Mariusz Flasiński

A SURVEY ON SYNTACTIC PATTERN RECOGNITION METHODS IN BIOINFORMATICS

Early bird

Abstract *Formal tools and models of syntactic pattern recognition which are used in bioinformatics are introduced and characterized in the paper. They include, among others: stochastic (string) grammars and automata, hidden Markov models, programmed grammars, attributed grammars, stochastic tree grammars, Tree Adjoining Grammars (TAGs), algebraic dynamic programming, NLC- and NCE-type graph grammars, and algebraic graph transformation systems. The survey of applications of these formal tools and models in bioinformatics is presented.*

Keywords bioinformatics, syntactic pattern recognition, generative grammar, hidden Markov model (HMM)

Citation Computer Science 25(1) 2024: 107–142

Copyright © 2024 Author(s). This is an open access publication, which can be used, distributed and reproduced in any medium according to the Creative Commons CC-BY 4.0 License.

1. Introduction

There are two main subareas in pattern recognition: the decision-theoretic subarea [42, 114], including the neural network-based approach [156, 167], and the syntactic/structural one. The latter subarea can be divided, in turn, into the structural approach [25] and syntactic pattern recognition, SPR, [60, 69]. In syntactic pattern recognition, a pattern takes the form of a string, a tree or a graph and a set of (structural) patterns is treated as a formal language. Then, a generative grammar is defined as a generator of this language and a syntax analyzer (formal automaton) is constructed for recognizing and/or interpreting of structural patterns. There are three groups of syntactic pattern recognition models depending on a type of a structure considered, namely: string-based models, tree-based models and graph-based models. We use this taxonomy for the presentation of syntactic pattern recognition methods in the paper. From the methodological point of view, syntactic pattern recognition is preferred if patterns considered are structural, a recognition process is multilevel and hierarchy-oriented and a structure-based interpretation is required [60].

As we will see in the next section, there are important issues considered in bioinformatics that can be characterized with the methodological requirements mentioned above. Indeed, syntactic pattern recognition has delivered formal models for recognizing and interpreting structural patterns in bioinformatics from the very beginning. In fact, the first SPR application took place in bioinformatics in the early 1960s. (The term *bioinformatics* had not yet been coined.) This was the development of the FIDAC system for karyotype analysis by Robert S. Ledley and his collaborators [120, 121].

The research areas and important problems of bioinformatics in the context of syntactic pattern recognition methods are introduced in Section 2. The basic formal tools and models of syntactic pattern recognition which are used in bioinformatics are characterized in the third section. It allows us to refer to these models in Section 4, in which the survey of syntactic pattern recognition applications in bioinformatics is presented. The last section contains conclusions.

Early bird

2. Issues of bioinformatics and syntactic pattern recognition

There are three basic formal tools in syntactic pattern recognition: a (generative) grammar, a syntax analyzer (formal automaton, parser) and a language inference (induction) algorithm [60]. A grammar is a formal tool for the generating of a set of strings/sequences (trees, graphs) which is treated here as a formal language. Thus, the grammar models sequences (structures) *via* their generation. On the other hand, a syntax analyzer (automaton) is a formal tool for the recognition/classification of a set of sequences (structures). Thus, it models sequences (structures) *via* their analysis. In any case, both formal tools are used for the modeling of sequences/structures in order to better understand their structural properties. The typical applications of these formal tools in bioinformatics include: finding a subsequence/substructure that relates to important features/functions in the whole sequence/structure, aligning sequences, predicting sequences on the basis of specific features, modeling a higher-level structure on the basis of a lower-level one (e.g. RNA tertiary structure on the basis of its secondary structure) and the like. A language inference (induction) algorithm is an algorithm which generates (automatically) a grammar or a syntax analyzer (automaton) on the basis of a sample of sequences (structures). In fact, it is a learning formal tool, i.e. it learns a model (represented by a grammar/automaton) on the basis of examples. For example, a task of this formal tool can be defined in the following way. Given a set of biological sequences, construct a stochastic automaton (or a grammar) that models these sequences.

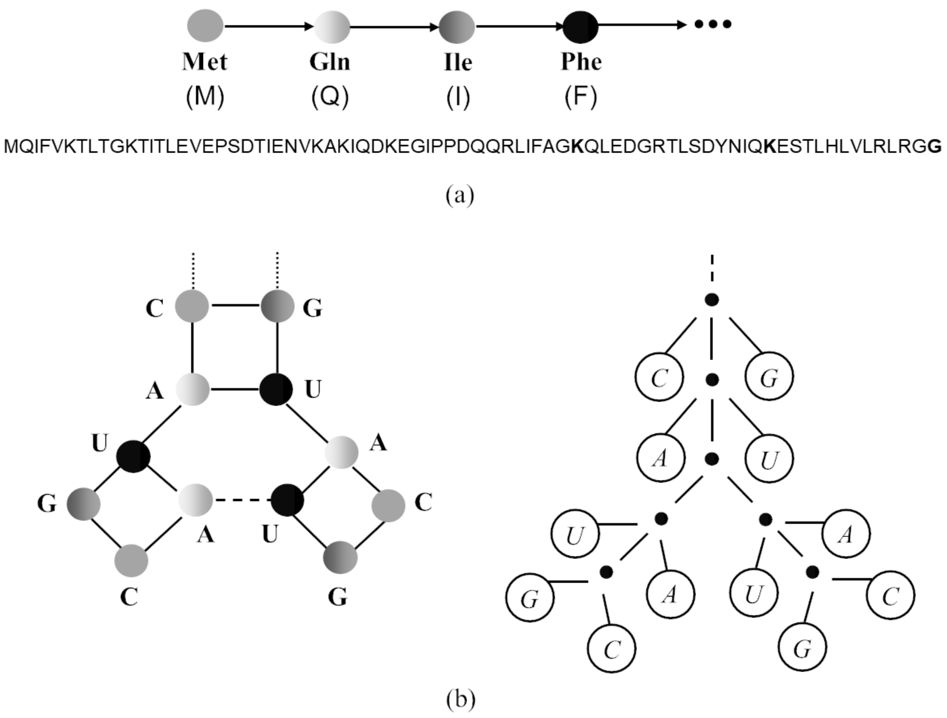
Bioinformatics applies (and sometimes develops) models of computer science in order to better understand biological processes. These models and software systems constructed on their bases are especially useful when the data sets to be analyzed and interpreted are complex and large. The main research areas of bioinformatics include:

- *sequence analysis*,
- *structural bioinformatics*,
- *gene and protein expression*,
- *analysis of cellular organization* and
- *network and systems biology.*

Syntactic pattern recognition models have been applied especially in the first three areas. We will characterize them in a general way by identifying their main problems, since we refer to these problems in Section 4 which contains the survey of syntactic pattern recognition applications in bioinformatics.

Sequential structures are structures which are the most frequently considered in bioinformatics. Their constitute the primary structures of DNA, RNA and proteins. For example, the primary structure of the form of the amino acids' sequence of the ubiquitin protein is shown in Fig. 1 (a). *Sequence analysis* consists in the analysis of DNA, RNA or protein sequence in order to understand their features, biological function or evolution. Its main issues involve, among others: a sequence alignment, a sequence assembly, and a gene prediction. A sequence alignment in bioinformatics is a way of arranging sequences (DNA, RNA, protein) in order to identify regions of similarity which may result from structural, functional or evolutionary relations. There are two issues here: a pairwise sequence alignment (an analysis of two sequences) and a multiple sequence alignment (an analysis of more than two sequences at a time). A sequence assembly consists in aligning and merging of fragments that belong to a longer sequence in order to obtain the original sequence. A gene prediction consists in finding the parts of genomic DNA that encode genes, mainly by identifying the stop and start regions of genes (which is called a gene annotation). Since sequence analysis problems concern an identification, recognition and/or interpretation of sequence, string-based models are the most convenient formal tools in this case. In Section 3.1 we present these models, including: stochastic grammars, stochastic automata, hidden Markov models, programmed grammars, and attributed grammars.

Early bird



MQIFVKTLTGKTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLIFAG**K**QLEDGRTLSDYNIQ**K**ESTLHLVLRLRG**G**

Figure 1. (a) The beginning of the primary structure (the sequence of amino acids) of the ubiquitin protein (M stands for methionine (Met), Q stands for glutamine (Gln), I stands for isoleucine (Ile), F stands for phenylalaline (Phe) etc.) and its complete string representation. (b) The exemplary part of the secondary structure of RNA (the branched RNA structure, and its tree representation generated by Tree Adjoining Grammar. Adapted from: M. Flasiński, *Syntactic Pattern Recognition*, World Scientific, New Jersey-London-Singapore, 2019.

Early

Structural bioinformatics involves the analysis and prediction of higher-level, three-dimensional structure of proteins, RNA, and DNA. For example, a part of the secondary structure of RNA (the branched RNA structure, and its tree representation is shown in Fig. 1 (b), whereas the graph structure of the nucleobase cytosine used in the modeling of the tertiary structure of RNA is shown in Fig. 2. In case of proteins four structural levels are identified: the primary level that can be represented by sequences and three higher levels (secondary, tertiary, and quaternary) that are usually represented by trees or graphs. Protein structure prediction is one of the most important issues in structural bioinformatics, since the structure of a protein relates to its function. Therefore, the problem is crucial for medicine (drug design) as well as for biotechnology (novel enzymes design). It can be defined as the prediction of the secondary level- and tertiary level-structure on the basis of the primary level-

(sequential) structure. The structures of RNA and DNA are represented by trees or graphs in bioinformatics. Therefore, tree-based and graph-based models are used in this case. In Section 3.2 we present stochastic tree grammars, Tree Adjoining Grammars and algebraic dynamic programming, whereas in Section 3.3 we introduce NLC- and NCE-type graph grammars and algebraic graph transformation systems.

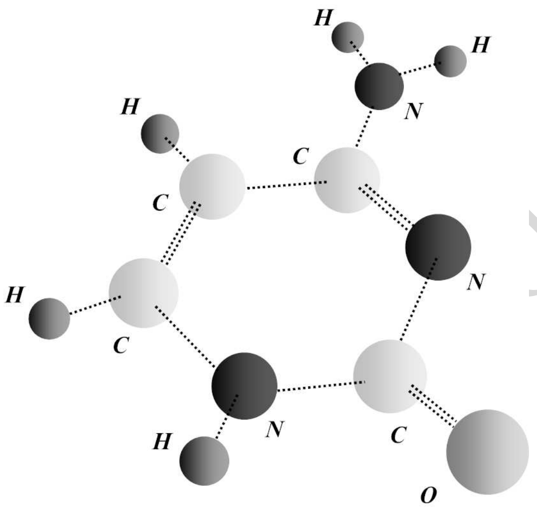


Figure 2. The graph structure of the nucleobase cytosine – a building block for the modeling of the tertiary structure of RNA.

Gene and protein expression area studies three main issues: an analysis of a gene expression, a gene regulation, and an analysis of protein expression. This area contributes to medicine, pharmacy, and agriculture considerably. A gene expression consists in affecting a phenotype by information from a gene. This information is used in the synthesis of a functional gene product (RNA, protein). A gene regulation, in turn, is a process of the increasing/decreasing of the production of gene products by cells as a result of the appearing of some signal. String-based models of syntactic pattern recognition are used in the area of gene and protein expression.

Early

3. Basic formal tools of syntactic pattern recognition for bioinformatics

Basic definitions and characteristics of main classes of grammars and automata used for bioinformatics are contained in this section. The string-, tree- and graph-based models are presented in the succeeding subsections.

3.1. String-based models

The generative power of grammars (and the discriminative power of the corresponding automata) of the standard Chomsky model is sometimes too small for their effective use in the real-world applications. Therefore, a variety of enhanced grammars and automata have been defined to solve this problem [60]. The most useful approaches include: stochastic grammars/automata [67], fuzzy grammars/automata [207], error-correcting automata [192], hidden Markov models [11], and other enhanced models, e.g., programmed grammars [160], attributed grammars [113], and vague languages/multi-derivational parsing [62].

Computational biologists usually reason in the presence of uncertainty, because many facts are missing and often data are noisy. In order to handle this problem, probabilistic models, e.g. Bayesian inference, Markov Random Fields, variational methods, Bayesian networks etc., are applied in bioinformatics [9, 43]. The sequence analysis tasks of modeling, aligning, predicting etc. which have been discussed in the previous section, are of the probabilistic nature as well. Therefore, enhanced probabilistic formal tools of syntactic pattern recognition are often used in bioinformatics. Let us begin their presentation with stochastic regular grammars [17, 67, 71, 85, 163].

