Pattern Analysis and Applications (2023) 26:605–615
https://doi.org/10.1007/s10044-022-01120-0

THEORETICAL ADVANCES

On the learning of vague languages for syntactic pattern recognition

Mariusz Flasiński[1] · **Janusz Jurek**[1] · **Tomasz Peszek**[1]

Received: 13 April 2021 / Accepted: 17 October 2022 / Published online: 23 November 2022

Abstract
The method of the learning of vague languages which represent distorted/ambiguous patterns is proposed in the paper. The goal of the method is to infer the quasi-context-sensitive string grammar which is used in our model as the generator of patterns. The method is an important component of the multi-derivational model of the parsing of vague languages used for syntactic pattern recognition.

Keywords Syntactic pattern recognition · Formal grammars · Grammatical induction · Vague languages

1 Introduction

Pattern recognition methods are included in two main approaches: the decision-theoretic approach [7, 10] and the syntactic-structural approach. The latter contains three groups of methods: the algebraic group [36], the structural group [8] and the syntactic group, called syntactic pattern recognition, SPR [20, 22, 24]. The problem of the self-learning of SPR systems is one of the basic open problems in syntactic pattern recognition [20, 22]. The learning modules in SPR systems are constructed according to the theory of language induction (grammatical inference). Although the first language induction algorithms were proposed in the late 1960s, the results in this research area seem to be still not satisfactory from the point of view of their practical applications [22]. Therefore, the requirement of the availability of a grammatical inference algorithm for any model of syntactic pattern recognition has been formulated as the condition *sine qua non* of the effectiveness of the model [22].

Among the variety of syntactic pattern recognition applications, signal analysis seems to be especially popular since the early 1970s [9]. These applications include electrocardiography, electroencephalography, pulse wave analysis, auditory brainstem response audiometry, cardiotocography, technical analysis in economics, information flow management, industrial signal processing, signal analysis in seismology, etc. [22, 24, 29, 39]. The syntactic pattern recognition method, based on the class of DPLL(k) context-free grammars which are able to generate a considerable subclass of context-sensitive languages, was proposed in [17, 30]. The method was applied for process monitoring and control in particle physics [5], auditory brainstem response audiometry [18], and fetal palates diagnosis [33].

Various pattern recognition and computer science methods have been widely used for the short-term electrical load, STEL, and prediction recently. The most recent publications in this application area include the following papers. Amjady and Keynia used neural networks and evolutionary algorithms in the context of price forecasting of electricity markets in [2]. Fan and Hyndman presented in [12] a STEL forecasting method for the National Electricity Market of Australia which is based on additive regression. Yang, Wu, Chen and Li proposed a hybrid model which consists of neural networks and autoregressive integrated moving average in [47]. Hong and Wang developed a method based on fuzzy interaction regression for applications in the areas of: operations, maintenance, demand response and energy market activities in [28]. Wang, Liu and Hong proposed a big data approach using multiple linear regression for STEL prediction with recency effects in [45]. A hybrid method of STEL forecasting in Philippines based on hidden Markov model and autoregressive integrated moving average was presented by Hermias, Teknomo and Monje in [25]. Tian, Ma, Zhang and Zhan developed a STEL prediction model based on long short-term memory neural networks and convolutional neural networks [44]. The variety of methods in

✉ Mariusz Flasiński
mariusz.flasinski@uj.edu.pl

[1] Information Technology Systems Department, Jagiellonian University, ul. prof. St. Łojasiewicza 4, 30-348 Cracow, Poland

this area include are surveyed in [1, 3, 4, 6, 11, 26, 27, 38, 43, 46].

In our previous paper [21], we have used the syntactic pattern recognition approach for short-term electrical load prediction. In the application of STEL prediction, the method had to be extended considerably in order to process ambiguous patterns. It was made by the introducing the class of vague DPLL(k) languages and the constructing of the *Syntactic Pattern Recognition-based Electrical Load Prediction, SPRELP,* system on the basis of this class [21]. Since new signal patterns occur very often in this application area, the requirement of the self-learning of the *SPRELP* system has turned out to be crucial for the effectiveness of the system. The purpose of the research presented in the paper has been to develop a holistic method of syntactic pattern learning that includes both the learning of self-organizing maps, SOMs (used for the generation of a structural pattern representation) and the learning in the parsing system for DPLL(k) grammars [21] (used for the structural pattern recognition).

The generic scheme of our hybrid model is presented in Sect. 2. The concept of the use of self-organizing maps for the generating of vague patterns which belong to a DPLL(k) language is introduced in Sect. 3. The algorithm of the language induction for the DPLL(k) class is discussed in Sect. 4. The application of the *SPRELP* system for short-term electrical load prediction is described in Sect. 5. The concluding remarks are contained in the final section.

2 Syntactic pattern recognition with vague languages

The solving of the following two fundamental open problems seems to be crucial for the construction of an effective syntactic/structural pattern recognition system:

- the generation of a structural pattern representation in case of distorted/ambiguous objects (including the development of stochastic models [24, 40]) and
- the learning of SPR systems (including the use of grammatical induction algorithms [20], learning automata [48] etc.).

For the solving of the first problem, the two-phase recognition model was proposed in [21]. In order to avoid the loss of information which represents the vagueness/ambiguity of objects or processes to be recognized, a model based on the so-called *vague structural patterns* has been defined.

The generic scheme of such a model/system is shown in Fig. 1. Firstly, let us present the recognition phase in the system. In the first step, a vague structural pattern is generated on the basis of an input feature vector. Such a pattern is defined with the help of vague primitives. A vague primitive allows us to describe the fuzzy nature of objects/processes to be recognized. Let us introduce it formally in the following way.

