Schedae Informaticae Vol. 26 (2017): 37–47
doi: 10.4467/20838476SI.17.003.8149

DPAR Grammars for ECG Diagnosis Justification

Piotr Flasiński
IT Systems Department, Jagiellonian University, ul. St. Łojasiewicza 4, Cracow 30-384
(Cooperation)

Abstract. A novel model of dynamically programmed attributed regular grammars, DPAR, for the ECG diagnosis justification purposes is presented in the paper. A formal model, power properties and a case of DPAR grammar are described. The formalism of DPAR grammars allows to differentiate between certain subclasses of ECG phenomena.

Keywords: syntactic pattern recognition, ECG analysis, diagnostic justification.

1. Introduction

ECG diagnosis process has been supported by computer systems for more than half a century [18]. The analysis of ECG signals has been aided both by decision-theoretic approach and by syntactic pattern recognition methods [1, 9, 11, 13, 16, 17, 20, 21, 22].

Syntactic pattern recognition methods treat a pattern as a complex structure and then decompose it into simpler subpatterns (that can also be decomposed into simpler subpatterns, etc. [2, 6, 8, 15, 19].

In electrocardiography, ECG signals are treated as complex structures consisting of substructures corresponding with different phases of electrical conduction of human heart's beating, for instance: P, T, Q, R, S waves, ST, PR segments and QRS complex.

This leads to a conclusion that the syntactic pattern recognition paradigm saying that a set of different structures is treated as a formal language, can be convenient for a descriptive structural characterization of ECG signals (e.g. for teaching medicine

Received: 25 November 2016 / Accepted: 20 December 2016

students how to describe and analyze ECG records properly). Of course, structural patterns of such a formal language can be then analyzed by formal automata [2, 6, 8, 15] which allows for identifying diseases in ECG records.

Basic skills that are required during medical studies and physician professional development include providing the right diagnosis along with its justification [14]. The justification itself should consist of a proper explanation on how the observations allowed to state the final diagnosis of the observed phenomena (diseases) [3]. This skill is crucial for the diagnosis of phenomena in charts like EEG or ECG [7, 10]. To be able to provide a correct interpretation, a physician has to determine both structural features of EEG/ECG charts and numerical parameters (frequency of QRS complexes, segments' lengths, etc.). A research into developing a syntactic pattern recognition-based system for teaching and evaluating students' diagnostic justification for ECG has been conducted by the author of this paper since 2012.

At the beginning, structural primitives to be used for the ECG diagnostic justification process have been specified in [4]. Based on the structural primitives, the author defined a new class of programmed attributed regular grammars, PARG which was used to generate ECG patterns. Eventually, the System for Teaching ElectroCardioGraphy (STECG) has been implemented. The core of the system was a syntax analyzer utilizing a class of programmed attributed finite-state automata, PAFSA [4].

The use of the STECG system showed that although the system is capable of differentiating between main classes of ECG phenomena (e.g. various atrioventricular and branch blocks) it fails when it comes to their specific subclasses in certain cases (e.g. between *Mobitz I* and *Mobitz II* subclasses of the *Second-degree atrioventricular block* class. This was caused by too weak generative power of PAR grammars and, in consequence, too weak discriminative power of corresponding PAFS automata.

The results of the research into enhancing a generative/discriminative power of the model have been presented in [5], where the author defined the dynamically programmed attributed regular grammar, DPAR grammar along with the programmed finite-state automaton, DPAFS automaton.

In this paper the author has presented a developed version of dynamically programmed attributed regular grammars - DPAR grammars along with their detailed formal model (presented in Section 2) , power properties (presented in Section 3) and an example of their application for the *second-degree atrioventricular block type Mobitz I* (presented in Section 4).

2. Formal model of dynamically programmed attributed regular grammars

In this chapter we introduce the formal model of DPAR grammars. At the beginning let us define a programmed attributed regular grammar, PAR grammar[4].

Let $\mathcal{B}$ be the set of the logical (Boolean) values (i.e. $\mathcal{B} = \{TRUE, FALSE\}$).

Definition 1. A programmed attributed regular grammar, PAR grammar is a quadruple

$$G = (V, \Sigma, P, S), \text{ where}$$

V is a finite set of symbols,
$\Sigma \subset V$ is a set of terminal symbols, $N = V \setminus \Sigma$ is a set of nonterminal symbols,
P is a finite set of productions of the form:

$$(\pi \ , \quad X \longrightarrow \alpha), \text{ in which}$$

$\pi : \mathcal{A} \longrightarrow \mathcal{B}$ is the predicate of the production applicability, $\mathcal{A}$ is a finite set of attributes,
$X \longrightarrow \alpha$, $X \in N$, $\alpha \in \Sigma \cup \Sigma N$ is called the core ,
$S \in N$ is the starting symbol. □

As described in the Introduction, PAR grammars are strong enough to model structural patterns of general classes of phenomena observed in ECG and their subclasses. This has been achieved by predicates of the production applicability, which assure that primitive parameters match predefined conditions.

