such that the mappings $\mathrm{ord}(B_i) \mapsto C$ and $\mathrm{ord}(B_{i-1}) \mapsto C'$ are compatible. Consequently, there are at most $|A_w| \times |H_{i-1}|$ tuples that need to be checked for whether or not they satisfy condition $(*)$. We conclude that the set H_i can be computed in time $\mathrm{O}(|A_w| \times |A_w|^{|B_{i-1}|} \times |B_i|^2) = \mathrm{O}(|A_w|^{|B_{i-1}|+1} \times |B_i|^2)$. Since, by Lemma 15, the width of the standard path decomposition is at most $k+1$, which implies $|B_i| \leq k+2$, for every i, $1 \leq i \leq |\alpha|$, we can conclude that the total runtime of the algorithm is $\mathrm{O}(|\alpha| \times |A_w|^{k+3} \times (k+2)^2) = \mathrm{O}(|\alpha| \times |w|^{2(k+3)} \times (k+2)^2)$. $\square$

The above result is similar to, but much stronger than the result that every class of patterns with a bounded variable distance has a polynomial time membership problem (see [19]). This is due to the fact that if the variable distance is bounded by a constant, then this constitutes a much stronger restriction on the structure of a pattern than if the scope coincidence degree is restricted. Intuitively, this can be illustrated by the following situation. For an arbitrary pattern $\alpha := \alpha_1 \cdot \alpha_2$, we insert a pattern β with $\mathrm{var}(\alpha) \cap \mathrm{var}(\beta) = \emptyset$ into α, i. e., $\alpha' := \alpha_1 \cdot \beta \cdot \alpha_2$. Now, if $\mathrm{var}(\alpha_1) \cap \mathrm{var}(\alpha_2) \neq \emptyset$, then the variable distance of α' increases at least by $|\mathrm{var}(\beta)| - \mathrm{vd}(\alpha)$ compared to α regardless of the structure of β. This implies that it is rather difficult to enlarge a pattern by inserting new variables without increasing its variable distance. On the other hand, the scope coincidence degree of α' increases at least by $\mathrm{scd}(\beta) - \mathrm{scd}(\alpha)$ compared to α. This implies that the scope coincidence degree of α' depends on the structure of β or, more precisely, on the scope coincidence degree of β.

In the next section, we shall identify another structural property of patterns that allows the membership problem to be solved in polynomial time and that is incomparable to the variable distance and the scope coincidence degree.

5 Mildly Entwined Patterns

Let α be a pattern. We say that two variables $x, y \in \mathrm{var}(\alpha)$ are *entwined* if and only if there exists a factorisation $\alpha = \beta \cdot x \cdot \gamma_1 \cdot y \cdot \gamma_2 \cdot x \cdot \gamma_3 \cdot y \cdot \delta$ or $\alpha = \beta \cdot y \cdot \gamma_1 \cdot x \cdot \gamma_2 \cdot y \cdot \gamma_3 \cdot x \cdot \delta$, where $\beta, \gamma_1, \gamma_2, \gamma_3, \delta \in (X \cup \Sigma)^*$. If no two variables in α are entwined, then α is a *nested* pattern. Intuitively, in a nested pattern, if a variable x occurs between two occurrences of another variable y, then *all* occurrences of x occur between these two occurrences of y. For example, $x_1x_3x_3x_4x_4x_1x_5x_5x_1x_2x_6x_7x_7x_6x_2$ is a nested pattern.

Next, we define a class of patterns that comprises entwined variables, but in a very restricted form.

Definition 17. *A pattern α is* closely entwined *if and only if, for all $x, y \in \mathrm{var}(\alpha)$, the existence of a factorisation $\alpha = \beta \cdot x \cdot \gamma_1 \cdot y \cdot \gamma_2 \cdot x \cdot \gamma_3 \cdot y \cdot \delta$ with $\beta, \gamma_1, \gamma_2, \gamma_3, \delta \in (X \cup \Sigma)^*$ and $|\gamma_2|_x = |\gamma_2|_y = 0$ implies $\gamma_2 = \varepsilon$.*

In a closely entwined pattern, we allow variables to be entwined, but in the closest possible way, i. e., we require γ_2 to be empty. The following is an example for a closely entwined pattern: $\beta := x_1x_4x_1x_4x_5x_5x_4x_2x_1x_3x_2x_3x_2$. In β the variables x_1 and x_4, the variables x_1 and x_2 and the variables x_2 and x_3 are all pairs of variables that are entwined and, furthermore, they all satisfy the condition of Definition 17. Obviously, the set of nested patterns is a proper subset of the class of closely entwined patterns. Next, we define a class of patterns that properly lies between the classes of nested patterns and closely entwined patterns.