Definition 1. *A stochastic regular grammar is a quadruple*

$$G = (\Sigma_N, \Sigma_T, P, S), \text{ where}$$

Σ_N *is a set of nonterminal symbols,*
Σ_T *is a set of terminal symbols,*
P *is a set of stochastic productions of the form:*

$$A_i \xrightarrow{p_{ij}} \gamma_{ij}, \quad i = 1, \ldots, n, \ j = 1, \ldots, m_i,$$

in which $A_i \in \Sigma_N$, $\gamma_{ij} \in \Sigma_T \cup \Sigma_T\Sigma_N$, p_{ij} *is the probability related to the application of the production such that*

$$0 < p_{ij} \leq 1 \quad , \quad \sum_{j=1}^{m_i} p_{ij} = 1 \ ,$$

S *is the start symbol (axiom),* $S \in \Sigma_N$. □

Thus, a stochastic grammar is a standard (Chomsky) grammar such that probabilities have been ascribed to productions. In this case a derivation definition has to be modified slightly. Let the string θ be derived directly from the string β, denoted $\beta \overset{p_{ij}}{\Longrightarrow} \theta$, as the result of applying the production $A_i \xrightarrow{p_{ij}} \gamma_{ij}$.

We say that α_1 derives α_r with the probability $p = \prod_{k=1}^{r} p_k$, denoted $\alpha_1 \overset{p}{\underset{*}{\Longrightarrow}} \alpha_r$ iff there exists the following sequence of derivational steps

$$\alpha_k \overset{p_k}{\Longrightarrow} \alpha_{k+1} \ , \ k = 1, \ldots, r-1 \ .$$

The stochastic language generated by the grammar is defined as follows.

Definition 2. *The language generated by the stochastic regular grammar* $G = (\Sigma_N, \Sigma_T, P, S)$ *is the set*

$$L(G) = \{(\phi, p(\phi)) : \phi \in \Sigma_T^*, \quad S \overset{p_v}{\underset{*}{\Longrightarrow}} \phi, \ v = 1, \ldots, s, \quad p(\phi) = \sum_{v=1}^{s} p_v\},$$

Early bird

where s is the number of all the different derivations of ϕ from S and p_v is the probability of the vth derivation of ϕ. □

For stochastic grammars of various types (i.e., regular, context-free etc.), the corresponding classes of stochastic automata (i.e., finite-state, pushdown etc.) are defined. They differ from their standard counterparts in the ascribing of probabilities to their transitions. Thus, stochastic finite-state automaton (FSA) is defined in the following way [71, 153, 194].

Definition 3. *A stochastic finite-state automaton is a quintuple*

$$A = (Q, \Sigma_T, \Pi, \pi_0, \pi_F), \text{ where}$$

Q is a set of n states,
Σ_T is a finite set of input symbols,
Π is a mapping of Σ_T into the set of $n \times n$ stochastic state-transition matrices such that

$$\Pi(a) = [\pi_{ij}(a)]_{n \times n} \ , \ \pi_{ij} \geq 0 \ , \ \sum_{j=1}^{n} \pi_{ij} = 1 \ , \ i = 1, \ldots, n \ ,$$

where $\pi_{ij}(a)$ is the probability of the transition from state q_i to state q_j when the symbol a has been read,
π_0 is an n-dimensional row vector representing the initial state distribution such that its first component is equal to 1 and the remaining components are equal to 0,
π_F is an n-dimensional column vector such that its kth component is equal to 1 if q_k is the final state and 0 otherwise. □

A stochastic FSA corresponds to a Markov chain defined in the theory of stochastic processes. In both cases, i.e. a stochastic FSA and a Markov chain, we assume that we the probabilities for sequences of observable events are known. (That is, a stochastic process is observable which means that any transition between two states in a stochastic FSA is related to one symbol.) In bioinformatics, however, such an assumption is too strong, i.e. the events we are interested in can be not observable directly. In the theory of syntactic pattern recognition we use an enhanced model of a stochastic FSA, namely *hidden Markov model, HMM* in such a case. (HMMs were firstly applied in the 1960s in the field of Natural Language Processing.) Then, in case of bioinformatics, a hidden Markov model transits through a series of "hidden" states, modeling a biological sequence (denoting e.g. a protein) by *emitting* succeeding terminal symbols (corresponding to e.g. amino acids). (In the case of *HMMs* we say that a terminal symbol is *emitted* instead of saying that it is *generated/read.*) Any state of a HMM does not have to be related one-to-one to the event observed (as in case of stochastic finite-state automata), but the probability distribution for a set of terminal symbols is defined for each state independently. Let us formalize our considerations with the following definition [11, 138].

Definition 4. *A hidden Markov (HMM) model is a quintuple*

$$HMM = (Q, \Sigma_T, \Pi, E, \pi_0), \text{ where}$$

$Q = \{q_1, q_2, \ldots, q_N\}$ is a set of N states,
$\Sigma_T = \{a_1, a_2, \ldots, a_M\}$ is a finite set of M symbols,

Early bird

$\Pi : Q \times Q \to \mathbb{R}_{\geq 0}$ *is the state-transition probability distribution,*
$E : Q \times \Sigma_T \to \mathbb{R}_{\geq 0}$ *is the state-based symbol emission probability distribution,*
$\pi_0 = [\pi(1), \pi(2), \ldots, \pi(N)]$ *is the initial state distribution vector,*
and the following conditions hold:

$$\forall q' \in Q \quad \sum_{q'' \in Q} \Pi(q', q'') = 1 \quad , \quad \sum_{a \in \Sigma_T} E(q', a) = 1 \quad , \quad \sum_{i=1}^{N} \pi(i) = 1 \ . \quad \square$$

$\Pi(q_i, q_j) = \pi_{ij}, i, j = 1, \ldots, N$ is the probability of the transition from state q_i to state q_j. $E(q_j, a_m) = e_j(a_m), j = 1, \ldots, N,\ m = 1, \ldots, M$ is the probability of emitting a_m in state q_j. $\pi(i), i = 1, \ldots, N$ is the probability the Markov chain starts in state q_i.

In the theory of syntactic pattern recognition, error-correcting automata [192] are applied in two cases [60]. Firstly, they are used, if we have to analyze distorted/deformed versions of structural representations of reference (template) patterns. Secondly, they can model the family of variant patterns belonging to the same category (class), yet differing from each other in some detailed structural features. Then, the "error"-transformations are defined for the string representations and the *expanded grammar* is constructed by adding "error"-productions which model these structural differences in the variant patterns. Finally, the error-correcting automaton that, apart from normal states, contains error-states (and error-transitions) is constructed. The biological sequences usually come in families. Then, the sequences which belong to the same family diverge from each other. For modeling protein family assignment, multiple sequence alignment, protein structure prediction, alignment segmentation, etc. hidden Markov models, presented above, also have been enhanced, by introducing the so-called, *profile hidden Markov models* [46, 87, 115].

In order to formalize our considerations, we introduce the notion of (error) string transformation [69, 125] as in [60].

Let there be given two strings $x, y \in \Sigma_T^*$. A transformation $\mathcal{F} : \Sigma_T^* \longmapsto \Sigma_T^*$ such that $y \in \mathcal{F}(x)$ is called a *string transformation.* The following string (error) transformations are defined

- Substitution transformation $\mathcal{F}_S \,:\, \eta_1 a \eta_2 \overset{\mathcal{F}_S}{\longmapsto} \eta_1 b \eta_2$, $a, b \in \Sigma_T$, $a \neq b,\ \eta_1, \eta_2 \in \Sigma_T^*$.
- Insertion transformation $\mathcal{F}_I \,:\, \eta_1 \eta_2 \overset{\mathcal{F}_I}{\longmapsto} \eta_1 a \eta_2$, $a \in \Sigma_T,\ \eta_1, \eta_2 \in \Sigma_T^*$.
- Deletion transformation $\mathcal{F}_D \,:\, \eta_1 a \eta_2 \overset{\mathcal{F}_D}{\longmapsto} \eta_1 \eta_2$, $a \in \Sigma_T,\ \eta_1, \eta_2 \in \Sigma_T^*$.

For defining profile hidden Markov models, only insertion and deletion transformations are used. A profile hidden Markov model is just a hidden Markov model such that three types of states, namely: (normal) match states, insert states (modeling insertion transformation), and delete states (modeling deletion transformations), are distinguished, and its generic structure is defined as it is shown in Fig. 3. Summing up, profile hidden Markov models can be considered to be *error-correcting* hidden Markov models, as it understood in syntactic pattern recognition.

Early bird

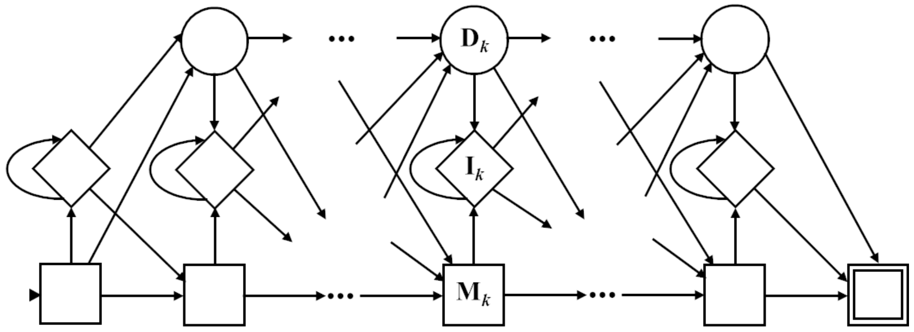


Figure 3. The generic structure of a profile HMM. (Normal (match) states are represented with squares, insert states are represented with diamonds and delete states are represented with circles. The begin state is marked with small black triangle and the end state is marked with double square.)

Early bird

Two basic classes of Chomsky grammars are used in syntactic pattern recognition: (weaker) regular grammars and (stronger) context-free grammars (CFGs). However, sometimes even CFGs are too weak if a generative power is concerned, i.e. a language (a set of sequential patterns) is too complex to be generated by any CFG. For example, a language considered can be context-sensitive (CSL). In bioinformatics such a problem arises quite frequently, e.g. in the case of some RNA pseudoknotted structures [43,157]. In syntactic pattern recognition, such structures can be viewed as crossing interactions which can be modeled with the help of the copy language L_c that is of the form $L_c = \{ww : w \in \Sigma_T^*\}$. However, L_c is the context-sensitive language generated by context-sensitive grammars (CSGs). The problem is that context-sensitive grammars are inefficient computationally and therefore they are not used in practical applications. Such a problem is effectively solved in syntactic pattern recognition by defining various classes of *enhanced CFGs* which can generate certain context-sensitive languages. (For a review of such enhanced grammars, see [60].) Programmed CFGs, introduced in [160], are one of the most popular enhanced CFGs. Let us present their definition.

Definition 5. *A programmed context-free grammar is a quintuple*

$$G = (\Sigma_N, \Sigma_T, J, P, S), \text{ where}$$

Σ_N *is a set of nonterminal symbols,*
Σ_T *is a set of terminal symbols,*
J *is a set of production labels,*
P *is a finite set of productions of the form:*

$$(r) \quad A \to \beta \quad S(U) \quad F(W), \text{ in which}$$

$A \to \beta$, $A \in \Sigma_N$, $\beta \in \Sigma^*$, *is called the core,* (r) *is the production label,* $r \in J$, $U \subset J$ *is the success field and* $W \subset J$ *is the failure field,*
S *is the start symbol (axiom),* $S \in \Sigma_N$. □

A derivation in a programmed CFG can be defined as follows. Firstly, the production labeled with (1) is applied. If any production is applied, then after its application the next production is chosen from its success field U. Otherwise, the next production is chosen from the failure field W. Intuitively speaking, a programming mechanism allows us to control the choice of subsequent productions during a derivation, and this way to force the applying of some (desirable) productions in case a certain production has been applied before. For example, we can generate the (context-sensitive) copy language L_c with a programmed context-free grammar.

The extension of programmed CFGs, namely (dynamically programmed) DPLL(k) grammars have been defined in [61]. They are more efficient computationally, i.e. their syntax analyzer is only of the $\mathcal{O}(n^2)$ time complexity. Their extensions to the error-correcting model and the stochastic model have been defined as well [60]. DPLL(k) grammars can generate such typical (complex) context sensitive-languages like, e.g., $L_1 = \{a^n b^n c^n : n \geq 0\}$, $L_2 = \{a^n b^m c^n d^m : n, m \geq 0\}$ [60].

If symbolic/structural information on structural patterns that is represented by a formal language/grammar should be supplied with numerical information, then attribute grammars are used in syntactic pattern recognition. Such a use of numerical information can be required in the case of minimum-distance alignment or folding operations performed for biological sequences [123,171]. Let us introduce the following notions and definitions.

Let A_X denote the set of attributes of the symbol $X \in \Sigma$, $X_\bullet\alpha$ denote the attribute α of X, D_α denote the set of possible values for the attribute α.

Let $(p)\ \ X^0 \ \rightarrow\ X^1 X^2 \ldots X^m$ be a production of a context-free grammar and $A^{(p)} = A_{X^0} \cup A_{X^1} \cup A_{X^2} \cup \ldots \cup A_{X^m}$. A *semantic rule* for the production (p) is an expression of the following form

$$\beta := f(\gamma_1, \gamma_2, \ldots, \gamma_k), \text{ where}$$

$\beta, \gamma_1, \gamma_2, \ldots, \gamma_k \in A^{(p)}$,
$f : D_{\gamma_1} \times D_{\gamma_2} \times \ldots \times D_{\gamma_k} \ \rightarrow\ D_\beta$ is a function. The set of semantic rules for the production (p) is denoted by $R^{(p)}$.