What is more, the research on ECG phenomena and their structural modelling by PAR grammars showed that in order to be able to differentiate specific classes of phenomena, like *Mobitz I* and *Mobitz II*, sometimes attributes of structural primitives modelling the ECG records have to be compared not only with constant (predefined) values, but also with some of the previously analyzed structural primitives' attributes. This leads to the conclusion that we needed to construct a stronger grammar than the PAR grammar. Then it has been decided that a programming formalism has to be included to enable such dynamic comparisons.

Let $\mathbb{R}$ be the set of real numbers. Before we introduce DPAR grammars let us define auxiliary notions.

Let $\mathcal{A}_\Sigma$ be a finite set of attributes of terminal symbols and $\mathcal{V}_A$ be a finite set of auxiliary variables.

Definition 2. A simple condition over $\mathcal{A}_\Sigma, \mathcal{V}_A$ is defined in the following way.

- $TRUE, FALSE$ are simple conditions.
- If $a_1, a_2 \in \mathcal{A}_\Sigma \cup \mathcal{V}_A$ is a numerical attribute, $r \in \mathbb{R}$, then:
  $a_1 = a_2,\ a_1 \neq a_2,\ a_1 > a_2,\ a_1 \geq a_2,\ a_1 < a_2,\ a_1 \leq a_2,$
  $a_1 = r,\ a_1 \neq r,\ a_1 > r,\ a_1 \geq r,\ a_1 < r,\ a_1 \leq r$
  are simple conditions.
- If $a \in \mathcal{A}_\Sigma \cup \mathcal{V}_A$ is a logical attribute (flag), then a is a simple condition.

□

Definition 3. A valid condition over $\mathcal{A}_\Sigma, \mathcal{V}_A$, denoted $\mathcal{C}_{\mathcal{A}_\Sigma, \mathcal{V}_A}$ is defined in the following way.

- Any simple condition is a valid condition.
- If $sc_1, sc_2, ..., sc_n$ are simple conditions, then $sc_1\ AND\ sc_2\ AND\ ...\ AND\ sc_n$ is a valid condition, where AND is the conjunction operator[1].
- If c_1, c_2 are simple conditions or conjunction conditions, then $IF\ c_1\ THEN\ c_2$ is a valid condition, where $IF\ ...\ THEN\ ...$ is the conditional statement[2].

□

Definition 4. A simple ascription over $\mathcal{V}_A$ is defined in the following way.

- *none* representing no ascription is a simple ascription.
- If $a_1, a_2 \in \mathcal{V}_A$, $r \in \mathbb{R}$, then $a_1 := a_2, a_1 := r$ are simple ascriptions.

□

Definition 5. A valid ascription over $\mathcal{V}_A$, denoted $\mathcal{U}_{\mathcal{V}_A}$ is defined in the following way.

- Any simple ascription is a valid ascription.
- If $sa_1, sa_2, ..., sa_n$ are simple ascriptions, then a sequence $sa_1; sa_2; ...; sa_n$ is a valid ascription.

□

Now we introduce a definition of a dynamically programmed attributed regular grammar - DPAR grammar.

Definition 6. A dynamically programmed attributed regular grammar, DPAR grammar is a quadruple

$$G = (V, \Sigma, P, S), \text{ where}$$

V is a finite set of symbols,
$\Sigma \subset V$ is a set of terminal symbols, $N = V \setminus \Sigma$ is a set of nonterminal symbols,
P is a finite set of productions of the form:

$$(\pi,\ \ X \longrightarrow \alpha,\ \ CM),\ \text{in which}$$

$\pi : \mathcal{C}_{\mathcal{A}_\Sigma, \mathcal{V}_A} \longrightarrow \mathcal{B}$ is the predicate of the production applicability, $\mathcal{C}_{\mathcal{A}_\Sigma, \mathcal{V}_A}$ is the valid condition over $\mathcal{A}_\Sigma, \mathcal{V}_A$,

$X \longrightarrow \alpha\ ,\ X \in N\ ,\ \alpha \in \Sigma \cup \Sigma N$ is called the core,

CM is the control mapping ascribing values to auxiliary variables in the form of a valid ascription $\mathcal{U}_{\mathcal{V}_A}$ over $\mathcal{V}_A$,

[1] In the sense assumed in programming languages.
[2] In the sense assumed in programming languages.