Definition 1 Let G be a context-free grammar. Let Σ_T be a t-element set of terminal symbols of G. A *vague primitive* is a vector $\mathbf{v} = (a_{k_1}\mu_{k_1}, a_{k_2}\mu_{k_2}, \ldots, a_{k_j}\mu_{k_j})$, $1 \le j \le t$, where: $a_{k_1}, a_{k_2}, \ldots, a_{k_j} \in \Sigma_T$ are different symbols, $\mu_{k_1}, \mu_{k_2}, \ldots, \mu_{k_j}$ are the measures, called *attributes*, which are ascribed to $a_{k_1}, a_{k_2}, \ldots, a_{k_j}$, correspondingly. □

In our approach, the attributes of vague primitives can be of the form of distance, probability, fuzziness measures, etc. Let us consider the following example. Let a set of structural primitives be defined as it is shown in Fig. 2a. The exemplary vague primitive is depicted in Fig. 2b, where attributes are assigned to its component primitives according to the Euclidean metric. We assume that this metric is used for the

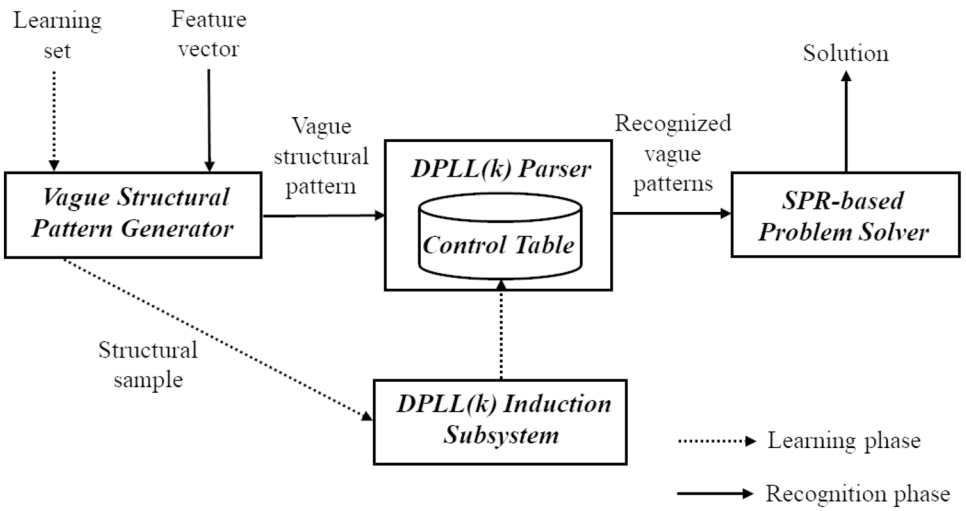

Fig. 1 The generic scheme of the vague DPLL(k) parsing-based recognition system

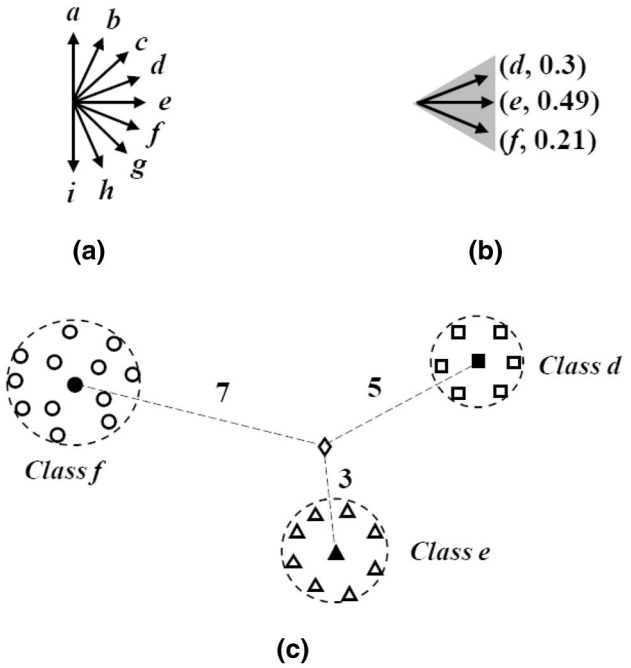

Fig. 2 **a** A set of structural primitives, **b** a vague primitive, **c** the example of the defining of attributes of the vague primitive by the SOM classifier

attributing of vague patterns in the examples presented in the paper. The first step, i.e., the generation of vague structural patterns, will be discussed in a more detailed way in the next section.

In the second step, the vague structural pattern is recognized by the DPLL(k) syntax analyzer (parser). The syntax analyzer determines b best vague patterns. These patterns are treated as acceptable approximations of the (idealized) template. This is the fundamental difference between our approach and standard syntactic pattern recognition approaches in which the single "correct" structural pattern is determined during parsing. The quality of these b best patterns is defined with the help of the attributes of their component vague primitives. Finally, the quality measures are used for the obtaining of the "averaged" structural pattern by the problem solver. The recognition path, presented briefly here, was described in [21] in a detailed way.

In the syntactic pattern recognition paradigm, the availability of a structural sample set is assumed for the purpose of the self-learning of a pattern recognition system. (A sample set in syntactic pattern recognition corresponds to a learning (training) set in the decision-theoretic approach.) This sample set consists of structural patterns, and it represents the formal language underlying. A sample set should be big enough to represent a variety of structural patterns. Therefore, the constructing of the formal grammar or the control table of the syntax analyzer (parser) "by hand" is impossible. Fortunately, grammatical induction algorithms can be used in order to automate this process [19, 22]. The availability of a grammatical induction algorithm for the class of grammars used in the model of syntactic pattern recognition proposed is so important that it has been formulated as the fundamental methodological principle in [20]:

A syntactic pattern recognition model should include the following three components: a generative grammar, a computationally efficient parsing algorithm and a grammatical induction algorithm of the polynomial time complexity.

Since we propose the two-step syntactic pattern recognition model, the learning should be performed at both steps of pattern processing, i.e., during the structural pattern generation and the grammar/control table induction. This two-step (self-)learning path in our model is denoted with dotted arrows in Fig. 1. In the model presented in the paper, the self-organizing map [34, 35], SOM, will be applied for the generation of vague structural patterns at the recognition phase [21]. Therefore, SOM is to be trained in the first step of the learning phase. At the same time, SOM generates vague patterns of a structural sample set. The SOM training process will be described in the next section. The induction algorithm for the generation of the DPLL(k) parser control table based on the structural sample set in the second step of the learning phase will be presented in Sect. 4.

3 Learning vague patterns with self-organizing maps

In machine learning, considered as the area of artificial intelligence [19], three generic approaches are considered: supervised learning, unsupervised learning and reinforcement learning. For the constructing of syntactic pattern recognition systems unsupervised learning is usually used [22]. Since in the first step of the learning in our system numerical patterns are assumed, the approaches corresponding to the decision-theoretic pattern recognition, like cluster analysis, principal component analysis, etc., or the neural network learning approaches can be applied. Let us note that during the learning process vague structural patterns are to be generated, i.e., the symbolic/discretized representations are to be defined and they are to be associated with the measures that can express their vagueness. (As it has been discussed in the previous section.) At the same time, the dimensionality reduction is to be done. (We assume an optimized, small number of terminal symbols of the underlying formal grammar in the second step.) Thus, a model that is centroid-based-clustering-like with a strong dimensionality reduction is preferable. In the area of neural networks typical models with unsupervised learning include self-organizing maps, SOM, adaptive resonance theory, ART, and Hebbian learning models. Taking into account the requirement analysis performed shortly above, self-organizing maps enhanced

with the centroid-based clustering seem to be the most suitable to meet these requirements, i.e., the generating of a low-dimensional discretized representation associated with the Euclidean distance in a feature space. Our model is just based on such an approach.