Now, we can present attributed context-free grammars as in [69, 113].

Definition 6. *An attributed context-free grammar is a sextuple*

$$G = (\Sigma_N, \Sigma_T, P, S, A, R), \textit{ where}$$

Σ_N, Σ_T, P, S *are defined as for a context-free grammar,*
$A = \bigcup_{X \in \Sigma} A_X$ *is a finite set of attributes,*
$R = \bigcup_{p \in P} R^{(p)}$ *is a finite set of semantic rules.* □

Since values can be ascribed to attributes according to semantic rules related to productions (syntactic rules) of a grammar, the corresponding syntax analyzer can compute certain measures during succeeding steps of parsing. These measures can be, then, used for the evaluation of distances between analyzed sequences, which is very useful in case of operations performed for biological sequences mentioned above.

Early bird

3.2. Tree-based models

As we have mentioned in Section 2, tree languages are used mainly for the analysis and prediction of higher-level structures. It includes: the prediction of protein secondary structures, the prediction of RNA secondary structures (cf. Fig. 1 (b) in Section 2), and the prediction of tertiary interactions over pseudoknots for RNA secondary structures. In this section, the most popular tree-based models used in these tasks are presented subsequently, i.e.: (stochastic) tree grammars, Tree Adjoining Grammars, and Algebraic Dynamic Programming.

We introduce the notions concerning tree structures [18, 69, 73, 84] as in [60].

Let $\mathcal{U} = (\mathbb{N}_+, \bullet, \lambda)$, where $\mathbb{N}_+$ is the set of positive integers, $\bullet$ is the operation, λ is the identity, be the free monoid. The partial ordering $\leq$ on $\mathcal{U}$ is defined as follows. $x \leq y,\ x, y \in \mathcal{U}$ iff there exists $z \in \mathcal{U}$ such that $x \bullet z = y$. x and y are incomparable iff $x \not\leq y$ and $y \not\leq x$. $\mathcal{U}$ is called the *Gorn universal tree domain*.

A subset $D \subset \mathcal{U}$ is a *tree domain* iff for all $x, y \in \mathcal{U}$ and all $i, j \in \mathbb{N}_+$ the following conditions are satisfied: (1) if $x \bullet y \in D$ then $x \in D$ and (2) if $x \bullet j \in D$ and $i \leq j$ then $x \bullet i \in D$. The *root* is represented by λ. The *leaves* are the nodes which are maximal with respect to $\leq$. A tree node which is not a leaf is called an *internal node*.

Let $\mathbb{N}$ be the set of nonnegative integers, A be a finite subset of $\mathbb{N}$. A *ranked alphabet* is a pair (Σ, r), where Σ is a finite alphabet, $r : \Sigma \rightarrow 2^A$ is a rank multi-valued mapping. $n \in r(a), a \in \Sigma$ is called the rank of a. We denote $\Sigma_n = \{a : n \in r(a)\}$.

A *tree* over (Σ, r) is a function $t : D \rightarrow \Sigma$, D is a tree domain, such that: (1) $t(x) \in \Sigma_0$, if x is a leaf in D and (2) $t(x) \in \Sigma_n$, where $n = max\{i \in \mathbb{N}_+ : x \bullet i \in D\}$, otherwise. The domain of a tree t is denoted by D_t. The set of all finite trees over Σ is denoted by T_Σ. A *node* is a pair $(x, a) \in D \times \Sigma$. The *frontier* of t is the sequence of its leaves.

Let $t \in T_\Sigma$ and $x \in D_t$. The subtree of t at x, denoted t/x, is defined by the function which is the set of pairs $\{(y, a) : (x \bullet y, a) \in t,\ a \in \Sigma\}$.

Now, we can introduce the definition of (expansive) stochastic regular tree grammars [16, 68, 69, 132].

Definition 7. *An (expansive) stochastic regular tree grammar over (Σ_T, r) is a quintuple*

$$G = (\Sigma_N, \Sigma_T, r, P, S),\ where$$

Σ_N is a finite set of nonterminal symbols,
(Σ_T, r) is a ranked alphabet of terminal symbols, $\Sigma_N \cap \Sigma_T = \emptyset$, $\Sigma = \Sigma_N \cup \Sigma_T$,
P is a set of productions of the form:

$$A_i \xrightarrow{p_{ij}} t_{ij},\quad i = 1, \ldots, n,\ j = 1, \ldots, m_i,$$

in which $A_i \in \Sigma_N$, $t_{ij} \in T_\Sigma$ is a tree which either consists of a terminal root and its nonterminal children or consists of a terminal node, p_{ij} is the probability related to the application of the production such that

$$0 < p_{ij} \leq 1 \quad , \quad \sum_{j=1}^{m_i} p_{ij} = 1 \ ,$$

Early bird

$S \in \Sigma_N$ is the start symbol. $\square$

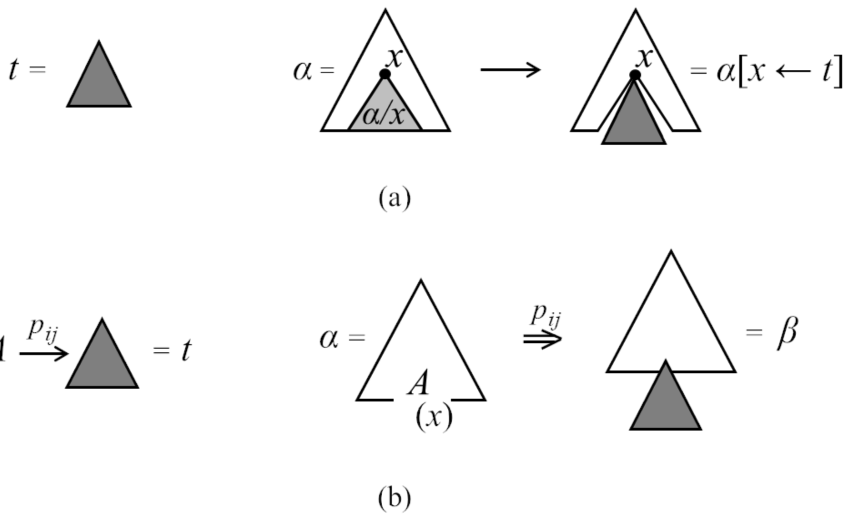


Figure 4. (a) Replacement of a subtree and (b) derivation in stochastic regular tree grammar.

The definition of standard (non-stochastic) tree grammar can be obtained by removing the probabilities in Definition 7.

A derivation step is introduced as a kind of more general operation of subtree replacement. The *replacement* of the subtree α/x by t, denoted $\alpha[x \leftarrow t]$, is the tree defined by the function which is the set of pairs (see Fig. 4 (a))

$$\{(y, \alpha(y)) : y \in D_\alpha \text{ , } x \text{ is not a prefix of } y\} \cup \{(x \bullet z, t(z)) : z \in D_t\} \ .$$

Let $\alpha, \beta \in T_\Sigma$ and $x \in D_\alpha$. α directly derives β with the probability p_{ij} in G, denoted $\alpha \xRightarrow{p_{ij}} \beta$, iff there exists $A \xrightarrow{p_{ij}} t \in P$ such that $\alpha(x) = A$ and $\beta = \alpha[x \leftarrow t]$ (see Fig. 4 (b)). The stochastic tree language generated by the stochastic tree grammar G is defined in an analogous way as the stochastic string language (cf. Definition 2).

Now, we present Tree Adjoining Grammars (*TAGs*) [100–102] according to [60].

Definition 8. *A Tree Adjoining Grammar, TAG, is a quintuple*

$$G = (\Sigma_N, \Sigma_T, S, I, A), \text{ where}$$

Σ_N is a finite set of nonterminal symbols,
Σ_T is a finite set of terminal symbols, $\Sigma_N \cap \Sigma_T = \emptyset$, $\Sigma = \Sigma_N \cup \Sigma_T$,
$S \in \Sigma_N$ is the initial symbol,
I is a finite set of initial trees such that for any $\alpha \in I$ the internal nodes of α are labelled by nonterminals and leaves are labeled by terminals or nonterminals; nonterminal leaves of α are marked for the substitution operation with a special symbol $\downarrow$,
A is a finite set of auxiliary trees such that for any $\beta \in A$ the internal nodes of β

Early

are labelled by nonterminals and leaves are labeled by terminals or nonterminals; non-terminal leaves of β are marked for substitution except for one node, called the foot node; the foot node has the same label as the root of β; the foot node is marked for the adjoining operation with a special symbol $*$. $\Box$

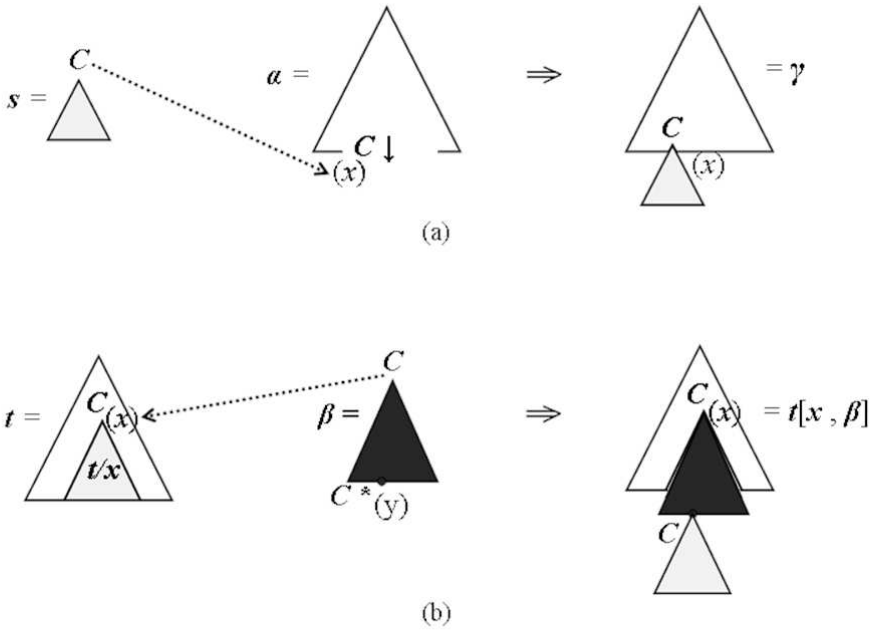


Figure 5. (a) Substitution in TAG. (b) Adjoining in TAG.

The scheme of substitution operation is shown in Fig. 5 (a). A nonterminal leaf marked $\downarrow$ of a derived tree is replaced with some tree s derived from an initial tree. The replaced node should have the same label as the root of s.

The scheme of adjoining operation is shown in Fig. 5 (b). An auxiliary tree β is inserted into an internal node having the address x of a derived tree t. The node of t having the address x should have the same label as the root of β. The subtree t/x is attached to the foot node of β which is marked with $*$.

Let $\theta, \gamma \in T_\Sigma$. θ directly derives γ in G, denoted $\theta \underset{G}{\Longrightarrow} \gamma$, iff either $\gamma = \theta[x, \beta]$, $x \in D_\theta$, $\beta \in A$ or γ results from the application of a substitution operation to θ.

The reflexive and transitive closure of the relation $\underset{G}{\Longrightarrow}$ is denoted with $\overset{*}{\underset{G}{\Longrightarrow}}$. If $\theta \overset{*}{\underset{G}{\Longrightarrow}} \gamma$, then γ is called a *derived tree of* θ. The set of all derived trees of θ is denoted with $DT(\theta)$.

Now, we can define the tree language generated by TAG G.

Definition 9. *The tree language generated by TAG G is the set*

$$T(G) = \{\gamma \in T_\Sigma : \gamma \in DT(\theta),\ \theta \in I,\ \theta(\lambda) = S, \text{ and } Y(\gamma) \in \Sigma_T^*\}. \ \Box$$

At the end of this section, we present the novel efficient approach of *Algebraic Dynamic Programming*, (*ADP*) [**?**, 75, 78–80, 166] which has been developed in bioinformatics. This approach is based on the (well-known in computer science) paradigm of *dynamic programming* which is a generic model of the constructing of efficient algorithms for complex problems which, by definition, involve the searching of a space of exponential size (that is inefficient computationally). The paradigm consists in breaking a complex problem into simpler subproblems recursively (in case these subproblems are shared) which allows one to search the space in polynomial time [12]. Dynamic programming algorithms are widely used in bioinformatics, including: optimal global alignment, local alignment, repeated matching, overlap matching, etc. [43].

Algebraic Dynamic Programming is a systematic methodology of the constructing of dynamic programming algorithms. Two main phases are defined in the methodology: the recognition phase and the evaluation phase.

During the recognition phase a *yield grammar* is used. The concept of *yield* has been introduced for Tree Adjoining Grammars, presented above. Let us introduce this concept according to [60].