Definition 18. *A pattern α is* mildly entwined *if and only if it is closely entwined and, for every $x \in \mathrm{var}(\alpha)$, if $\alpha = \beta \cdot x \cdot \gamma \cdot x \cdot \delta$ with $\beta, \gamma, \delta \in (X \cup \Sigma)^*$ and $|\gamma|_x = 0$, then γ is nested.*

Intuitively, a mildly entwined pattern is by definition a closely entwined pattern with the additional condition that every factor that lies in between two consecutive occurrences of a variable is a nested pattern. Obviously, there exist closely entwined patterns that are not mildly entwined (e. g., $x_1x_2x_3x_2x_3x_1$) and mildly entwined patterns that are not nested (e. g., $x_1x_2x_1x_2$). The following constitutes a more involved example for a mildly entwined pattern: $\beta := x_1x_3x_4x_4x_3x_3x_1x_2x_3x_5x_5x_2x_5x_6x_6x_2$. First, we can note that the variables x_1 and x_3 are closely entwined, x_2 and x_3 are closely entwined, x_2 and x_5 are closely entwined and these are the only pairs of variables that are entwined. Furthermore, every factor between two consecutive occurrences of the same variable is nested. We emphasise that a factor γ between two consecutive occurrences of the same variable can still contain occurrences of a variable that is entwined with other variables in β, as long as γ, considered individually, is nested. For example, the factor $x_3x_4x_4x_3x_3$ in between the first two occurrences of x_1 in β contains variable x_3, which is entwined with variables x_1 and x_2.

Since we can decide in polynomial time on whether or not a given pattern is nested or closely entwined, we can also decide on whether or not a given pattern is mildly entwined in polynomial time.

Next, we shall show that the membership problem for the class of mildly entwined patterns can be decided in polynomial time. To this end, we need to introduce a special class of graphs:

Definition 19. *A graph is called* outerplanar *if and only if it can be drawn on the plane in such a way that no two edges cross each other and no vertex is entirely surrounded by edges (or, equivalently, all vertices lie on the exterior face).*

For example, a cycle with 4 vertices is outerplanar, but the complete graph with 4 vertices, although planar, is not outerplanar.

Next, we show that a pattern is mildly entwined if and only if its standard α-structure is outerplanar.

Lemma 20. *Let α be a pattern. The Gaifman graph of the standard α-structure is outerplanar if and only if α is mildly entwined.*

Proof. Let $\mathcal{G}$ be the Gaifman graph of the standard α-structure and let $V := \{v_1, v_2, \ldots, v_{|\alpha|}\}$ be its set of vertices, where, for every i, $1 \leq i \leq |\alpha|$, v_i corresponds to $\alpha[i]$. We first show the *only if* direction by contraposition. To this end, we assume that α is not mildly entwined, which implies that α is either not closely entwined or there exists an $x \in \mathrm{var}(\alpha)$ such that $\alpha = \beta \cdot x \cdot \gamma \cdot x \cdot \delta$ with $|\gamma|_x = 0$ and γ is not nested. If α is not closely entwined, then there are $x, y \in \mathrm{var}(\alpha)$ such that $\alpha = \beta \cdot x \cdot \gamma_1 \cdot y \cdot \gamma_2 \cdot x \cdot \gamma_3 \cdot y \cdot \delta$ with $|\gamma_2|_x = |\gamma_2|_y = 0$ and $\gamma_2 \neq \varepsilon$. Furthermore, without loss of generality, we can assume that $|\gamma_1|_x = |\gamma_3|_y = 0$. Now let p_x, q_x, p_y, q_y be the positions of the occurrences of x and y shown by the above factorisation of α, i. e., $p_x = |\beta|+1$, $p_y = p_x+|\gamma_1|+1$, $q_x = p_y + |\gamma_2| + 1$ and $q_y = q_x + |\gamma_3| + 1$. We note that, since $|\gamma_1 \cdot \gamma_2|_x = 0$ and $|\gamma_2 \cdot \gamma_3|_y = 0$, there are edges $\{v_{p_x}, v_{q_x}\}$ and $\{v_{p_y}, v_{q_y}\}$ in $\mathcal{G}$ and, furthermore, there exists paths $(v_{p_x}, v_{p_x+1}, \ldots, v_{p_y})$ and $(v_{q_x}, v_{q_x+1}, \ldots, v_{q_y})$. This directly implies that, for every i, $p_y < i < q_x$, the vertex v_i is necessarily entirely surrounded by edges. Since $\alpha[p_y + 1, q_x - 1] = \gamma_2 \neq \varepsilon$, there exists at least one such vertex and, thus, $\mathcal{G}$ is not outerplanar.