In our model, we follow the principles of competitive learning used in SOMs, i.e., the winner-take-all principle and the limiting of the strength of each neuron. The SOM neurons correspond to the classes which, in turn, are represented by the terminal symbols (from Σ_T, cf. Definition 1) of the underlying formal grammar. The Euclidean metric is used for the defining of the activation function. The function is calculated as the reciprocal of the distance between the unknown pattern and the centroid of the class/cluster corresponding to this neuron. The neurons' activations are normalized to the unit vector.

Let us consider the following example which is depicted in Fig. 2c. A new pattern is denoted with the small diamond and the distances between this pattern and the centroids of the classes (clusters): d, e and f are equal to: 5, 3 and 7, correspondingly. The activations for the classes, which correspond to the attributes of the related structural primitives (cf. Fig. 2b), are calculated as follows:

$$\mu_d = \frac{1/5}{1/5+1/3+1/7} = 0.3, \qquad \mu_e = \frac{1/3}{1/5+1/3+1/7} = 0.49,$$
$$\mu_f = \frac{1/7}{1/5+1/3+1/7} = 0.21.$$

If the unknown pattern is recognized, then the neurons which exhibit the highest activations are chosen for the generation of a fuzzy primitive.

Now, we can present the learning phase. Firstly, for each class (represented by the corresponding neuron) the maximum diameter is set. The new (unknown) pattern is added to the class, if the condition defining the (maximum) diameter of the corresponding cluster is not violated. Let us assume the following denotations.

$N_\Omega(j)$ denotes the cluster which relates to the neuron Ω after the j-th iteration, $N_\Omega(j) = \{\mathbf{V}_1, \ldots, \mathbf{V}_u\}$.
$\mathbf{C}_\Omega(j) = \frac{\mathbf{V}_1+\ldots+\mathbf{V}_u}{u}$ denotes the centroid of $N_\Omega(j)$.
$\mathbf{Y}$ denotes the unknown pattern shown at the $(j+1)$-th iteration.

The learning process is defined with the formulas 1 and 2.

$$N_\Omega(j+1) = \begin{cases} N_\Omega(j), & \mathbf{Y} \text{ has not been added to the class } \Omega \\ N_\Omega(j) \cup \{\mathbf{Y}\}, & \mathbf{Y} \text{ has been added to the class } \Omega \end{cases} \tag{1}$$

and

$$\mathbf{C}_\Omega(j+1) = \begin{cases} \mathbf{C}_\Omega(j), & \mathbf{Y} \text{ has not been added to the class } \Omega \\ \frac{u \cdot \mathbf{C}_\Omega(j)+\mathbf{Y}}{u+1}, & \mathbf{Y} \text{ has been added to the class } \Omega \end{cases} \tag{2}$$

4 Learning parser control table with DPLL(k) grammar induction

Before we define the induction algorithm which is applied for the generating of the parser control table during the learning phase in our approach (cf. Fig. 1), we introduce the class of the grammars used. The class of DPLL(k) grammars [17] is used because it fulfills two basic methodological requirements of syntactic pattern recognition [20, 22]. On the one hand, this class is of the *big generative power*. DPLL(k) grammars are able to generate all the context-free languages and the remarkable subclass of context-sensitive languages, including, e.g., $L_1 = \{a^n b^n c^n : n \geq 0\}$, $L_2 = \{a^n b^m c^n d^m : n, m \geq 0\}$ [22]. On the other hand, the *syntax analysis model* constructed for this class *is efficient*, i.e., the DPLL(k) parser is of the $\mathcal{O}(n^2)$ time complexity [17].

The big descriptive power of DPLL(k) grammars results from the fact that they belong to the family of programmed grammars. The first *(statically) programmed* context-free grammars were proposed by Rosenkrantz in 1969 [41] in order to generate some context-sensitive languages. The increase in the generative power of these grammars has been obtained by the controlling of derivations with the help of the static fields associated with the grammar productions. The static fields have been pre-specified during the defining of the grammar. Then, the concept of *dynamically programmed, DP,* context-free grammars was introduced in 1995 [16] (also described in [22]). The dynamic fields which can be processed (by the storing and retrieving of the indices of productions) during a derivation are used in these grammars. Let us introduce them formally with the following definition.

Definition 2 A *dynamically programmed, DP, (context-free) grammar* is a quadruple $G = (\Sigma_N, \Sigma_T, P, S)$, where: Σ_N is a set of nonterminal symbols; Σ_T is a set of terminal symbols; P is a set of n productions of the form: $p_i = (\pi_i, L_i, R_i, A_i, DCL_i)$, in which $\pi_i : \bigcup_{k=1,\ldots,n} DCL_k \rightarrow \{TRUE, FALSE\}$ is the predicate of applicability of p_i; $L_i \in \Sigma_N$ and $R_i \in \Sigma^*$ are the left- and right-hand sides of p_i, respectively; A_i is the sequence of actions *add, read, move* performed over $\bigcup_{k=1,\ldots,n} DCL_k$; DCL_i is the derivation control tape for p_i; S is the start symbol (axiom), $S \in \Sigma_N$. □

A pair (L_i, R_i) is called the core of p_i. For every $p_i, p_j \in P$, $i \neq j$, the core of p_i differs from the core of p_j. For every $p_i \in P$, the derivation control tape DCL_i with the head operations *add, read, move* is defined, where *add*(k, m) writes a symbol m on the cell of DCL_k under the head, *read*(k) returns the value which has been read by the head, and *move*(k) moves the head of DCL_k right.

A derivation is defined in the following way. At the beginning, the production (1) is applied. The production (i) is applied if its predicate of applicability π_i is true. (The predicate is defined with the help of the operation *read*.) After the application of the core of the production, the sequence of actions *add, move* is performed for selected tapes.

DP grammars and standard top-down parsable LL(k) grammars [37, 42] have been incorporated into DPLL(k) grammars in [17] in order to define a polynomial parser. The *LL*(k) parser makes a derivational step looking ahead to the successive k-length prefixes of the input word. (Let us remind that a word/string u is a *prefix* of a word/string w if there is a word/string v (possible empty) such that $w = uv$.) Before we present the definition of DPLL(k) grammars, we introduce the following notions and denotations.