Let us define the *yield mapping* $Y : T_\Sigma \longrightarrow \Sigma^*_{T,0}$ in the following way.
(1) If $a \in \Sigma_{T,0}$ then $Y(a) = a$.
(2) If $a \in \Sigma_{T,n}$, $k > 0$ and $t_1, t_2, \ldots, t_n \in T_\Sigma$ then

$$Y(a(t_1 t_2 \ldots t_n)) = Y(t_1) \ \cdot \ Y(t_2) \ \cdot \ \ldots \ \cdot \ Y(t_n)) \ ,$$

where $\cdot$ is the concatenation operation.

Thus, yield mapping delivers the sequence of the labels of the frontier nodes (i.e. the leaves), writing them from left to right.

For tree grammars we can define the tree language generated by them, as it has been made by Definition 9 for Tree Adjoining Grammars. On the other hand, we can also define the string language generated by them in the following way.

Definition 10. *The string language generated by TAG G is the set*

$$L(G) = \{v : v = Y(\gamma),\ \gamma \in T(G)\}.\ \Box$$

In this case the strings defined by the terminal labels of the frontiers of the derived trees are treated as the words of this (string) language. Then, we say that G is the yield grammar. In fact, Tree Adjoining Grammars have been introduced in syntactic pattern recognition for generating *enhanced context-free (string) grammars* that has been discussed in the previous section. The search space of the problem considered is described by the yield grammar.

During the evaluation phase, the so-called *evaluation* Σ*-algebra* (an *interpretation*, as it is understood in algebraic semantics) is used to comprise the aspects relevant to the objective assumed, independently of the description of the search space by the yield grammar. This way the dynamic programming algorithms can be developed on a more abstract level than in the standard dynamic programming approach. *Algebraic Dynamic Programming* methodology has been successfully used, among others, for sequence alignment and RNA folding.

Early bird

3.3. Graph-based models

Graph grammars are the strongest generative formalism in syntactic pattern recognition [60, 69, 145], because every kind of relation among the elements of a structure can be defined. Due to their big generative power, graph grammars have been used for such complex problems in bioinformatics as, e.g.: modeling RNA tertiary structure motifs, modeling RNA folding, modeling protein structures, genetic regulation, analyzing metabolic networks.

There are many classes of graph grammars [50]. In this subsection we present three classes which, on one hand, are classic in the theory of graph grammars and, on the other hand, are applied in bioinformatics. They include: Node Label Controlled (NLC) graph grammars, Neighborhood-Controlled Embedding (NCE) graph grammars, and algebraic (DPO) graph transformation systems.

As we have discussed in Section 3.1, the application of a certain class of a generative grammar is conditioned by the computational efficiency of the corresponding type of a syntax analyzer. In case of graph grammars this problem is especially crucial, because the research into the efficiency of graph parsing revealed a hard membership problem, PSPACE-complete or NP-complete, for graph grammars [20, 95, 181, 195]. (The reasons for the intractability of this problem were identified in [57, 60]). Fortunately, for the graph grammars of the Node Label Controlled (NLC) class (*edNLC* graph grammars), efficient, $\mathcal{O}(n^2)$, top-down ($ETPL(k)$) and bottom-up ($ETPR(k)$) syntax analyzers [53–55, 59, 60] as well as an efficient inference algorithm [58] have been defined. In result $ETPL(k)/ETPR(k)$ subclasses of NLC grammars could have been applied, among others, for scene analysis [53, 55], CAD/CAM integration [56], Polish Sign Language recognition [65]. The error-correcting $ETPL(k)$ syntax analyzer and its attributed version have been used for the recognition of vague/variant patterns [54, 64]. Stochastic $ETPL(k)$ grammars were applied for manufacturing quality control [60], and attributed programmed $ETPL(k)$ grammars - for process monitoring and control [63].

Let us introduce the notions concerning this class of graph grammars according to [94, 95, 97].

A directed node- and edge-labeled graph, *EDG* graph, over Σ and Γ is a quintuple $H = (V, E, \Sigma, \Gamma, \phi)$, where V is a finite, non-empty set of nodes, Σ is a finite, non-empty set of node labels, Γ is a finite, non-empty set of edge labels, E is a set of edges of the form (v, γ, w), in which $v, w \in V, \gamma \in \Gamma$, and $\phi : V \to \Sigma$ is a node-labeling function.

The family of the *EDG* graphs over Σ and Γ is denoted by $EDG_{\Sigma,\Gamma}$. The components V, E, ϕ of a graph H are sometimes denoted with V_H, E_H, ϕ_H.

Let $A = (V_A, E_A, \Sigma, \Gamma, \phi_A)$, $B = (V_B, E_B, \Sigma, \Gamma, \phi_B)$ and $C = (V_C, E_C, \Sigma, \Gamma, \phi_C)$ be *EDG* graphs. An isomorphism from A onto B is a bijective function h from V_A

Early bird

onto V_B such that

$$\phi_B \circ h = \phi_A \;\; \textit{and} \;\; E_B = \{(h(v), \gamma, h(w)) : (v, \gamma, w) \in E_A\}.$$

We say that A is *isomorphic to* B, and denote this with $A \cong B$.

Definition 11. *An edge-labeled directed Node Label Controlled, edNLC, graph grammar is a quintuple*

$$G = (\Sigma, \Sigma_T, \Gamma, P, Z), \; \textit{where}$$

Σ *is a finite, non-empty set of node labels,*
$\Sigma_T \subseteq \Sigma$ *is a set of terminal node labels,*
Γ *is a finite, non-empty set of edge labels,*
P *is a finite set of productions of the form* (l, D, C)*, in which*
$l \in \Sigma \setminus \Sigma_T, \quad D \in EDG_{\Sigma,\Gamma},$
$C : \Gamma \times \{in, out\} \to 2^{\Sigma \times \Sigma \times \Gamma \times \{in,out\}}$ *is the embedding transformation,*
$Z \in EDG_{\Sigma,\Gamma}$ *is the start graph called the axiom.* $\Box$

We have presented definitions for languages which consist of directed node- and edge-labeled graphs. If we use undirected node- and edge-labeled graphs, we denote the family of such graphs by $EG_{\Sigma,\Gamma}$ and the class of the corresponding graph grammars by *eNLC*. If we use undirected node-labeled graphs, we denote the family of such graphs by G_Σ and the class of the corresponding graph grammars by *NLC*. In both cases, the corresponding definitions are just simplified with relation to the definitions of *EDG* graphs and *edNLC* grammars. (The same holds for definitions presented below.)

A direct derivational step in edNLC graph grammars is defined as follows.

Definition 12. *Let* $G = (\Sigma, \Sigma_T, \Gamma, P, Z)$ *be an edNLC graph grammar.*
Let $H, \overline{H} \in EDG_{\Sigma,\Gamma}$*. Then* H *directly derives* $\overline{H}$ *in* G*, denoted by* $H \underset{G}{\Longrightarrow} \overline{H}$*, if there exists a node* $v \in V_H$ *and a production* (l, D, C) *in* P *such that the following holds.*

(a) $l = \phi_H(v)$.

(b) There exists an isomorphism from $\overline{H}$ *onto the graph* X *in* $EDG_{\Sigma,\Gamma}$ *constructed as follows. Let* $\overline{D}$ *be a graph isomorphic to* D *such that* $V_H \cap V_{\overline{D}} = \emptyset$ *and let* h *be an isomorphism from* D *onto* $\overline{D}$*. Then*

$$X = (V_X, E_X, \Sigma, \Gamma, \phi_X), \; \textit{where}$$

$V_X = (V_H \setminus \{v\}) \cup V_{\overline{D}}$,

$$\phi_X(y) = \begin{cases} \phi_H(y), & \textit{if } y \in V_H \setminus \{v\}, \\ \phi_{\overline{D}}(y), & \textit{if } y \in V_{\overline{D}} \; , \end{cases}$$

$E_X = (E_H \setminus \{(n, \gamma, m) : n = v \;\textit{or}\; m = v\}) \cup$
$\cup \; \{(n, \gamma, m) : n \in V_{\overline{D}} \;, m \in V_{X \setminus \overline{D}}$ *and there exists an edge* $(m, \lambda, v) \in E_H$ *such that* $(\phi_X(n), \phi_X(m), \gamma, out) \in C(\lambda, in)\} \cup$
$\cup \; \{(m, \gamma, n) : n \in V_{\overline{D}} \;, m \in V_{X \setminus \overline{D}}$ *and there exists an edge* $(m, \lambda, v) \in E_H$ *such that* $(\phi_X(n), \phi_X(m), \gamma, in) \in C(\lambda, in)\} \cup$

Early bird

$\cup \{(n,\gamma,m) : n \in V_{\overline{D}}\ , m \in V_{X\setminus\overline{D}}$ *and there exists an edge* $(v,\lambda,m) \in E_H$ *such that* $(\phi_X(n),\phi_X(m),\gamma,out) \in C(\lambda,out)\} \cup$
$\cup \{(m,\gamma,n) : n \in V_{\overline{D}}\ , m \in V_{X\setminus\overline{D}}$ *and there exists an edge* $(v,\lambda,m) \in E_H$ *such that* $(\phi_X(n),\phi_X(m),\gamma,in) \in C(\lambda,out)\}$. □

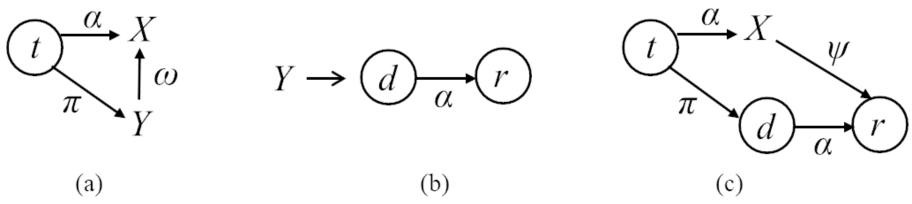


Figure 6. (a) The start graph Z of the *edNLC* grammar G, (b) a production of G and (c) the derived graph h.

Since the definition of a derivation step for *edNLC* graph grammar is a little bit complicated, let us consider the following example. The start graph Z which a production is to be applied for, is shown in Fig. 6 (a).

The left- and right-hand sides of a production to be applied are shown in Fig. 6 (b). The embedding transformation of the production is defined in the following way.
(i) $C(\omega, out) = \{(r, X, \psi, in)\}$,
(ii) $C(\pi, in) = \{(d, t, \pi, in)\}$.
A derived graph h (the result of applying the production to the start graph Z) is shown in Fig. 6 (c).

The derivation step has two phases. During the first phase, the node labeled with Y of the graph Z is removed, and the graph of the right-hand side replaces the removed node. The transformed graph obtained by removing the node and its adjacent edges is called the rest graph. During the second phase, the embedding transformation is applied to connect some nodes of the right-hand side graph with the rest graph. The item **(i)** is interpreted as follows.

1. Each edge labeled with ω and going *out* from the node corresponding to the left-hand side of a production, i.e. Y, has to be replaced by
2. the edge:
   (a) which connects the node of the graph of the right-hand side of the production and labeled with r with the node of the rest graph and labeled with X,
   (b) is labeled with ψ,
   (c) and comes *in* to the node r.

One can easily notice that the item **(ii)** just preserves the edge labeled with π.

(Indirect) derivations in the *edNLC* graph grammar G and the language generated by G are defined in an analogous way as for Chomsky (standard) grammars.

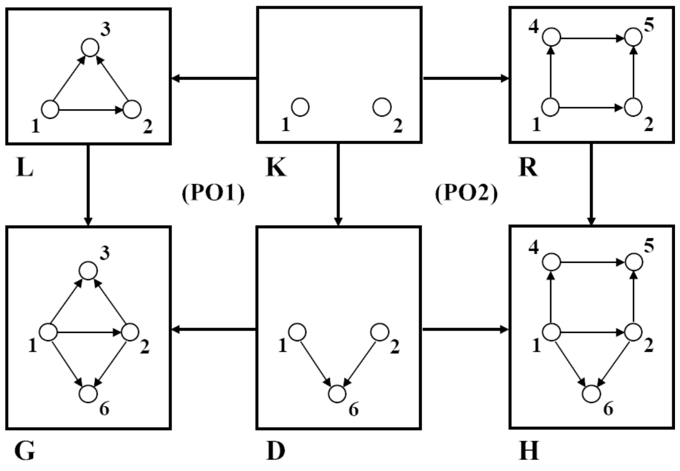
The class of *Neighborhood-Controlled Embedding* (*NCE*) *graph grammars* [96] is the extension (and enhancement) of the class of NLC graph grammars. The left-hand

Early bird

side of a production can be a graph (not only a nonterminal symbol). The embedding transformation uses node identifiers (not node labels) which allows us to distinguish various nodes having the same label.

$$\begin{tikzcd} p\colon (L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "(2)"] & R) \arrow[d] \\ G & D \arrow[l] \arrow[r] & H \end{tikzcd}$$

(a)



(b)

Figure 7. (a) The scheme of DPO graph transformation and (b) its example.