If, on the other hand, there exists an $x \in \mathrm{var}(\alpha)$ such that $\alpha = \beta \cdot x \cdot \gamma \cdot x \cdot \delta$ with $|\gamma|_x = 0$ and γ is not nested, then we can conclude that, for some $y, z \in \mathrm{var}(\alpha)$, $\alpha = \beta \cdot x \cdot \gamma_1 \cdot y \cdot \gamma_2 \cdot z \cdot \gamma_3 \cdot y \cdot \gamma_4 \cdot z \cdot \gamma_5 \cdot x \cdot \delta$ and, without loss of generality, $|\gamma_2 \cdot \gamma_3|_y = |\gamma_3 \cdot \gamma_4|_z = 0$. Now let p_x, q_x, p_y, q_y, p_z and q_z be the positions of the occurrences of variables x, y and z, respectively, as highlighted by the above factorisation. We note that in $\mathcal{G}$ there are edges $\{v_{p_x}, v_{q_x}\}$, $\{v_{p_y}, v_{q_y}\}$ and $\{v_{p_z}, v_{q_z}\}$ and, in a similar way as above, this implies that vertex v_{p_z} or v_{q_y} is necessarily entirely surrounded by edges.

It remains to show that if α is mildly entwined, then $\mathcal{G}$ is outerplanar. To this end, we assume that α is mildly entwined and show how to draw a diagram of $\mathcal{G}$ on the plane that satisfies the following condition referred to as $(*)$: no two edges cross each other and no vertex is entirely surrounded by edges. First, we draw the path $(v_1, v_2, \ldots, v_{|\alpha|})$ in a straight line and note that the diagram of this path satisfies condition $(*)$. We shall now step by step add the remaining

edges, which we call E-edges, since they are induced by the relation symbol E, and then show that in every step condition $(*)$ is maintained. In the following procedure, each of the E-edges will be drawn either above or below the path and we call a vertex v_i *covered above* or *covered below* (by an edge) if and only if we have already drawn an E-edge $\{v_j, v_{j'}\}$ with $j < i < j'$ above (or below, respectively) the path. We note that a vertex in the diagram is entirely surrounded by edges if and only if it is covered below and above at the same time. Next, we pass through the path from left to right, vertex by vertex. If for the current vertex v_p there does not exist an E-edge $\{v_p, v_q\}$ with $p < q$ (e. g., if v_p corresponds to a terminal symbol or to the rightmost occurrence of a variable), then we simply ignore this vertex and move on to the next one. If, one the other hand, such an E-edge exists, then we carry out one of the following steps.

1. If v_p is not covered above or below, then we draw the edge $\{v_p, v_q\}$ above the path.

2. If v_p is covered above or below by some edge and v_q is covered by the same edge, then we draw $\{v_p, v_q\}$ above the path (or below the path, respectively).

3. If v_p is covered above or below by some edge and v_q is not covered by this edge, then we draw $\{v_p, v_q\}$ below the path (or above the path, respectively).