Let $G = (\Sigma_N, \Sigma_T, P, S)$ be a context-free (dynamically programmed) grammar, $\eta \in \Sigma^*$, and $|\, x \,|$ denote the length of a string $x \in \Sigma^*$. $FIRST_k(\eta)$ denotes a set of all the terminal prefixes of the strings of the length k (or of a length less than k, if a terminal string shorter than k is derived from α) that can be derived from η in the grammar G, i.e.,

$$FIRST_k(\eta) = \{x \in \Sigma_T^* : (\eta \stackrel{*}{\Longrightarrow} x\beta \wedge \ |\, x \,| = k) \vee (\eta \stackrel{*}{\Longrightarrow} x \wedge \ |\, x \,| < k)\ ,\ \beta \in \Sigma^*\}.$$

Now, we can define DPLL(k) grammars [17].

Definition 3 Let $G = (\Sigma_N, \Sigma_T, P, S)$ be a (context-free) dynamically programmed grammar,

$$\stackrel{*}{\underset{\text{core}}{\Longrightarrow}}$$

denotes a sequence of derivation steps consisting in the applying of production cores only. *G* is called a *DPLL(k) grammar* iff the following two conditions are fulfilled.

1. For every two leftmost derivations in G

$$S \stackrel{*}{\Longrightarrow} \alpha A\theta \Longrightarrow \alpha\beta\theta \stackrel{*}{\underset{\text{core}}{\Longrightarrow}} \alpha x$$

$$S \stackrel{*}{\Longrightarrow} \alpha A\theta \Longrightarrow \alpha\gamma\theta \stackrel{*}{\underset{\text{core}}{\Longrightarrow}} \alpha y,$$

where $\alpha, x, y \in \Sigma_T^*, \beta, \gamma, \theta \in \Sigma^*, A \in \Sigma_N$, the following condition holds.

If $FIRST_k(x) = FIRST_k(y)$, then $\beta = \gamma$.

2. For G there exists $m > 0$ such that for any leftmost derivation

$$S \stackrel{*}{\Longrightarrow} \alpha A\theta \stackrel{\sigma}{\Longrightarrow} \alpha\beta\theta\ ,$$

where σ is the string of the indices of productions applied, if $|\, \sigma \,| \geq m$ then the first symbol of $\beta\theta$ is terminal. □

Now we can define the induction algorithm for DPLL(k) grammars. Firstly, let us introduce the preliminary notions and denotations, which concern the problem of grammatical induction [22].

A *sample* of a language L over an alphabet Σ_T is an ordered pair (S_+, S_-), where a finite set $S_+ \subseteq L$, $S_+ \neq \emptyset$ is called a *positive sample* and a finite set $S_- \subseteq (\Sigma_T^* \setminus L)$ is called a *negative sample* (i.e., $S_+ \cap S_- = \emptyset$). We talk about *text learning* (induction from a positive sample), if $S_- = \emptyset$. We talk about *informed learning* (induction from positive and negative samples), if $S_- \neq \emptyset$. The problem of grammatical induction consists in looking for a grammar G (or the control table of the underlying syntax analyzes A) which generates the language L. We say also that we look for a grammar which is *consistent* with the sample (S_+, S_-). In the case of text learning, G is *consistent* with (S_+, S_-) iff $\forall x \in \Sigma_T^* : x \in S_+ \Rightarrow x \in L(G)$.

Our induction method is, as usually in syntactic pattern recognition, a text learning method. Let us introduce the so-called *polynomial specification of the language* [32], which will be used by the grammatical induction rules for DPLL(k) grammars.

Definition 4 Let T be the set of terminal symbols occurring in S_+, V be a subset of positive integers. *Polynomial specification of a language* is of the form $L(T, V) = S_i^{w_j(n_k)}$, where w_j is a polynomial of a variable $n_k \in V$. S_i is called a *polynomial structure*, and it is defined in a recursive way as follows: (1) $S_i = (a_{i_1} \cdots a_{i_r})$, where $a_{i_j} \in T$ (Then S_i is called a *basic* polynomial structure.) or (2) $S_i = (S_{i_1}^{w_{i_1}(n_{i_1})} \cdots S_{i_r}^{w_{i_r}(n_{i_r})})$, where S_{i_k} is defined as in (1) or (2). (Then S_i is called a *complex* polynomial structure.) □

Let us consider the following example of a polynomial specification of a language. Let there be given $L(T, V) = (a^n b^{2n} (ab)^{n^2+1})^n$, where $T = \{a, b\}$ is a set of terminal symbols and $V = \{n\}$ is a set of integer variables. Then, the polynomial structures are defined for $L(T, V)$ as it is shown in Table 1.

The structure of the polynomial specification is shown in Fig. 3.

The rules of grammatical induction are based on the concept of the polynomial specification and its structures. Let us introduce them according to the theory introduced in [31].

The Rules of DPLL(k) Grammar Induction Let $L(T, V)$ be a polynomial specification of a language L. The DPLL(k) grammar $G = (\Sigma_N, \Sigma_T, P, S)$, which generates L, is constructed as follows.

Table 1 Exemplary polynomial structures

$S_1 = a^n b^{2n} (ab)^{n^2+1}$	$w_1(n) = n$
$S_{1_1} = a$	$w_{1_1}(n) = n$
$S_{1_2} = b$	$w_{1_2}(n) = 2n$
$S_{1_3} = ab$	$w_{1_3}(n) = n^2 + 1$

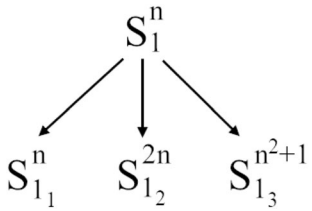

Fig. 3 The structure of the polynomial specification of the language $L = \{(a^n b^{2n} (ab)^{n^2+1})^n\}$

1. $\Sigma_T := T$.

2. For every $n \in V$, two positive integer variables v_n and d_n are defined.

3. For every polynomial structure of $L(T, V)$, which is denoted by S, the nonterminal $X_S \in \Sigma_N$ and positive integer variables c_S and e_S are defined.

4. For every polynomial structure of $L(T, V)$, which is denoted by S, the following productions of P are defined.

a) If S is of the form of $(a_1 \cdots a_r)$, then the corresponding productions are defined as it is shown in Table 2.

b) If S is of the form of $(S_1^{w_1(n_1)} \cdots S_r^{w_r(n_r)})$, then the corresponding productions are defined as it is shown in Table 3.

5. Let X_{S_1} be the nonterminal defined for the first polynomial structure. The initial production of P is defined as it is shown in Table 4.