Algebraic graph transformation systems - double pushout model (*DPO*) - were introduced in [51, 52]. Let us present the notions of: a production and a direct graph transformation in the DPO model according to [49].

A *production* in DPO is a triple $p = (L, K, R)$, where L is the left-hand side graph of p, R is the right-hand side graph of p, K is used for defining the gluing conditions.

The scheme of a DPO graph transformation is shown in Fig. 7 (a). Let G be a graph which is to be transformed as the result of the application of a production $p = (L, K, R)$. Let $L \setminus K$ denote the part of G which is to be removed from as the result of the application of p, $R \setminus K$ denote the part of G which is to be added to. A *direct graph transformation* with $p = (L, K, R)$ is performed in the following two steps.
(1) A match m of L in G is found such that m is structure-preserving. Then, all the nodes and edges which are matched with $L \setminus K$ are removed from G. (Let us note

that m should satisfy a gluing condition, i.e. the gluing of $L \setminus K$ and D equals to G.) This step is depicted schematically in the part (1) of in Fig. 7 (a).)
(2) The graph D is glued with $R \setminus K$ in order to obtain the derived graph H, as it is depicted schematically in the part (2) of in Fig. 7 (a). The graph K is used for gluing the nodes and edges which has been newly created into D. (It allows us to define the gluing points at which the right-hand side graph R is embedded into D.)

The example of the DPO graph transformation is shown in Fig. 7 (b).

Both an *indirect graph transformation* and the *graph language* defined by a DPO algebraic graph transformation system are defined in an analogous way as for graph grammars introduced above.

4. Applications of syntactic pattern recognition models in bioinformatics

The survey of the applications of syntactic pattern recognition methods in bioinformatics is presented in this section. Due to the presentation of string-based models, tree-based models and graph-based models in subsections: 3.1, 3.2 and 3.3 of the previous section, we can just refer to these models during the survey of their applications below in subsections: 4.1, 4.2 and 4.3, respectively. The summary of these applications is included in Subsection 4.4.

4.1. Applications of string-based models

In the area of bioinformatics, syntactic pattern recognition methods were applied firstly for chromosome analysis. The research team conducted by R.S. Ledley constructed the FIDAC system for scanning the chromosome photomicrographs for karyotype analysis in the 1960s [82,120,121]. The problem of biological images was studied by R.A. Kirsch [109]. K. S. Fu with collaborators led research into the analysis of photomicrographs of chromosomes in the 1970s. Precedence parsing [122] and stochastic context-free programmed grammars [70,92] were applied. In [192] the error-correcting recognition system was applied. The direct parsing model was presented in [189].

Research into the use of formal languages, grammars and automata in molecular biology and genetics was led in the 1980s and early 1990s [19,22,40,88,168,169]. tRNA modeling was performed with the help of *stochastic context-free grammars* (*CFGs*) [162]. For the parsing of DNA sequences, string variable grammars (an extension of definite clause grammars) were used in [170]. Syntactic pattern recognition-based methods were applied for the identification of regulatory sites in [159]. Stochastic *CFGs* were used for the modeling of RNA pseudoknot structures in [23]. Multiple sequence alignment was performed with the help of multi-tape S-attribute grammars in [123]. The inference of strictly locally testable languages for DNA sequence analysis was presented in [204].

In the area of gene expression and regulation, generative grammars have been used since 1989 [32–34]. Finite-state automata (and transducers) for applications in

Early bird

this area were presented in [22] and HMMs - in [208]. Context-sensitive grammars were applied for describing biological binding operators to model gene regulation in [13]. Modeling gene expression and regulation based on the operon model of Jacob and Monod with the help of finite-state automata was presented in [108]. Attributed context-free grammars were used for testing relationships between DNA sequences and phenotypes in [29]. The identification of the promoter regions with the help of context-free grammars was presented in [36].

Small subunit ribosomal RNA multiple alignments were constructed with the help of stochastic *CFGs* in [24]. The studies into the issue of predicting RNA secondary structures containing pseudoknots resulted in the proof of NP-completeness of this problem [133]. A polynomial time syntax analyzer for augmented *CFGs* generating pseudoknotted structures was constructed in [157]. An ncRNA gene detection was made with the help of pair stochastic *CFGs* in [158]. Basic gene grammars were defined for processing DNA sequences in [124]. The model of grammatical induction for the recognition of human neuropeptide precursors was defined in [141]. A pairwise RNA structure comparison was made with stochastic *CFGs* in [89]. Parallel communicating grammars were constructed for modeling RNA pseudoknotted structures in [28]. The extraction of protein interaction information was made with the help of context-free grammars in [190]. An RNA secondary structure prediction with the help of stochastic *CFGs* was studied in [5, 31, 38, 41, 110, 111]. Dependency grammars were used for the analysis of protein-protein interactions in [164]. Link grammars and they parsing were applied for the extraction of protein interaction information in [176]. The induction of even linear grammars was applied for predicting transmembrane domains in proteins in [147]. The prediction of RNA-RNA interaction was made with stochastic multiple *CFGs* in [105, 107, 175]. The studies of RNA pseudoknotted secondary structures with the help of multiple context-free grammars were presented in [48, 143, 155]. The analysis of protein sequences was performed with the help of stochastic *CFGs* in [44, 45]. The use of inference of regular grammars for larger-than-gene structures was discussed in [193]. Multi-dimensional (based on linear and context-free) grammars were used for DNA-protein alignment in [178]. A grammatical inference method was constructed for classification of amyloidogenic hexapeptides in [200].

Hidden Markov models (HMMs) are one of the most popular syntactic pattern recognition formal tools which are applied in bioinformatics [47, 66]. HMMs were applied for gene/sequence prediction and modeling [26,27,103,117,126,140,142,154,188], sequence alignment [10, 144], protein secondary structure prediction [119, 202], base calling [128, 177], modeling sequencing errors [131], predicting transmembrane protein topology [116, 212], predicting and discriminating beta-barrel outer membrane proteins [6–8], RNA folding and alignment [86], ncRNA identification [180, 209], ncRNA annotation [21, 199], ncRNA structural alignment [206] and identification of protein domains [74, 191]. Novel models based on HMMs were defined in bioinformatics. The most popular ones include: *profile hidden Markov models* [3,15,46,87,90,98,104,115,148,150,183,184,186,187,201] which are used for represent-

Early bird

ing and analyzing sequence profiles and *pair hidden Markov models* [39,43,112,144,198] which are applied for finding sequence alignments by emitting two (aligned) strings. *Generalized hidden Markov models* which emit a string at a state, were applied for gene prediction in [118, 134, 154]. Previously emitted substrings are used for determining the probabilities of future states in *context-sensitive hidden Markov models* [2, 206], which allows one to represent correlations between subsequences. (Standard) HMMs are combined with with continuous Markov chains to define *evolutionary hidden Markov models* [146] used to represent the evolution of biological sequences. Profile HMMs were applied for a viral discovery from metagenomic data in [4].

Fundamental studies into the use of syntactic pattern recognition in bioinformatics and comprehensive synthetics overview were presented in seminal monographs and papers. The most important include [171–173]. The applications of HMMs in bioinformatics were summarized in [83, 205] and the use of formal lingustics tools can be found in [9, 35, 43, 149, 165]. The problem of inferencing stochastic grammars from biological sequences was studied in [161]. The research into the applying of Natural Language Processing for genomics was presented in [203]. The studies into the possible existence of protein grammar which generates folding patterns in protein domains were presented in [151]. The application of computational linguistics for biopolymer structure studies was considered in [37].

Early bird

4.2. Applications of tree-based models

Stochastic tree grammars were used for the prediction of protein secondary structures in [1,135] and for the prediction of RNA/protein tertiary structures in [38]. Tree grammars were applied for the modeling of multiple biomolecular structures in [210, 211], for the computation of exact RNA shape probabilities in [93], for RNA analysis [129] and for RNA pseudoknot comparison in [152]. The mining of human-viral infection patterns was performed with the help of regular tree grammars in [182]. Rectangle tree grammars were used for predicting RNA secondary structures in [127].

RNA structure prediction with the help of *Tree Adjoining Grammars* (*TAGs*) was presented in [196]. TAGs were used for pseudoknot identification in [174]. The generating of RNA secondary structure including pseudoknots with the help of extended simple linear Tree Adjoining Grammars (ESL-TAG) was presented in [106]. This class of TAGs was used to construct an algorithm for tertiary interactions over pseudoknots for the predicting of RNA secondary structures in [91]. Pair stochastic Tree Adjoining Grammars (PSTAG) were used for a pseudoknot RNA structure prediction in [139]. The grammatical representation of macromolecular structures with the help of Tree Adjoining Grammars and related formalisms was proposed in [30].

Algebraic Dynamic Programming (ADP), based on tree grammars, was firstly used for RNA folding [75]. Its applications include: RNA folding [137], aligning recombinant DNA sequences [77], pairwise sequence comparison [76], RNA structure prediction and analysis [80] and the alignment of bio-structure tree representations [14, 81].

4.3. Applications of graph-based models

The induction (inference) of node label controlled (NLC) graph grammars was applied for analyzing protein sequence data in [99]. The modeling of protein structures with the help of neighborhood-controlled embedding (eNCE) graph grammars was presented in [197]. Algebraic (DPO) graph transformation systems were used for modeling RNA folding in [136]. These systems were also applied for analyzing metabolic networks in [179] and for modeling RNA tertiary structure motifs [185]. String-regulated rewriting graph grammars were used for genetic regulation in [130]. The use of an inference algorithm for k-testable graph languages in order to analyze hairpin RNA molecules data sets was presented in [72].

4.4. Summary of applications

A summary of applications of syntactic pattern recognition models in bioinformatics described in the previous subsections is presented in Table 1.

Early bird

Table 1

The summary of applications of syntactic pattern recognition models in bioinformatics (in chronological order)

Model type	Models	References
String-based models	Regular grammars, stochastic context-free grammars, programmed context-free grammars, multiple context-free grammars, context-sensitive grammars, attributed grammars, finite-state automata, hidden Markov models, precedence parsing, CYK parsing, algebraic dynamic programming	[120], [121], [82], [109], [70], [122], [192], [22], [189], [88], [32–34], [168–173], [40], [115, 117], [162], [19], [23], [118], [13], [123], [159], [46, 47], [104], [204], [110], [24], [133], [154], [157, 158], [108], [116], [124], [141], [89], [144], [151], [203], [3], [28], [111], [112], [131], [146], [190], [6–8], [26,27], [41], [201], [39], [103], [134], [161], [183], [31], [105, 107], [142], [198], [148], [186,187], [199], [209], [15], [21], [37], [66], [86], [128], [164], [176], [184], [202], [147], [206], [29], [44], [180], [205], [2], [90], [98], [212], [193], [5], [143], [177], [191], [36], [45], [38], [74], [178], [4], [35], [155], [200], [119], [140], [188], [208], [149], [126], [48], [165]
Tree-based models	(Stochastic) regular tree grammars, Tree Adjoining Grammars, algebraic dynamic programming	[135], [1], [196], [75], [77], [76], [78], [79], [80], [106], [139], [30], [174], [93], [127], [210, 211], [91], [38], [81], [166], [182], [14], [129], [152], [137]
Graph-based models	*NLC* graph grammars, *NCE* graph grammars, algebraic (*DPO*) graph transformation systems, string-regulated rewriting graph grammars, k-testable graph languages	[99], [185], [130], [136], [72], [179], [197]

5. Conclusions

In Section 3 we have presented the basic formal tools of syntactic pattern recognition (SPR) which are used in bioinformatics. As one could see in Section 4, SPR models and methods, namely various classes of generative grammars, syntax analyzers of many types and a lot of language inference (induction) algorithms have been successfully used in bioinformatics. Indeed, the popularity of these methods resulting in plenty of applications in this research area is amazing. At the same time, bioinformatics is an interesting and challenging research area for computer scientists developing novel syntactic pattern recognition models for real-world applications.

References

[1] Abe N., Mamitsuka H.: Predicting protein secondary structure using stochastic tree grammars, *Machine Learning*, vol. 29, pp. 275–301, 1997.

[2] Agarwal S., Vaz C., Bhattacharya A., Srinivasan A.: Prediction of novel precursor miRNAs using a context-sensitive hidden Markov model (CSHMM), *BMC Bioinformatics*, vol. 11 (Suppl 1): S29, 2010.

[3] Ahola V., Aittokallio T., Uusipaikka E., Vihinen M.: Efficient estimation of emission probabilities in profile hidden Markov models, *Bioinformatics*, vol. 19, pp. 2359–2368, 2003.

[4] Alves J., etal: GenSeed-HMM: A tool for progressive assembly using profile HMMs as seeds and its application i Alpavirinae viral discovery from metagenomic data, *Frontiers in Microbiology*, vol. 7, p. 269, 2016.

[5] Anderson J.W.J., Tataru P., Staines J., Hein J., Lyngso R.: Evolving stochastic context-free grammars for RNA secondary structure prediction, *BMC Bioinformatics*, vol. 13:78, 2012.