It remains to show that each of the three steps above maintain condition $(*)$. If step 1 applies, then, since v_p is not covered by an edge, the subgraph with vertices $v_p, v_{p+1}, \ldots, v_{|\alpha|}$ is still a path and, thus, drawing $\{v_p, v_q\}$ above that path does not violate condition $(*)$. Now let us assume that step 2 applies and v_p is covered above by some edge $\{v_{p'}, v_{q'}\}$ with $p' < p < q < q'$. This implies that none of the vertices v_i, $p' < i < q'$, can be covered below by some edge, as otherwise they would be entirely surrounded by edges. So we can draw the edge $\{v_p, v_q\}$ above the path and still no vertex is entirely surrounded by edges. However, we have to show that we do not cross another edge by drawing $\{v_p, v_q\}$ in this way. To this end, we assume that there exists another edge $\{v_{\widehat{p}}, v_{\widehat{q}}\}$ that has already be drawn and that now crosses $\{v_p, v_q\}$ and we shall show that this assumption contradicts with the fact that α is mildly entwined. First, we can note that $\{v_{\widehat{p}}, v_{\widehat{q}}\}$ must be an E-edge that has been drawn above with either $p < \widehat{p} < q < \widehat{q}$ or $\widehat{p} < p < \widehat{q} < q$. We shall only consider the first of these two cases, since the second one can be handled analogously. Now, if $\widehat{q} < q'$, then $\alpha[p'+1, q'-1]$ is not nested, but, for some $x \in \mathrm{var}(\alpha)$, $\alpha[p'] = \alpha[q'] = x$ and $|\alpha[p'+1, q'-1]|_x = 0$. This is a contradiction to the fact that α is mildly entwined. If, on the other hand, $q' < \widehat{q}$, then we can observe the following. Let $\alpha = \beta{\cdot}x{\cdot}\gamma_1{\cdot}y{\cdot}\gamma_2{\cdot}x{\cdot}\gamma_3{\cdot}y{\cdot}\delta$ with $p' = |\beta|{+}1$, $\widehat{p} = |\beta{\cdot}x{\cdot}\gamma_1|{+}1$, $q' = |\beta{\cdot}x{\cdot}\gamma_1{\cdot}y{\cdot}\gamma_2|{+}1$ and $\widehat{q} = |\beta \cdot x \cdot \gamma_1 \cdot y \cdot \gamma_2 \cdot x \cdot \gamma_3| + 1$. Since $\{v_{p'}, v_{q'}\}$ and $\{v_{\widehat{p}}, v_{\widehat{q}}\}$ are E-edges, we can conclude that $|\gamma_2|_x = |\gamma_2|_y = 0$, but, since $\widehat{p} < q < q'$, $\gamma_2 \neq \varepsilon$. This is a contradiction to the fact that α is closely entwined. Therefore, we can

conclude that in fact $\{v_p, v_q\}$ does not cross an already existing edge and, thus, the diagram still satisfies condition $(*)$. If in step 2 vertex v_p is covered below instead of above, then an analogous argumentation can be used.

Finally, we assume that step 3 applies and v_p is covered above by some edge $\{v_{p'}, v_{q'}\}$ with $p' < p < q' < q$. We recall that $\alpha[p'] = \alpha[q']$ and $\alpha[p] = \alpha[q]$ and, since α is closely entwined, this implies that $p + 1 = q'$. Now we assume that no edge other than $\{v_{p'}, v_{q'}\}$ covers v_p, which particularly means that $v_{q'}$ is not covered by any edge. We conclude that we can draw the edge $\{v_p, v_q\}$ below the path without crossing an existing edge and since $p + 1 = q'$, i. e., there are no vertices between v_p and $v_{q'}$, no vertex is entirely surrounded by edges. It remains to show that there is in fact no other edge $\{v_{\widehat{p}}, v_{\widehat{q}}\}$ that covers v_p. To this end, we assume that there exists such an edge and note that this implies that one of the following 4 cases holds (recall that $p + 1 = q'$):

1. $\widehat{p} < p' < p < q' < q < \widehat{q}$,
2. $\widehat{p} < p' < p < q' < \widehat{q} < q$,
3. $p' < \widehat{p} < p < q' < q < \widehat{q}$,
4. $p' < \widehat{p} < p < q' < \widehat{q} < q$.

We can now show in a similar way as above, that cases 2 to 4 imply that α is not closely entwined and case 1 implies that there exists a variable $x \in \mathrm{var}(\alpha)$ such that $\alpha = \beta \cdot x \cdot \gamma \cdot x \cdot \delta$ with $|\gamma|_x = 0$ and γ is not nested. This contradicts our assumption that α is mildly entwined and, thus, we can conclude that in fact no edge other than $\{v_{p'}, v_{q'}\}$ covers v_p. If in step 3 vertex v_p is covered below instead of above, then an analogous argumentation can be used. This shows that the diagram drawn by the above procedure satisfies condition $(*)$, which proves that $\mathcal{G}$ is outerplanar. □

It is a well known fact that the class of outerplanar graphs has a bounded treewidth:

Theorem 21 (Bodlaender [5])**.** *If $\mathcal{G}$ is an outerplanar graph, then* $\mathrm{tw}(\mathcal{G}) \leq 2$.