Table 4 The rules of the initial production definition

l	π_l	$L_l \longrightarrow R_l$	A_l
1	*true*	$S \longrightarrow X_{S_1}$	$c_{S_1} := 0; \cdots ; c_{S_q} := 0;$ $e_{S_1} := -1; \cdots ; e_{S_q} := -1$ $v_{n_1} := 1; \cdots ; v_{n_t} := 1;$ $d_{n_1} := false; \cdots ; d_{n_t} := false$

Now, we can consider the example of the inducing of DPLL(k) grammar based of the rules introduced above. Let us assume that the polynomial specification of the language is given as in the example above, i.e., $L(T, V) = ((ab)^n c^{n^2+2})^n$. The only one variable $n \in V$ is used in $L(T, V)$. Therefore, we define two variables v_n and d_n. Since there are four polynomial structures: S_1, S_{1_1}, S_{1_2}, S_{1_3} (cf. Fig. 3), we should define four corresponding nonterminals: X_1, X_{11}, X_{12}, X_{13} in Σ_N, and eight variables: c_1, c_{11}, c_{12}, c_{13}, e_1, e_{11}, e_{12}, e_{13}. One can easily notice that S_{1_1}, S_{1_2}, and S_{1_3} are basic polynomial structures, and S_1 is a complex polynomial structure.

The set of DPLL(k) grammar productions is generated with the help of the induction rules introduced above as it is shown in Table 5.

At the end the initial production of P is defined in the following way. $\pi_1 = true,\ S \longrightarrow X_1,\ A_1 : c_1 := 0; c_{11} := 0; c_{12} := 0; c_{13} := 0;$ $e_1 := -1; e_{11} := -1; e_{12} := -1; e_{13} := -1; v_n := 1; d_n := false.$.

At the end of this section, let us consider the following example of the derivation of the word *abbabab* by the DPLL(k) grammar induced above. We monitor the values of the variables: v_n, d_n, c_1, c_{11}, c_{12}, c_{13}, e_1, e_{11}, e_{12} and e_{13} during the derivation process.

Table 2 The rules of the production definition for a basic polynomial structure

l	π_l	$L_l \longrightarrow R_l$	A_l
1_S	$c_S = 0$	$X_S \longrightarrow a_1 \cdots a_r X_S$	$c_S := 1; e_S := w(v_n)$
2_S	$c_S < e_S$	$X_S \longrightarrow a_1 \cdots a_r X_S$	$c_S := c_S + 1$
3_S	$(c_S = e_S)$ and $(d_n = true)$	$X_S \longrightarrow \lambda$	$c_S := 0;$
4_S	$(c_S = e_S)$ and $(d_n = false)$	$X_S \longrightarrow \lambda$	$c_S := 0; d_n := true$
5_S	$(c_S = e_S)$ and $(d_n = false)$	$X_S \longrightarrow a_1 \cdots a_r X_S$	$c_S := c_S + 1; v_n := v_n + 1; e_S := w(v_n)$

Table 3 The rules of the production definition for a complex polynomial structure

l	π_l	$L_l \longrightarrow R_l$	A_l
1_S	$c_S = 0$	$X_S \longrightarrow X_{S_1} \cdots X_{S_r} X_S$	$c_S := 1; e_S := w(v_n)$
2_S	$c_S < e_S$	$X_S \longrightarrow X_{S_1} \cdots X_{S_r} X_S$	$c_S := c_S + 1$
3_S	$(c_S = e_S)$ and $(d_n = true)$	$X_S \longrightarrow \lambda$	$c_S := 0;$
4_S	$(c_S = e_S)$ and $(d_n = false)$	$X_S \longrightarrow \lambda$	$c_S := 0; d_n := true$
5_S	$(c_S = e_S)$ and $(d_n = false)$	$X_S \longrightarrow X_{S_1} \cdots X_{S_r} X_S$	$c_S := c_S + 1; v_n := v_n + 1; e_S := w(v_n)$

Table 5 The generation of exemplary grammar productions

l	π_l	$L_l \longrightarrow R_l$	A_l
1_{11}	$c_{11} = 0$	$X_{11} \longrightarrow aX_{11}$	$c_{11} := 1; e_{11} := v_n$
2_{11}	$c_{11} < e_{11}$	$X_{11} \longrightarrow aX_{11}$	$c_{11} := c_{11} + 1$
3_{11}	$(c_{11} = e_{11})$ and $(d_n = true)$	$X_{11} \longrightarrow \lambda$	$c_{11} := 0;$
4_{11}	$(c_{11} = e_{11})$ and $(d_n = false)$	$X_{11} \longrightarrow \lambda$	$c_{11} := 0; d_n := true$
5_{11}	$(c_{11} = e_{11})$ and $(d_n = false)$	$X_{11} \longrightarrow aX_{11}$	$c_{11} := c_{11} + 1; v_n := v_n + 1;$ $e_{11} := v_n$
1_{12}	$c_{12} = 0$	$X_{12} \longrightarrow bX_{12}$	$c_{12} := 1; e_{12} := 2 * v_n$
2_{12}	$c_{12} < e_{12}$	$X_{12} \longrightarrow bX_{12}$	$c_{12} := c_{12} + 1$
3_{12}	$(c_{12} = e_{12})$ and $(d_n = true)$	$X_{12} \longrightarrow \lambda$	$c_{12} := 0;$
4_{12}	$(c_{12} = e_{12})$ and $(d_n = false)$	$X_{12} \longrightarrow \lambda$	$c_{12} := 0; d_n := true$
5_{12}	$(c_{12} = e_{12})$ and $(d_n = false)$	$X_{12} \longrightarrow bX_{12}$	$c_{12} := c_{12} + 1; v_n := v_n + 1;$ $e_{12} := 2 * v_n$
1_{13}	$c_{13} = 0$	$X_{13} \longrightarrow abX_{13}$	$c_{13} := 1; e_{13} := v_n * v_n + 1$
2_{13}	$c_{13} < e_{13}$	$X_{13} \longrightarrow abX_{13}$	$c_{13} := c_{13} + 1$
3_{13}	$(c_{13} = e_{13})$ and $(d_n = true)$	$X_{13} \longrightarrow \lambda$	$c_{13} := 0;$
4_{12}	$(c_{13} = e_{13})$ and $(d_n = false)$	$X_{13} \longrightarrow \lambda$	$c_{13} := 0; d_n := true$
5_{13}	$(c_{13} = e_{13})$ and $(d_n = false)$	$X_{13} \longrightarrow abX_{13}$	$c_{13} := c_{13} + 1; v_n := v_n + 1;$ $e_{13} := v_n * v_n + 1$
1_1	$c_1 = 0$	$X_1 \longrightarrow X_{11}X_{12}X_{13}X_1$	$c_1 := 1; e_1 := v_n$
2_1	$c_1 < e_1$	$X_1 \longrightarrow X_{11}X_{12}X_{13}X_1$	$c_1 := c_1 + 1$
3_1	$(c_1 = e_1)$ and $(d_n = true)$	$X_1 \longrightarrow \lambda$	$c_1 := 0;$
4_1	$(c_1 = e_1)$ and $(d_n = false)$	$X_1 \longrightarrow \lambda$	$c_1 := 0; d_n := true$
5_1	$(c_1 = e_1)$ and $(d_n = false)$	$X_1 \longrightarrow X_{11}X_{12}X_{13}X_1$	$c_1 := c_1 + 1; v_n := v_n + 1;$ $e_1 := v_n$