[6] Bagos P., Liakopoulos T., Hamodrakas S.: Evaluation of methods for predicting the topology of beta-barrel outer membrane proteins and a consensus prediction method, *BMC Bioinformatics*, vol. 6, p. 7, 2005.

[7] Bagos P., Liakopoulos T., Spyropoulos I., Hamodrakas S.: A hidden Markov model method, capable of predicting and discriminating beta-barrel outer membrane proteins, *BMC Bioinformatics*, vol. 5, p. 29, 2004.

[8] Bagos P., Liakopoulos T., Spyropoulos I., Hamodrakas S.: PRED-TMBB: a web server for predicting the topology of beta-barrel outer membrane proteins, *Nucleic Acids Research*, vol. 32, pp. W400–W404, 2004.

[9] Baldi P., Brunak S.: *Bioinformatics: The Machine Learning Approach*, MIT Press, Cambridge, MA, 2001.

[10] Baldi P., Chauvin Y., Hunkapillar T., McClure M.: Hidden Markov models of biological primary sequence information, *Proceedings of the National Academy of Sciences of the USA*, vol. 91, pp. 1059–1063, 1994.

Early bird

[11] Baum L.E., Petrie T.: Statistical inference for probabilistic functions of finite state Markov chains, *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[12] Bellman R.: *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[13] Bentolila S.: A grammar describing biological binding operators to model gene regulation, *Biochimie*, vol. 78, pp. 335–350, 1996.

[14] Berkemer S.J., zu Siederdissen Höner H., Stadler P.: Algebraic dynamic programming on trees, *Algorithms*, vol. 10, p. 135, 2017.

[15] Bernardes J.S., Dávila A.M., Costa V.S., Zaverucha G.: Improving model construction of profile HMMs for remote homology detection through structural alignment, *BMC Bioinformatics*, vol. 8:435, 2007.

[16] Bhargava B.K., Fu K.: Stochastic tree systems for syntactic pattern recognition. In: *Proc. Twelfth Annual Allerton Conference on Circuit and System Theory*, pp. 278–287, Monticello, IL, 1974.

[17] Booth T.L., Thompson R.A.: Applying probability measures to abstract languages, *IEEE Trans Computers*, vol. 22, pp. 442–450, 1973.

[18] Brainerd W.S.: Tree generating regular systems, *Information and Control*, vol. 14, pp. 217–239, 1969.

[19] Bralley P.: An introduction to molecular linguistics, *Bioscience*, vol. 46, pp. 146–153, 1996.

[20] Brandenburg F.J.: On the complexity of the membership problem of graph grammars. In: *Proc. Int. Workshop on Graphtheoretic Concepts in Computer Science*, pp. 40–49, Osnabrück, Germany, 1983.

[21] Brejová B., Brown D.G., Vinař T.: The most probable annotation problem in HMMs and its application to bioinformatics, *Journal of Computer and System Sciences*, vol. 73, pp. 1060–1077, 2007.

[22] Brendel V., Busse H.G.: Genome structure described by formal languages, *Nucleic Acids Research*, vol. 12, pp. 2561–2568, 1984.

[23] Brown M., Wilson C.: RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. In: *Proc. 1996 Pacific Symposium on Biocomputing*, pp. 109–125, Hawaii, 1996.

[24] Brown M.P.: Small subunit ribosomal RNA modeling using stochastic context-free grammars. In: *Proc. 8th Int. Conf. on Intelligent Systems for Molecular Biology*, pp. 57–66, San Diego, CA, USA, 2000.

[25] Bunke H., Sanfeliu A. (eds.): *Syntactic and Structural Pattern Recognition - Theory and Applications*, World Scientific, Singapore, 1990.

[26] Bystroff C., Krogh A.: Hidden Markov models for prediction of protein features. In: M. Zaki, C. Bystroff (eds.), *Protein Structure Prediction. Methods in Molecular Biology*, pp. 173–198, Humana Press, New Jersey, 2008.

[27] Bystroff C., Shao Y., Yuan X.: Five hierarchical levels of sequence-structure correlation in proteins, *Applied Bioinformatics*, vol. 3, pp. 97–104, 2004.

Early bird

[28] Cai L., Malmberg R., Wu Y.: Stochastic modeling of RNA pseudoknotted structures: A grammatical approach, *Bioinformatics*, vol. 19, pp. i66–i73, 2003.

[29] Cai Y., Lux M.W., Adam L., Peccoud J.: Modeling structure-function relationships in synthetic DNA sequences using attribute grammars, *PLoS Computational Biology*, vol. 5, p. e1000529, 2009.

[30] Chiang D., Joshi A.K., Searls D.B.: Grammatical representations of macromolecular structure, *Journal of Computational Biology*, vol. 13, pp. 1077–1100, 2006.

[31] Chuong B.D., Daniel A.W., Serafim B.: CONTRAfold: RNA secondary structure prediction without physics-based models, *Bioinformatics*, vol. 22, pp. e90–e98, 2006.

[32] Collado-Vides J.: A transformational-grammar approach to the study of the regulation of gene expression, *Journal of Theoretical Biology*, vol. 136, pp. 403–425, 1989.

[33] Collado-Vides J.: A syntactic representation of units of genetic information - A syntax of units of genetic information, *Journal of Theoretical Biology*, vol. 148, pp. 401–429, 1991.

[34] Collado-Vides J.: Grammatical model of the regulation of gene expression, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, pp. 9405–9409, 1992.

[35] Coste F.: Learning the language of biological sequences. In: J. Heinz, J.M. Sempere (eds.), *Topics in Grammatical Inference*, pp. 215–247, Springer, 2016.

[36] Datta S., Mukhopadhyay S.: A composite method based on formal grammar and DNA structural features in detecting human polymerase II promoter region, *PLOS ONE*, vol. 8, p. e54843, 2013.

[37] Dill K.E., Lucas A., Hockenmaier J., Huang L., Chiang D., Joshi A.K.: Computational linguistics: A new tool for exploring biopolymer structures and statistical mechanics, *Polymer*, vol. 48, pp. 4289–4300, 2007.

[38] Ding L., Samad A., Xue X., Huang X., Malmberg R.L., Cai L.: Stochastic k-tree grammar and its application in biomolecular structure modeling, *Lecture Notes in Computer Science*, vol. 8370, pp. 308–322, 2014.

[39] Do C.B., Mahabhashyam M.S., Brudno M., Batzoglou S.: ProbCons: Probabilistic consistency-based multiple sequence alignment, *Genome Research*, vol. 15, pp. 330–340, 2005.

[40] Dong S., Searls D.B.: Gene structure prediction by linguistic methods, *Genomics*, vol. 23, pp. 540–551, 1994.

[41] Dowell R.D., Eddy S.R.: Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction, *BMC Bioinformatics*, vol. 5:71, 2004.

[42] Duda R.O., Hart P.E., Stork D.G.: *Pattern Classification*, Wiley, New York, 2001.

Early bird

[43] Durbin R., Eddy S.R., Krogh A., Mitchison G.: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, UK, 2002.

[44] Dyrka W., Nebel J.C.: A stochastic context free grammar based framework for analysis of protein sequences, *BMC Bioinformatics*, vol. 10:323, 2009.

[45] Dyrka W., Nebel J.C., Kotulska M.: Probabilistic grammatical model of protein language and its application to helix-helix contact site classification, *Algorithms for Molecular Biology*, vol. 8:31, 2013.

[46] Eddy S.R.: Profile hidden Markov models, *Bioinformatics*, vol. 14, pp. 755–763, 1998.

[47] Eddy S.R.: What is a hidden Markov model?, *Nature Biotechnology*, vol. 22, pp. 1315–1316, 2004.

[48] Eggers D., zu Siederdissen Höner H.C., Stadler P.F.: Accuracy of RNA structure prediction depends on the pseudoknot grammar, *Lecture Notes in Computer Science*, vol. 13523, pp. 20–31, 2022.

[49] Ehrig H., Ehrig K., Prange U., Taentzer G.: *Fundamentals of Algebraic Graph Transformation*, Springer, Berlin-Heidelberg, 2006.

[50] Ehrig H., Engels G., Kreowski H.J., Rozenberg G. (eds.): *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools*, World Scientific, Singapore, 1999.

[51] Ehrig H., Kreowski H.J.: Pushout-properties: An analysis of gluing constructions for graphs, *Mathematische Nachrichten*, vol. 91, pp. 135–149, 1979.

[52] Ehrig H., Pfender M., Schneider H.J.: Graph grammars: An algebraic approach. In: *Proc. 14th Annual IEEE Symposium on Switching and Automata Theory*, pp. 167–180, USA, 1973.

[53] Flasiński M.: Parsing of edNLC-graph grammars for scene analysis, *Pattern Recognition*, vol. 21, pp. 623–629, 1988.

[54] Flasiński M.: Distorted pattern analysis with the help of Node Label Controlled graph languages, *Pattern Recognition*, vol. 23, pp. 765–774, 1990.

[55] Flasiński M.: On the parsing of deterministic graph languages for syntactic pattern recognition, *Pattern Recognition*, vol. 26, pp. 1–16, 1993.

[56] Flasiński M.: Use of graph grammars for the description of mechanical parts, *Computer-Aided Design*, vol. 27, pp. 403–433, 1995.

[57] Flasiński M.: Power properties of NLC graph grammars with a polynomial membership problem, *Theoretical Computer Science*, vol. 201, pp. 189–231, 1998.

[58] Flasiński M.: Inference of parsable graph grammars for syntactic pattern recognition, *Fundamenta Informaticae*, vol. 80, pp. 379–413, 2007.

[59] Flasiński M.: *Introduction to Artificial Intelligence*, Springer International, Switzerland, 2016.

[60] Flasiński M.: *Syntactic Pattern Recognition*, World Scientific, New Jersey-London-Singapore, 2019.

Early bird

[61] Flasiński M., Jurek J.: Dynamically programmed automata for quasi context sensitive languages as a tool for inference support in pattern recognition-based real-time control expert systems, *Pattern Recognition*, vol. 32, pp. 671–690, 1999.

[62] Flasiński M., Jurek J., Peszek T.: Multi-derivational parsing of vague languages - the new paradigm of syntactic pattern recognition, *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 46, 2024.

[63] Flasiński M., Kotulski L.: On the use of graph grammars for the control of a distributed software allocation, *The Computer Journal*, vol. 35, pp. A165–A175, 1992.

[64] Flasiński M., Lewicki G.: The convergent method of constructing polynomial discriminant functions for pattern recognition, *Pattern Recognition*, vol. 24, pp. 1009–1015, 1991.

[65] Flasiński M., Myśliński S.: On the use of graph parsing for recognition of isolated hand postures of Polish Sign Language, *Pattern Recognition*, vol. 43, pp. 2249–2264, 2010.

[66] Fonzo de V., Aluffi-Pentini F., Parisi V.: Hidden Markov models in bioinformatics, *Current Bioinformatics*, vol. 2, pp. 49–61, 2007.

[67] Fu K.S.: Stochastic automata, stochastic languages and pattern recognition, *Journal of Cybernetics*, vol. 1, pp. 31–49, 1971.

[68] Fu K.S.: Stochastic tree languages and their application to picture processing. In: P.R. Krishnaiah (ed.), *Multivariate Analysis V*, North-Holland, Amsterdam, 1980.

[69] Fu K.S.: *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs, 1982.

[70] Fu K.S., Huang T.: Stochastic grammars and languages, *International Journal of Computer and Information Sciences*, vol. 1, pp. 135–170, 1972.

[71] Fu K.S., Li T.: On stochastic automata and languages, *Information Sciences*, vol. 1, pp. 403–419, 1969.

[72] Gallego A.J., López D., Calera-Rubio J.: Grammatical inference of directed acyclic graph languages with polynomial time complexity, *Journal of Computer and System Sciences*, vol. 95, pp. 19–34, 2018.

[73] Gécseg F., Steinby M.: *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.

[74] Ghouila A., etal: Identification of divergent protein domains by combining HMM-HMM comparisons and co-occurrence detection, *PLoS ONE*, vol. 9, p. e95275, 2014.

[75] Giegerich R.: *A Declarative approach to the development of dynamic programming algorithms, applied to RNA folding*, Tech. rep., Bielefeld University, Bielefeld, Germany, 1998.

[76] Giegerich R.: Explaining and controlling ambiguity in dynamic programming, *Lecture Notes in Computer Science*, vol. 1848, pp. 46–59, 2000.

[77] Giegerich R.: Systematic approach to dynamic programming in bioinformatics, *Bioinformatics*, vol. 16, pp. 665–667, 2000.

Early bird

[78] Giegerich R., Meyer C.: Algebraic Dynamic Programming, *Lecture Notes in Computer Science*, vol. 2422, pp. 349–364, 2002.

[79] Giegerich R., Meyer C., Steffen P.: Towards a discipline of dynamic programming, *Lecture Notes in Informatics*, vol. P-147, pp. 3–44, 2002.