Consequently, by Lemma 20 and Theorem 21, the class of mildly entwined patterns has bounded treewidth. Using Corollary 11, we can conclude that the membership problem for mildly entwined patterns is decidable in polynomial time.

Theorem 22. *Let $Z \in \{\mathrm{E}, \mathrm{NE}\}$ and let P be the class of mildly entwined patterns. The problem Z-PATMem(P) is solvable in polynomial time.*

We now compare patterns with bounded scope coincidence degree and mildly entwined patterns. If a pattern has a scope coincidence degree of 1, then it is a non-cross pattern and, thus, it is also mildly entwined. The converse of this statement is not true, i. e., there are mildly entwined patterns with an arbitrarily large scope coincidence degree. This is illustrated by the pattern

$\alpha := x_1 \cdot x_2 \cdot \cdots \cdot x_k \cdot x_k \cdot x_{k-1} \cdot \cdots \cdot x_1$, $k \in \mathbb{N}$. It can be easily verified that α is nested and, thus, also mildly entwined and, furthermore, $\mathrm{scd}(\alpha) = k$. Consequently, for every $k \geq 2$, the class of patterns with a scope coincidence degree of at most k and the class of mildly entwined patterns are incomparable, which shows that by our general approach, we have identified two completely different structural aspects of patterns that both contribute to the complexity of the membership problem.

We conclude this section by mentioning that the concept of outerplanarity can be generalised to k-outerplanarity in the following way. The 1-outerplanar graphs are exactly the outerplanar graphs and, for every $k \geq 2$, a graph is k-outerplanar if and only if it can be drawn on the plane in such a way that no two edges cross each other and, furthermore, if we remove all vertices on the exterior face and all their adjacent edges, then all remaining components are $(k-1)$-outerplanar. It can be shown that if a graph $\mathcal{G}$ is k-outerplanar, then $\mathrm{tw}(\mathcal{G}) \leq 3^k - 1$ (see Bodlaender [5] for further details on k-outerplanarity). Consequently, the definition of mildly entwined patterns can be generalised to a parameter of patterns that corresponds to the k-outerplanarity of their standard α-structures. However, it is not straightforward to identify such a parameter of patterns, and therefore it is left to future research.

6 Conclusions

In this work, we have defined a way of encoding patterns as relational structures, and we have shown that any parameter of patterns that is an upper bound for the treewidth of these encodings, if restricted, allows the membership problem for pattern languages to be solved in polynomial time. We have then applied this meta-result in order to prove that all classes of patterns with a bounded scope coincidence degree and the class of mildly entwined patterns have a polynomial time membership problem.

In the definition of an α-structure (Definition 7), there are several different ways of how the relation symbol E can be interpreted. Thus, for a single pattern α, there are many possible α-structures that all permit an application of Theorem 11. However, the standard way of encoding patterns (Definition 8) has turned out to be sufficient for all results in the present paper. It would be interesting to know whether or not, for some pattern α, there exists an α-structure $\mathcal{A}_\alpha$ that is better than the standard one, i. e., $\mathrm{tw}(\mathcal{A}_\alpha) < \mathrm{tw}(\mathcal{A}_\alpha^s)$. We conjecture that this question can be answered in the negative.

Section 5 constitutes an application of a more general technique that can be described in the following way. We consider an arbitrary class A of graphs with bounded treewidth and then we identify a class of patterns P and a polynomial time computable function g that maps the patterns of P to α-structures such that $\widehat{P} \subseteq A$, where $\widehat{P} := \{\mathcal{G}_\alpha \mid \alpha \in P, \mathcal{G}_\alpha \text{ is the Gaifman graph of } g(\alpha)\}$. Ideally, the class P can be characterised in terms of a parameter or a property of patterns that can be computed in polynomial time.

This indicates that, by applying the above described general technique, other

classes of patterns with a polynomial time membership problem can be found, for example, by using the class of k-outerplanar graphs as the class of graphs with bounded treewidth, as outlined at the end of Section 5.

Acknowledgements

The authors wish to thank the anonymous referees of the conference version [20] of this paper for their valuable comments on the results of Section 4, for pointing out the similarities between the membership problem for pattern languages and parameterised pattern matching, and for providing references to the respective literature.