Production :	*Sentence derived* : S	v_n	d_n	c_1	e_1	c_{11}	e_{11}	c_{12}	e_{12}	c_{13}	e_{13}
1	X_1	1	*false*	0	−1	0	−1	0	−1	0	−1
1_1	$X_{11}X_{12}X_{13}X_1$	1	*false*	1	1	0	−1	0	−1	0	−1
1_{11}	$aX_{11}X_{12}X_{13}X_1$	1	*false*	1	1	1	1	0	−1	0	−1
4_{11}	$aX_{12}X_{13}X_1$	1	*true*	1	1	0	1	0	−1	0	−1
1_{12}	$abX_{12}X_{13}X_1$	1	*true*	1	1	0	1	1	2	0	−1
2_{12}	$abbX_{12}X_{13}X_1$	1	*true*	1	1	0	1	2	2	0	−1
3_{12}	$abbX_{13}X_1$	1	*true*	1	1	0	1	0	2	0	−1
1_{13}	$abbabX_{13}X_1$	1	*true*	1	1	0	1	0	2	1	2
2_{13}	$abbababX_{13}X_1$	1	*true*	1	1	0	1	0	2	2	2
3_{13}	$abbababX_1$	1	*true*	1	1	0	1	0	2	0	2
3_1	*abbabab*	1	*true*	0	1	0	1	0	2	0	2

5 Syntactic pattern recognition-based electrical load prediction system

As we have mentioned in the introduction, the *Syntactic Pattern Recognition-based Electrical Load Prediction, SPRELP, system* has been implemented [21] on the basis of the hybrid two-step scheme presented in Sect. 2. The system has been developed with one of the biggest Polish companies for electricity distribution. (The company supplies about 50 TWh to more than 5.5 million customer each year.) The prediction of customers' electrical demand for one day ahead is performed daily. It concerns a 24-hours period, and it is defined for each hour. This forecast is made on the basis of the following two groups of input data:

- day characteristics which include: a type of the day (a working day or non-working day), a day of a week (Monday - Sunday), a season (spring, summer, autumn or winter), an amount of non-working days before a given day, an amount of non-working days after a given day, etc.,

- an hourly temperature forecast for two days ahead and an hourly insolation forecast for two days ahead.

These data define a feature vector which is read into the *SPRELP* system (cf. Fig. 1). The short-term electric load, STEL, prediction in the *SPRELP* system was described in [21] in a detailed way. Let us only mention here that the generating of a vague pattern is made on the basis of the feature vector with the help of a self-organizing map as it has been presented is Sect. 3. During the self-learning stage, the *SPRELP* system uses the rules of the DPLL(k) grammar induction introduced in Sect. 4.

The example of daily actual electrical loads and short-term electric load predictions performed by the *SPRELP* system for selected months are shown in Fig. 4a–d. It can be easily seen that accuracy of predictions differs for various seasons. For months of winter and autumn when forecasts of temperature and insolation are less precise the load forecast errors are bigger than for months of spring and summer. The monthly forecast errors are shown in Fig. 4e–f.

The accuracy of short-term electrical load prediction methods which have been published recently and the accuracy of the *SPRELP* system forecast are included in Table 6. For the comparison of the forecast accuracy, the following metrics have been used: MAPE (Mean Absolute Percentage Error—the metric which is used commonly to evaluate the performance of STLF methods), MAE (Mean Absolute Error) and RMSE (Root Mean Square Error). (Some authors do not present MAE and/or RMSE errors.) As we can see, the variety of methodologies are used for the short-term electrical load forecasting, including: neural networks, evolutionary algorithms, various regression methods and hidden Markov models. Our method is the first syntactic pattern recognition method which has been used for short-term electrical load prediction. As one can notice, this method generates reasonably good forecasts with respect to the criteria of: MAPE, MAE and RMSE. As we have mentioned it in the introduction, the first version of the *SPRELP* system was presented in 2016 in [21]. The hybrid syntactic pattern recognition methodology that is the theoretical framework for the system has been developed since then, and the results