[80] Giegerich R., Meyer C., Steffen P.: A discipline of dynamic programming over sequence data, *Science of Computer Programming*, vol. 51, pp. 215–263, 2004.

[81] Giegerich R., Touzet H.: Modeling dynamic programming problems over sequences and trees with inverse coupled rewrite systems, *Algorithms*, vol. 7, pp. 62–144, 2014.

[82] Golab T., Ledley R.S., Rotolo L.S.: FIDAC: Film input to digital automatic computer, *Pattern Recognition*, vol. 3, pp. 123–156, 1971.

[83] Gollery M. (ed.): *Handbook of Hidden Markov Models in Bioinformatics*, Chapman and Hall / CRC, Boca Raton, FL, 2008.

[84] Gorn S.: Explicit definitions and linguistics dominoes. In: J.F. Hart, S. Takasu (eds.), *Systems and Computer Science*, University of Toronto Press, Toronto, 1967.

[85] Grenander U.: *Syntax-controlled probabilities*, Tech. rep., Brown University, Providence, R.I., 1967.

[86] Harmanci A.O., Sharma G., Mathews D.H.: Efficient pairwise RNA structure prediction using probabilistic alignment constraints in *Dynalign*, *BMC Bioinformatics*, vol. 8:357, 2007.

[87] Haussler D., Krogh A., Mian I.S., Sjöander K.: Protein modeling using hidden Markov models: analysis of globins. In: *Proc. 26th Ann. Hawaii Int. Conf. Systems Sciences*, pp. 792–802, Hawaii, 1993.

[88] Head T.: Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bulletin of Mathematical Biology*, vol. 49, pp. 737–759, 1987.

[89] Holmes I., Rubin G.M.: Pairwise RNA structure comparison with stochastic context-free grammars. In: *Proc. 2002 Pacific Symposium on Biocomputing*, pp. 163–174, Hawaii, 2002.

[90] Horan K., Shelton C., Girke T.: Predicting conserved protein motifs with sub-HMMs, *BMC Bioinformatics*, vol. 11, p. 205, 2010.

[91] Hsu B.Y., Wong T.K.F., Hon W.K., Liu X., Lam T.W., Yiu S.M.: A local structural prediction algorithm for RNA triple helix structure, *Lecture Notes in Computer Science*, vol. 7968, pp. 102–113, 2013.

[92] Huang T., Fu K.S.: Stochastic syntactic analysis for programmed grammars and syntactic pattern recognition, *Computer Graphics and Image Processing*, vol. 1, pp. 257–283, 1972.

[93] Janssen S., Giegerich R.: Faster computation of exact RNA shape probabilities, *Bioinformatics*, vol. 26, pp. 632–639, 2010.

[94] Janssens D., Rozenberg G.: On the structure of node-label-controlled graph languages, *Information Sciences*, vol. 20, pp. 191–216, 1980.

Early bird

[95] Janssens D., Rozenberg G.: Restrictions, extensions, and variations of NLC grammars, *Information Sciences*, vol. 20, pp. 217–244, 1980.

[96] Janssens D., Rozenberg G.: Graph grammars with neighbourhood-controlled embedding, *Theoretical Computer Science*, vol. 21, pp. 55–74, 1982.

[97] Janssens D., Rozenberg G., Verraedt R.: On sequential and parallel node-rewriting graph grammars, *Computer Graphics and Image Processing*, vol. 18, pp. 279–304, 1982.

[98] Johnson L., Eddy S., Portugaly L.: Hidden Markov model speed heuristic and iterative HMM search procedure, *BMC Bioinformatics*, vol. 11, p. 431, 2010.

[99] Jonyer I., Holder L.B., Cook D.J.: MDL-based context free graph grammar induction and applications, *International Journal on Artificial Intelligence Tools*, vol. 13, pp. 65–79, 2004.

[100] Joshi A.K.: How much context-sensitivity is necessary for characterizing structural descriptions – Tree Adjoining Grammars. In: D. Dowty et al. (ed.), *Natural Language Processing – Theoretical, Computational and Psychological Perspective*, Cambridge University Press, New York, NY, 1985.

[101] Joshi A.K., Levy L.S., Takahashi M.: Tree adjunct grammars, *Journal of Computer and System Sciences*, vol. 10, pp. 136–163, 1975.

[102] Joshi A.K., Schabes Y.: Tree adjoining grammars. In: G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages - III*, pp. 69–123, Springer, New York, NY, 1997.

[103] Käll L., Krogh A., Sonnhammer E.: An HMM posterior decoder for sequence feature prediction that includes homology information, *Bioinformatics*, vol. 21, pp. i251–i257, 2005.

[104] Karplus K., Barrett C., Hughey R.: Hidden Markov models for detecting remote protein homologies, *Bioinformatics*, vol. 14, pp. 846–856, 1998.

[105] Kato Y., Akutsu T., Seki H.: A grammatical approach to RNA-RNA interaction prediction, *Pattern Recognition*, vol. 42, pp. 531–538, 2009.

[106] Kato Y., Seki H., Kasami T.: Subclasses of tree adjoining grammars for RNA secondary structure. In: *Proc. 7th Int. Workshop on Tree Adjoining Grammar and Related Formalisms*, pp. 48–55, Vancouver, Canada, 2004.

[107] Kato Y., Seki H., Kasami T.: Stochastic multiple context-free grammar for RNA pseudoknot modeling. In: *Proc. 8th Int. Workshop on Tree Adjoining Grammar and Related Formalisms*, pp. 57–64, Sydney, Australia, 2006.

[108] Kennedy P.J., Osborn T.R.: A model of gene expression and regulation in an artificial cellular organism, *Complex Systems*, vol. 13, pp. 33–59, 2001.

[109] Kirsch R.A.: Computer determination of the constituent structure of bio1ogical images, *Computers and Biomedical Research*, vol. 4, pp. 315–328, 1971.

[110] Knudsen B., Heyn J.: RNA secondary structure prediction using stochastic context-free grammars and evolutionary history, *Bioinformatics*, vol. 15, pp. 446–454, 1999.

[111] Knudsen B., Heyn J.: Pfold: RNA secondary structure prediction using stochastic context-free grammars, *Nucleic Acids Research*, vol. 31, pp. 3423–3428, 2003.

Early bird

[112] Knudsen B., Miyamoto M.M.: Sequence alignments and pair hidden Markov models using evolutionary history, *Journal of Molecular Biology*, vol. 333, pp. 453–460, 2003.

[113] Knuth D.: Semantics of context-free languages, *Mathematical Systems Theory*, vol. 2, pp. 127–145, 1968.

[114] Koutroumbas K., Theodoridis S.: *Pattern Recognition*, Academic Press, Boston, 2008.

[115] Krogh A., Brown M., Mian I.S., Sjöander K., Haussler D.: Hidden Markov models in computational biology: Applications to protein modeling, *Journal of Molecular Biology*, vol. 235, pp. 1501–1531, 1994.

[116] Krogh A., Larsson B., Heijne von G., Sonnhammer E.: Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes, *Journal of Molecular Biology*, vol. 305, pp. 567–580, 2001.

[117] Krogh A., Mian I.S., Haussler D.: A hidden Markov model that finds genes in E.coli DNA, *Nucleic Acids Research*, vol. 22, pp. 4768–4778, 1994.

[118] Kulp D., Haussler D., Reese M.G., Eeckman F.H.: A generalized hidden Markov model for the recognition of human genes in DNA. In: *Proc. 4th Int. Conf. on Intelligent Systems for Molecular Biology*, pp. 134–142, St. Louis, MO, USA, 1996.

[119] Lasfar M., Bouden H.: A method of data mining using hidden Markov models (HMMs) for protein secondary structure prediction, *Procedia Computer Science*, vol. 127, pp. 42–51, 2018.

[120] Ledley R.S.: High-speed automatic analysis of biomedical pictures, *Science*, vol. 146, pp. 216–223, 1964.

[121] Ledley R.S., Rotolo L.S., Golab T.J., Jacobsen J.D., Ginsberg M.D., Wilson J.B.: FIDAC: Film input to digital automatic computer and associated syntax-directed pattern-recognition programming system. In: J.T. Tippet, D. Beckovitz, L. Clapp, C. Koester, A. Vanderburgh Jr. (eds.), *Optical and Electro-optical Information Processing*, pp. 591–613, MIT Press, Cambridge, MA, 1965.

[122] Lee H.C., Fu K.S.: A stochastic syntax analysis procedure and its application to pattern classification, *IEEE Trans Computers*, vol. 21, pp. 660–666, 1972.

[123] Lefebvre F.: A grammar-based unification of several alignment and folding algorithms. In: *Proc. 4th Int. Conf. on Intelligent Systems for Molecular Biology*, pp. 143–154, St. Louis, MO, USA, 1996.

[124] Leung S.W., Mellish C., Robertson D.: Basic Gene Grammars and DNA-Chart Parser for language processing of *Escherichia coli* promoter DNA sequences, *Bioinformatics*, vol. 17, pp. 226–236, 2001.

[125] Levenshtein V.I.: Binary codes capable of correcting deletions, insertions and reversals, *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1966.

[126] Li J., Lee J., Liao L.: A new algorithm to train hidden Markov models for biological sequences with partial labels, *BMC Bioinformatics*, vol. 22, p. 162, 2021.

Early bird

[127] Li M., etal: Predicting RNA secondary structures: One-grammar-fits-all solution, *Lecture Notes in Computer Science*, vol. 9096, pp. 211–222, 2015.

[128] Liang K.C., Wang X., Anastassiou D.: Bayesian basecalling for DNA sequence analysis using hidden Markov models, *IEEE/ACM Trans Computational Biology and Bioinformatics*, vol. 4, pp. 430–440, 2007.

[129] Liu L., etal: Euler string-based compression of tree-structured data and its application to analysis of RNAs, *Current Bioinformatics*, vol. 13, pp. 25–33, 2018.

[130] Lobo D., Vico F.J., Dassow J.: Graph grammars with string-regulated rewriting, *Theoretical Computer Science*, vol. 412, pp. 6101–6111, 2011.

[131] Lottaz C., Iseli C., Jongeneel C.V., Bucher P.: Modeling sequencing errors by combining hidden Markov models, *Bioinformatics*, vol. 19 (Suppl. 2), pp. i103–i112, 2003.

[132] Lu S.Y., Fu K.S.: Structure-preserved error-correcting tree automata for syntactic pattern recognition. In: *Proc. IEEE Conf. on Decision and Control*, pp. 413–419, Clearwater, FL, USA, 1976.

[133] Lyngso R.B., Pedersen C.N.: RNA pseudoknot prediction in energy-based models, *Journal of Computational Biology*, vol. 7, pp. 409–427, 2000.

[134] Majoros W.H., Pertea M., Delcher A.L., Salzberg S.L.: Efficient decoding algorithms for generalized hidden Markov model gene finders, *BMC Bioinformatics*, vol. 6:16, 2005.

[135] Mamitsuka H., Abe N.: Predicting location and structure of beta-sheet regions using stochastic tree grammars. In: *Proc. 2nd Int. Conf. on Intelligent Systems for Molecular Biology*, pp. 276–284, Stanford, CA, USA, 1994.

[136] Mamuye A., Merelli E., Tesei L.: A graph grammar for modelling RNA folding. In: *Proc. 2nd Graphs as Models Workshop*, pp. 31–41, Eindhoven, The Netherlands, 2016.

[137] Marchand B., etal: Automated design of dynamic programming schemes for RNA folding with pseudoknots. In: *Proc. 22nd Int. Workshop on Algorithms in Bioinformatics*, pp. 7:1–7:24, Potsdam, Germany, 2022.

[138] Markov A.A.: Essai d'une recherche statistique sur le texte du roman *Eugene Onegin* illustrant la liaison des epreuve en chain, *Bulletin de l'Académie Impériale des Sciences de St-Pétersbourg*, vol. 7, pp. 153–162, 1913.

[139] Matsui H., Sato K., Sakakibara Y.: Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures, *Bioinformatics*, vol. 21, pp. 2611–2617, 2005.

[140] Menichelli C., Gascuel O., Bréhélin L.: Improving pairwise comparison of protein sequences with domain co-occurrence, *PLoS Computational Biology*, vol. 14, p. e1005889, 2018.

[141] Muggleton S., Bryant C., Srinivasan A., Whittaker A., Topp S., Rawlings C.: Are grammatical representations useful for learning from biological sequence data?, *Journal of Computational Biology*, vol. 8, pp. 493–522, 2001.

Early bird

[142] Munch K., Krogh A.: Automatic generation of gene finders for eukaryotic species, *BMC Bioinformatics*, vol. 7:263, 2006.

[143] Nebel M.E., Weinberg F.: Algebraic and combinatorial properties of common RNA pseudoknot classes with applications, *Journal of Computational Biology*, vol. 19, pp. 1134–1150, 2012.

[144] Pachter L., Alexandersson M., Cawley S.: Applications of generalized pair hidden Markov models to alignment and gene finding problems, *Journal of Computational Biology*, vol. 9, pp. 389–399, 2002.