References

[1] A. Amir, Y. Aumann, R. Cole, M. Lewenstein, and E. Porat. Function matching: Algorithms, applications, and a lower bound. In *Proc. 30th International Colloquium on Automata, Languages and Programming, ICALP 2003*, pages 929–942, 2003.

[2] A. Amir and I. Nor. Generalized function matching. *Journal of Discrete Algorithms*, 5:514–523, 2007.

[3] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[4] B. S. Baker. Parameterized pattern matching: Algorithms and applications. *Journal of Computer and System Sciences*, 52:28–42, 1996.

[5] H.L. Bodlaender. Classes of graphs with bounded tree-width. Technical Report RUU-CS-86-22, Department of Information and Computing Sciences, Utrecht University, 1986.

[6] J. Bremer and D. D. Freydenberger. Inclusion problems for patterns with a bounded number of variables. In *Proc. 14th International Conference on Developments in Language Theory, DLT 2010*, volume 6224 of *Lecture Notes in Computer Science*, pages 100–111, 2010.

[7] C. Câmpeanu, K. Salomaa, and S. Yu. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14:1007–1018, 2003.

[8] R. Clifford, A. W. Harrow, A. Popa, and B. Sach. Generalised matching. In *Proc. 16th International Symposium on String Processing and Information Retrieval, SPIRE 2009*, volume 5721 of *Lecture Notes in Computer Science*, pages 295–301, 2009.

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

[10] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proc. 8th National Conference on Artificial Intelligence, AAAI 1990*, pages 4–9, 1990.

[11] M. Geilke and S. Zilles. Learning relational patterns. In *Proc. 22nd International Conference on Algorithmic Learning Theory, ALT 2011*, volume 6925 of *Lecture Notes in Computer Science*, pages 84–98, 2011.

[12] O. Ibarra, T.-C. Pong, and S. Sohn. A note on parsing pattern languages. *Pattern Recognition Letters*, 16:179–182, 1995.

[13] S. Lange and R. Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.

[14] A. Mateescu and A. Salomaa. Patterns. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 230–242. Springer, 1997.

[15] Y.K. Ng and T. Shinohara. Developments from enquiries into the learnability of the pattern languages from positive data. *Theoretical Computer Science*, 397:150–165, 2008.

[16] D. Reidenbach. A non-learnable class of E-pattern languages. *Theoretical Computer Science*, 350:91–102, 2006.

[17] D. Reidenbach. Discontinuities in pattern inference. *Theoretical Computer Science*, 397:166–193, 2008.

[18] D. Reidenbach and M. L. Schmid. Finding shuffle words that represent optimal scheduling of shared memory access. In *Proc. 5th International Conference on Language and Automata Theory and Applications, LATA 2011*, volume 6638 of *Lecture Notes in Computer Science*, pages 465–476, 2011.

[19] D. Reidenbach and M. L. Schmid. A polynomial time match test for large classes of extended regular expressions. In *Proc. 15th International Conference on Implementation and Application of Automata, CIAA 2010*, volume 6482 of *Lecture Notes in Computer Science*, pages 241–250, 2011.

[20] D. Reidenbach and M. L. Schmid. Patterns with bounded treewidth. In *Proc. 6th International Conference on Language and Automata Theory and Applications, LATA 2012*, volume 7183 of *Lecture Notes in Computer Science*, pages 468–479, 2012.

[21] R. Reischuk and T. Zeugmann. An average-case optimal one-variable pattern language learner. *Journal of Computer and System Sciences*, 60:302–335, 2000.

[22] P. Rossmanith and T. Zeugmann. Stochastic finite learning of the pattern languages. *Machine Learning*, 44:67–91, 2001.

[23] T. Shinohara. Polynomial time inference of extended regular pattern languages. In *Proc. RIMS Symposium on Software Science and Engineering*, volume 147 of *Lecture Notes in Computer Science*, pages 115–127, 1982.

[24] T. Shinohara. Polynomial time inference of pattern languages and its application. In *Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science*, pages 191–209, 1982.

[25] F. Stephan, R. Yoshinaka, and T. Zeugmann. On the parameterised complexity of learning patterns. In *Proc. 26th International Symposium on Computer and Information Sciences, ISCIS 2011*, pages 277–281.

[26] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 7, pages 389–455. Springer, 1997.