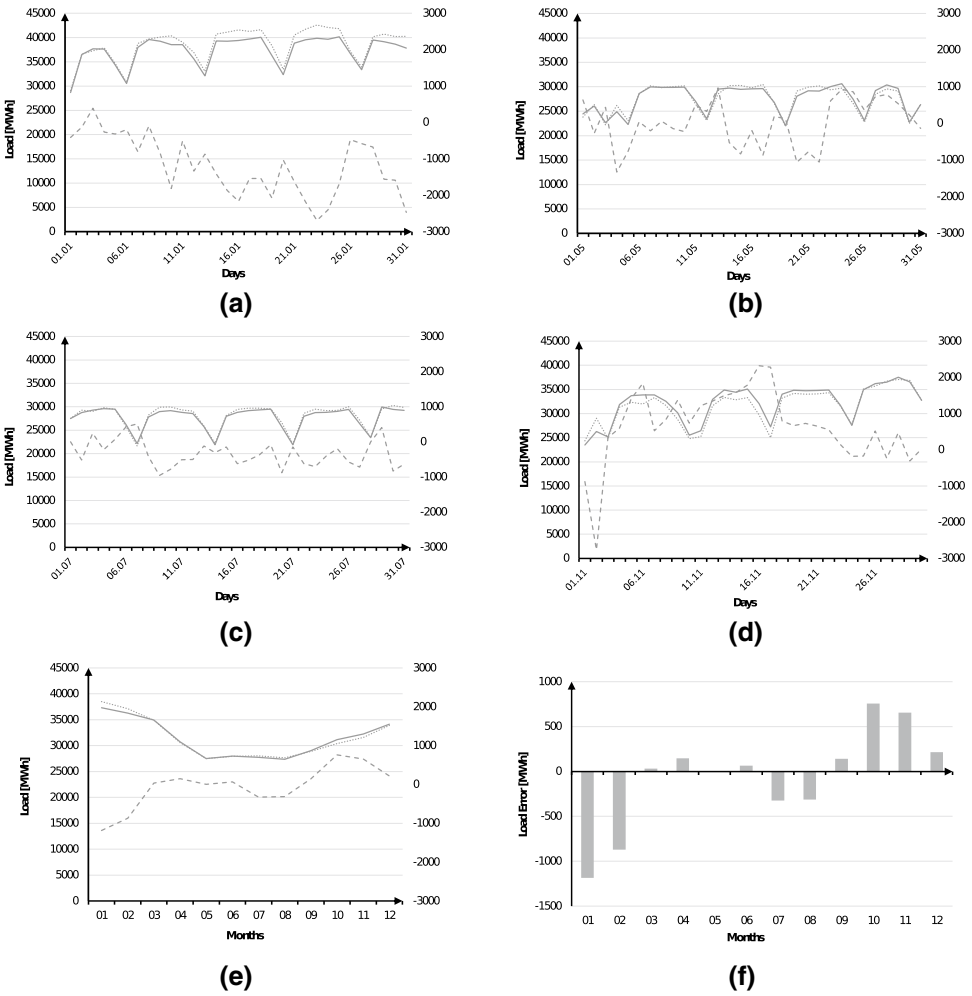

Fig. 4 The daily actual load in MWh (solid line) versus *SPRELP* forecast (dotted line) on the left scale and the forecast error (dashed line) on the right scale for typical months of: winter (**a**), spring (**b**), summer (**c**), autumn (**d**); the chart for the whole year—monthly (**e**); the monthly forecast error for the whole year (**f**)

Table 6 The performance of the STEL prediction methods

Method	Methodology	MAPE	MAE	RMSE
Amjady and Keynia [2]	NN/EA	1.68–2.02	87–124	*NA*
Fan and Hyndman [12]	AM	1.88	110	*NA*
Yang et al. [47]	ARIMA/NN	5.13	97	*NA*
Hong and Wang [28]	FIR	2.81	*NA*	*NA*
Wang et al. [45]	MLR	3.86	*NA*	*NA*
Hermias et al. [25]	ARIMA/HMM	5.92	454	579
Tian et al. [44]	NN	3.96	692	1134
Our method	SPR/NN	2.84	370	399

Methodologies (abbreviations): *NN* neural networks, *EA* evolutionary algorithm, *AM* additive regression, *ARIMA* autoregressive integrated moving average, *FIR* fuzzy interaction regression, *MLR* multiple linear regression, *HMM* hidden Markov model, *SPR* syntactic pattern recognition. *NA* means *not available* (in the original paper)

of this research are presented in the paper. The comparison of the *SPRELP* system performance (v. 2016 vs v. 2021) is shown in Table 7. In order to compare the recognition rates scored by *SPRELP* v. 2016 and *SPRELP* v. 2021, 2-sample McNemar statistical significance test has been applied. The tested hypothesis (H0 hypothesis) has stated that the recognition rates of *SPRELP* v. 2016 and *SPRELP* v. 2021 are equal, based on provided samples. To validate the H0 hypothesis, the significance levels of 9% and 5% have been assumed, with 2.87, 3.84 as the respective critical values of chi-square statistic. The chi-square statistic value of 4.19 has been obtained from the samples. The calculated value exceeds the critical values of chi-square statistics by both assumed significance levels. Thus, the tested H0 hypothesis can be rejected at both significance levels, i.e., the difference between the recognition rates scored by *SPRELP* v. 2016 and *SPRELP* v. 2021 is statistically significant at the levels.

6 Concluding remarks

As we have mentioned in the introduction, the availability of a self-learning method is a fundamental methodological requirement when the new syntactic pattern recognition model is proposed [20, 22]. The novel syntactic pattern recognition method which solves one of the crucial open problems of the losing of information about the uncertainty/unambiguity of objects during the generating their structural patterns on the basis of feature vectors was introduced in [21]. This pattern recognition method has been used successfully for the short-term electrical load prediction [21]. However, the occurring of the variety of numerical patterns at the input of the constructed short-term electrical load prediction system has encouraged us to develop the two-step hybrid learning method. This method uses self-organizing maps to generate vague structural patterns in the first step. The rules of the grammar induction are applied to generate the control table of the syntax analyzer which is used in the system for syntactic pattern recognition. On the one hand, the comparison of the method with other methods of the short-term electrical load prediction has shown that the implemented system generates reasonably good forecasts with respect to the criteria used for the assessment of the performance of such systems, i.e., MAPE, MAE and RMSE. On the other hand, it seems that the generating of forecasts for subareas having the homogenous characteristics instead of the making the forecast for the really big area (as it is made in case of the electricity distribution company mentioned) as a whole would give better results. For such a separation of areas, the graph model can be used as a representation formalism and the graph parsing [13–15] can be applied for syntactic pattern recognition at the meta-level of the whole structure. The research into developing such a two-level structural approach is to be started, and its results will be the subject of further publications.

Table 7 The performance of *SPRELP* system v. 2016 and *SPRELP* system v. 2021

Time period	SPR result	*SPRELP* v. 2016		*SPRELP* v. 2021	
		Number of patterns	Recognition rate	Number of patterns	Recognition rate
I/00:00-05:59	Recognized	267	0.73	290	0.79
	Unrecognized	98		75	
II/06:00-11:59	Recognized	276	0.76	296	0.81
	Unrecognized	89		69	
III/12:00-17:59	Recognized	287	0.79	306	0.84
	Unrecognized	78		59	
IV/18:00-23:59	Recognized	294	0.81	290	0.79
	Unrecognized	71		75	
Average	Recognized	281	0.77	296	0.81
	Unrecognized	84		69	