[145] Pavlidis T.: Structural descriptions and graph grammars. In: S.K. Chang, K.S. Fu (eds.), *Pictorial Information Systems*, pp. 86–103, Springer, Berlin - Heidelberg - New York, 1980.

[146] Pedersen J.C., Hein J.: Gene finding with a hidden Markov model of genome structure and evolution, *Bioinformatics*, vol. 19, pp. 219–227, 2003.

[147] Peris P., López D., Campos M.: IgTM: An algorithm to predict transmembrane domains and topology in proteins, *BMC Bioinformatics*, vol. 9:367, 2008.

[148] Plötz T., Fink G.: Pattern recognition methods for advanced stochastic protein sequence analysis using HMMs, *Pattern Recognition*, vol. 39, pp. 2267–2280, 2006.

[149] Ponty Y.: *Ensemble Algorithms and Analytic Combinatorics in RNA Bioinformatics and Beyond*, Tech. rep., Universite Paris-Saclay, Paris, France, 2020.

[150] Porter T., Hajibabaei M.: Profile hidden Markov model sequence analysis can help remove putative pseudogenes from DNA barcoding and metabarcoding datasets, *BMC Bioinformatics*, vol. 22, p. 256, 2021.

[151] Przytycka T., Srinivasan R., Rose G.D.: Recursive domains in proteins, *Protein Science*, vol. 11, pp. 409–417, 2002.

[152] Quadrini M., Tesei L., Merelli E.: An algebraic language for RNA pseudoknots comparison, *BMC Bioinformatics*, vol. 20, p. p1, 2019.

[153] Rabin M.O.: Probabilistic automata, *Information and Control*, vol. 6, pp. 230–245, 1963.

[154] Reese M.G., Kulp D., Tammana H., Haussler D.: *Genie* - gene finding in *Drosophila melanogaster*, *Genome Research*, vol. 10, pp. 529–538, 2000.

[155] Riechert M., zu Siederdissen Höner H.C., Stadler P.F.: Algebraic dynamic programming for multiple context-free grammars, *Theoretical Computer Science*, vol. 639, pp. 91–109, 2016.

[156] Ripley B.D.: *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge, 2008.

[157] Rivas E., Eddy S.R.: The language of RNA: a formal grammar that includes pseudoknots, *Bioinformatics*, vol. 16, pp. 334–340, 2000.

[158] Rivas E., Eddy S.R.: Noncoding RNA gene detection using comparative sequence analysis, *BMC Bioinformatics*, vol. 2:8, 2001.

[159] Rosenblueth D.A., Thieffry D., Huerta A.M., Salgado H., Collado-Vides J.: Syntactic recognition of regulatory regions in *Escherichia coli*, *Computer Applications in the Biosciences*, vol. 12, pp. 15–22, 1996.

Early bird

[160] Rosenkrantz D.J.: Programmed grammars and classes of formal languages, *Journal of the Association for Computing Machinery*, vol. 16, pp. 107–131, 1969.
[161] Sakakibara Y.: Grammatical inference in bioinformatics, *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1051–1062, 2005.
[162] Sakakibara Y., etal: Stochastic context-free grammars for tRNA modeling, *Nuclear Acids Research*, vol. 22, pp. 5112–5120, 1994.
[163] Salomaa A.: Probabilistic and weighted grammars, *Information and Control*, vol. 15, pp. 529–544, 1969.
[164] Sanchez-Graillet O., Poesio M.: Negation of proteinprotein interactions: analysis and extraction, *Bioinformatics*, vol. 23, pp. i424–i432, 2007.
[165] Sato K., Hamada M.: Recent trends in RNA informatics: a review of machine learning and deep learning for RNA secondary structure prediction and RNA drug discovery, *Briefings in Bioinformatics*, vol. 24, pp. 1–13, 2023.
[166] Sauthoff G., Giegerich R.: Yield grammar analysis and product optimization in a domain-specific language for dynamic programming, *Science of Computer Programming*, vol. 87, pp. 2–22, 2014.
[167] Schalkoff R.: *Pattern Recognition: Statistical, Structural and Neural Approaches*, Wiley, New York, 2005.
[168] Searls D.B.: The linguistics of DNA, *American Scientist*, vol. 80, pp. 579–591, 1992.
[169] Searls D.B.: The computational linguistics of biological sequences. In: L. Hunter (ed.), *Artificial Intelligence and Molecular Biology*, pp. 47–120, AAAI/MIT Press, Menlo Park, CA, 1993.
[170] Searls D.B.: String Variable Grammar: a logic grammar formalism for DNA sequences, *The Journal of Logic Programming*, vol. 24, pp. 73–102, 1995.
[171] Searls D.B.: Linguistic approaches to biological sequences, *Computer Applications in the Biosciences*, vol. 13, pp. 333–344, 1997.
[172] Searls D.B.: Reading the book of life, *Bioinformatics*, vol. 17, pp. 579–580, 2001.
[173] Searls D.B.: The language of genes, *Nature*, vol. 420, pp. 211–217, 2002.
[174] Seesi al S., Rajasekaran S., Ammar R.: Pseudoknot Identification through Learning TAG(RNA), *Lecture Notes in Computer Science*, vol. 5265, pp. 132–143, 2008.
[175] Seki H., Matsumura T., Fujii M., Kasami T.: On multiple context-free grammars, *Theoretical Computer Science*, vol. 88, pp. 191–229, 1991.
[176] Seoud R.A.A., M. Y.A.B., Kadah Y.M.: Extraction of protein interaction information from unstructured text using a link grammar parser. In: *Proc. 2007 International Conference on Computer Engineering and Systems*, pp. 70–75, Cairo, Egypt, 2007.
[177] Shen X., Vikalo H.: ParticleCall: A particle filter for base calling in next-generation sequencing systems, *BMC Bioinformatics*, vol. 13:160, 2012.
[178] zu Siederdissen Höner H.C., Hofacker I.L., Stadler P.F.: Product grammars for alignment and folding, *IEEE/ACM Trans Computational Biology and Bioinformatics*, vol. 12, pp. 507–519, 2015.

Early bird

[179] Silva da W.M.C., Andersen J.L., Holanda M.T., Walter M.E.M.T., Brigido M.M., Stadler P.F., Flamm C.: Exploring plant sesquiterpene diversity by generating chemical networks, *Processes*, vol. 7, p. 240, 2019.

[180] Singh P., Bandyopadhyay P., Bhattacharya S., Krishnamachari A., Sengupta S.: Riboswitch detection using profile hidden Markov models, *BMC Bioinformatics*, vol. 10:325, 2009.

[181] Slisenko A.O.: Context-free grammars as a tool for describing polynomial-time subclasses of hard problems, *Information Processing Letters*, vol. 14, pp. 52–56, 1982.

[182] Smoly I., Carmel A., Shemer-Avni Y., Yeger-Lotem E., Ziv-Ukelson M.: Algorithms for regular tree grammar network search and their application to mining human-viral infection patterns, *Journal of Computational Biology*, vol. 23, pp. 165–179, 2016.

[183] Söding J.: Protein homology detection by HMM-HMM comparison, *Bioinformatics*, vol. 21, pp. 951–960, 2005.

[184] Srivastava P.K., Desai D.K., Nandi S., Lynn A.M.: HMM-ModE-improved classification using profile hidden Markov models by optimising the discrimination threshold and modifying emission probabilities with negative training sequences, *BMC Bioinformatics*, vol. 8:104, 2007.

[185] St-Onge K., Thibault P., Hamel S., Major F.: Modeling RNA tertiary structure motifs by graph-grammars, *Nucleic Adids Research*, vol. 35, pp. 1726–1736, 2007.

[186] Sun Y., Buhler J.: Designing patterns for profile HMM search, *Bioinformatics*, vol. 23, pp. e36–e43, 2006.

[187] Sun Y., Buhler J.: Designing patterns and profiles for faster HMM search, *IEEE/ACM Trans Computational Biology and Bioinformatics*, vol. 6, pp. 232–243, 2009.

[188] Tamposis I., etal: Semi-supervised learning of hidden Markov models for biological sequence analysis, *Bioinformatics*, vol. 35, pp. 2208–2215, 2019.

[189] Tanaka E., Ikeda M., Ezure K.: Direct parsing, *Pattern Recognition*, vol. 19, pp. 315–323, 1986.

[190] Temkin J.M., Gilder M.R.: Extraction of protein interaction information from unstructured text using a context-free grammar, *Bioinformatics*, vol. 19, pp. 2046–2053, 2003.

[191] Terrapon N., Gascuel O., Maréchal E., Bréhélin L.: Fitting hidden Markov models of protein domains to a target species: application to Plasmodium falciparum, *BMC Bioinformatics*, vol. 13, p. 67, 2012.

[192] Thomason M.G., Gonzales R.C.: Syntactic recognition of imperfectly specified patterns, *IEEE Trans Computers*, vol. 24, pp. 93–95, 1975.

[193] Tsafnat G., Schaeffer J., Clayphan A., Iredell J.R., Partridge S.R., Coiera E.: Computational inference of grammars for larger-than-gene structures from annotated gene sequences, *Bioinformatics*, vol. 27, pp. 791–796, 2011.

Early bird

[194] Turakainen P.: On stochastic languages, *Information and Control*, vol. 12, pp. 304–313, 1968.

[195] Turan G.: *On the complexity of graph grammars*, Rep. Automata Theory Research Group, Szeged, Hungary, 1982.

[196] Uemura Y., Hasegawa A., Kobayashi S., Yokomori T.: Tree adjoining grammars for RNA structure prediction, *Theoretical Computer Science*, vol. 210, pp. 277–303, 1999.

[197] Vijayakumar J., Mathew L., Nagar A.K.: A New class of graph grammars and modelling of certain biological structures, *Symmetry*, vol. 15, p. 349, 2023.

[198] Wang J., Keightley P.D., Johnson T.: MCALIGN2: faster, accurate global pairwise alignment of non-coding DNA sequences based on explicit models of indel evolution, *BMC Bioinformatics*, vol. 7:292, 2006.

[199] Weinberg Z., Ruzzo W.L.: Sequence-based heuristics for faster annotation of non-coding RNA families, *Bioinformatics*, vol. 22, pp. 35–39, 2006.

[200] Wieczorek W., Unold O.: Use of a novel grammatical inference approach in classification of amyloidogenic hexapeptides, *Computational and Mathematical Methods in Medicine*, vol. 2016:1782732, 2016.

[201] Wistrand M., Sonnhammer E.L.: Improving profile HMM discrimination by adapting transition probabilities, *Journal of Molecular Biology*, vol. 338, pp. 847–854, 2004.

[202] Won K.J., Hamelryck T., Prügel-Bennett A., Krogh A.: An evolutionary method for learning HMM structure: prediction of protein secondary structure, *BMC Bioinformatics*, vol. 8:357, 2007.

[203] Yandell M.D., Majoros W.H.: Genomics and natural language processing, *Nature Reviews Genetics*, vol. 3, pp. 601–610, 2002.

[204] Yokomori T., Kobayashi S.: Learning local languages and their application to DNA sequence analysis, *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1067–1079, 1998.

[205] Yoon B.J.: Hidden Markov models and their applications in biological sequence analysis, *Current Genomics*, vol. 10, pp. 402–415, 2009.

[206] Yoon B.J., Vaidyanathan P.P.: Structural alignment of RNAs using profile-csHMMs and its application to RNA homology search: Overview and new results, *IEEE Trans Automatic Control*, vol. 53, pp. 10–25, 2008.

[207] Zadeh L.A.: Note on fuzzy languages, *Information Sciences*, vol. 1, pp. 421–434, 1969.

[208] Zehnder T., Benner P., Vingron M.: Predicting enhancers in mammalian genomes using supervised hidden Markov models, *BMC Bioinformatics*, vol. 20, p. 157, 2019.

[209] Zhang S., Borovok I., Aharonowitz Y., Sharan R., Bafna V.: A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements, *Bioinformatics*, vol. 22, pp. e557–e565, 2006.

Early bird

[210] Zhao Y., etal: Grammar-based compression approach to extraction of common rules among multiple trees of glycans and RNAs, *BMC Bioinformatics*, vol. 16, p. 128, 2015.

[211] Zhao Y., Hayashida M., Akutsu T.: Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures, *BMC Bioinformatics*, vol. 11, p. 154, 2010.

[212] Zou L., Wang Z., Wang Y., Hu F.: Combined prediction of transmembrane topology and signal peptide of beta-barrel proteins: Using a hidden Markov model and genetic algorithms, *Computers in Biology and Medicine*, vol. 40, pp. 621–628, 2010.

Affiliations

Mariusz Flasiński
Jagiellonian University, Information Technology Systems Department, Cracow 30-348, ul. prof. St. Łojasiewicza 4, Poland, e-mail: mariusz.flasinski@uj.edu.pl

Received: 10.03.2024
Revised: 10.03.2024
Accepted: 10.03.2024

Early bird