References

1. Alfares HK, Nazeeruddin M (2002) Electric load forecasting—literature survey and classification methods. Int J Syst Sci 33:23–34
2. Amjady N, Keynia F (2009) Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm. Energy 34:46–57
3. Baliyan A, Kumar G, Mishra SK (2015) A review of short term load forecasting using artificial neural networks models. Procedia Comput Sci 48:121–125
4. Bansal RC, Pandey JC (2005) Load forecasting using artificial intelligence techniques—a literature survey. Int J Comput Appl Technol 22:109–119
5. Behrens U, Flasiński M, Hagge L, Jurek J, Ohrenberg K (1996) Recent developments of the ZEUS expert system ZEX. IEEE Trans Nucl Sci NS 43:65–68
6. Bianchi FM, Maiorino E, Kampffmeyer MC, Rizzi A, Jenssen R (2017) Recurrent neural networks for short-term load forecasting—an overview and comparative analysis. Springer, Berlin
7. Bishop CM (2006) Pattern recognition and machine learning. Springer, New York
8. Bunke H, Sanfeliu A (eds) (1990) Syntactic and structural pattern recognition—theory and applications. World Scientific, Singapore
9. Chen CH (ed) (1978) Pattern recognition and signal processing. Springer, Netherlands
10. Duda RO, Hart PE, Stork DG (2001) Pattern classification. Wiley, New York
11. Fallah SN, Ganjkhani M, Shamshirband S, Chau KW (2019) Computational intelligence on short-term load forecasting—a methodological overview. Energies 12:393
12. Fan S, Hyndman RJ (2012) Short-term load forecasting based on a semi-parametric additive model. IEEE Trans Power Syst 27:134–141
13. Flasiński M (1988) Parsing of edNLC-graph grammars for scene analysis. Pattern Recognit 21:623–629
14. Flasiński M (1990) Distorted pattern analysis with the help of node label controlled graph languages. Pattern Recognit 23:765–774
15. Flasiński M (1993) On the parsing of deterministic graph languages for syntactic pattern recognition. Pattern Recognit 26:1–16
16. Flasiński M (1995) Towards quasi context sensitive structure grammars model for inference support in hybrid expert systems. Schedae Inform 6:161–173
17. Flasiński M, Jurek J (1999) Dynamically programmed automata for quasi contexts sensitive languages as a tool for inference support in pattern recognition-based real-time control expert systems. Pattern Recognit 32:671–690
18. Flasiński M, Reroń E, Jurek J, Wójtowicz P, Atłasiewicz K (2005) On the construction of the syntactic pattern recognition-based expert system for Auditory Brainstem Response analysis. In: Kurzyński M, Puchała E, Woźniak M, Żołnierek A (eds) CORES 2005. Advances in soft computing, vol 30. Springer, Cham, pp 503–510
19. Flasiński M (2016) Introduction to artificial intelligence. Springer, Switzerland
20. Flasiński M (2016) Chapter 1: Syntactic pattern recognition: paradigm issues and open problems. In: Chen CH (ed) Handbook of pattern recognition and computer vision, 5th edn. World Scientific, New Jersey-London-Singapore, pp 3–25
21. Flasiński M, Jurek J, Peszek T (2016) Application of syntactic pattern recognition methods for electrical load forecasting. In: Burduk R, Jackowski K, Kurzyński M, Woźniak M, Żołnierek A (eds) CORES 2015. Advances in intelligent systems and computing, vol 403. Springer, Cham, pp 599–608
22. Flasiński M (2019) Syntactic pattern recognition. World Scientific, New Jersey-London-Singapore
23. Flasiński M, Myśliński S (2010) On the use of graph parsing for recognition of isolated hand postures of Polish Sign Language. Pattern Recognit 43:2249–2264
24. Fu KS (1982) Syntactic pattern recognition and applications. Prentice Hall, Hoboken
25. Hermias JP, Teknomo K, Monje JCN (2017) Short-term stochastic load forecasting using autoregressive integrated moving average models and Hidden Markov model. In: Proceeding of the 2017 international conference on information and communication technologies (ICICT), Karachi, pp 131–137
26. Hippert HS, Pedreira CE, Souza RC (2001) Neural networks for short-term load forecasting: a review and evaluation. IEEE Trans Power Syst 16:44–55
27. Hong T, Fan S (2016) Probabilistic electric load forecasting: a tutorial review. Int J Forecast 32:914–938
28. Hong T, Wang P (2014) Fuzzy interaction regression for short term load forecasting. Fuzzy Optim Decis Mak 13:91–103
29. Huang KY (2002) Syntactic pattern recognition for seismic oil exploration. World Scientific, New Jersey-Singapore-London
30. Jurek J (2000) On the linear computational complexity of the parser for quasi context sensitive languages. Pattern Recognit Lett 21:179–187
31. Jurek J (2005) Syntactic pattern recognition with the GDPLL(k) grammars (in polish), vol 365. Habilitation dissertations series. Jagiellonian University Publishers, Cracow
32. Jurek J (2007) Generalisation of a language sample for grammatical inference of GDPLL(k) grammars. In: Kurzyński M, Puchała E, Woźniak M, Żołnierek A (eds) CORES 2007. Advances in soft computing, vol 45. Springer, Cham, pp 282–288
33. Jurek J, Wójtowicz W, Wójtowicz A (2020) Syntactic pattern recognition-based diagnostics of fetal palates. Pattern Recognit Lett 133:144–150
34. Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69
35. Kohonen T (1984) Self-organization and associative memory. Springer, Berlin
36. Kulikowski JL (1971) Algebraic methods in pattern recognition. Springer, Wien
37. Lewis PM II, Stearns RE (1968) Syntax-directed transduction. J ACM 15:465–488
38. Nti IK, Teimeh M, Nyarko-Boateng O, Edekoya AF (2020) Electricity load forecasting: a systematic review. J Electr Syst Inf Technol 7:13
39. Ogiela MR, Ogiela U (2014) Secure information management using linguistic threshold approach. Springer, London

40. Oommen BJ, Kashyap RL (1998) A formal theory for optimal and information theoretic syntactic pattern recognition. Pattern Recognit 31:1159–1177
41. Rosenkrantz DJ (1969) Programmed grammars and classes of formal languages. J ACM 16:107–131
42. Rosenkrantz DJ, Stearns RE (1970) Properties of deterministic top-down grammars. Inf Control 17:226–256
43. Taylor JW, McSharry PE (2007) Short-term load forecasting methods—an evaluation based on European data. IEEE Trans Power Syst 22:2213–2219
44. Tian C, Ma J, Zhang C, Zhan P (2018) A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. Energies 11:3493
45. Wang P, Liu B, Hong T (2016) Electric load forecasting with recency effect: a big data approach. Int J Forecast 32:585–597
46. Weron R (2007) Modeling and forecasting electricity loads and prices: a statistical approach. Wiley, Chichester
47. Yang Y, Wu J, Chen Y, Li C (2013) A new strategy for short-term load forecasting. In: Abstract and applied analysis, vol 208964
48. Yazidi A, Granmo O-C, Oommen BJ (2013) Learning automaton based on-line discovery and tracking of spatio-temporal event patterns. IEEE Trans Syst Man Cybern 43:1118–1130