$S \in N$ is the starting symbol. □

We define the derivation in a DPAR grammar in an analogical way as for Chomsky's grammar.

Definition 7. Let $\beta, \delta \in V^*$, $G = (V, \Sigma, P, S)$ be a DPAR grammar. We write

$$\beta \underset{G}{\Longrightarrow} \delta \qquad (\text{or } \beta \Longrightarrow \delta, \text{ if } G \text{ is known})$$

if the following conditions hold:

- $\beta = \eta_1 X \eta_2$, $\delta = \eta_1 \alpha \eta_2$ and $X \longrightarrow \alpha$ is the core of $(\pi : X \longrightarrow \alpha, CM) \in P$,
- π is $TRUE$,
- attributes of terminal symbols of δ have been set according to CM,
- auxiliary variables have been set according to CM.

We say that β directly derives δ in G. Such direct deriving is called a derivational step in G. The reflexive and transitive closure of the relation $\Longrightarrow$, denoted $\overset{*}{\Longrightarrow}$, is called a derivation in G. □

3. Power properties of DPAR grammars

In this section we introduce two theorems. In the first theorem we show that DPAR (and PAR) grammars are regular grammars, i.e. the form of their cores is in the same as that for regular grammars, so that DPAR grammars generate all regular languages. By the second theorem we show that DPAR grammars generate some proper context-free languages.

Theorem 1. $\mathcal{L}(REG) \subset \mathcal{L}(DPAR)$

To be able to prove the above statement let us define (accordingly to the DPAR grammar definition) a DPAR grammar with such predicates of the grammar's productions applicability that are always mapped to $TRUE$. Then we have:
$\mathcal{A}_\Sigma = \varnothing$
$\mathcal{V}_A = \varnothing$
We can see that we do not have any auxiliary variables, so CM control mapping will not be able to ascribe values.
Then all the productions are of the form:

$$(\pi = TRUE, \quad X \longrightarrow \alpha, \quad CM = none),$$

which is equal to the form:

$$X \longrightarrow \alpha,$$

The above form of productions is the same as the form of REG grammar productions. □

Theorem 2. There exists $L \in \mathcal{L}(CFG) \setminus \mathcal{L}(REG)$ *such that* $L \in \mathcal{L}(DPAR)$.

Let us take the typical context-free language $L_1 = \{a^n b^n \mid n \geq 1\}$.

We have to show that $L_1 \in \mathcal{L}(DPAR)$, i.e. there exists a grammar G_1 of the class DPAR such that $L_1 = L(G_1)$. To prove that $L_1 = L(G_1)$ we will show that $L_1 \subset L(G_1)$ (a) and that $L(G_1) \subset L_1$ (b).

Let us define $G_1 = (V_1, \Sigma_1, P_1, S_1)$.
$V_1 = \{S_1, A, B, a, b\}$,
$\Sigma_1 = \{a, b\}$,
$\mathcal{V}_A = \{l_{ab}\}$,
$P_1 = \{$

1. $\pi = TRUE, \quad S_1 \longrightarrow aA, \quad CM = (l_{ab} := 1),$
2. $\pi = TRUE, \quad A \longrightarrow aA, \quad CM = (l_{ab} := l_{ab} + 1),$
3. $\pi = (l_{ab} > 1), \quad A \longrightarrow bB, \quad CM = (l_{ab} := l_{ab} - 1),$
4. $\pi = (l_{ab} > 1), \quad B \longrightarrow bB, \quad CM = (l_{ab} := l_{ab} - 1),$
5. $\pi = (l_{ab} = 1), \quad B \longrightarrow b, \quad CM = none,$
6. $\pi = (l_{ab} = 1), \quad A \longrightarrow b, \quad CM = none.$

$\}$.

(a) $L_1 \subset L(G_1)$

Let us take any $\alpha \in \Sigma_1^*$ such that $\alpha \in L_1$. Let $n = k (k \geq 1)$ so $\alpha = a^k b^k$. We will show that $\alpha \in L(G_1)$. To do so we will show that $S \overset{*}{\underset{G_1}{\Longrightarrow}} \alpha$. Let us show the string of productions P_1 that derives α.

For $k = 1$: $S_1 \overset{(1)}{\underset{G_1}{\Longrightarrow}} aA \overset{(6)}{\underset{G_1}{\Longrightarrow}} ab$

For $k \geq 2$: $S_1 \overset{(1)}{\underset{G_1}{\Longrightarrow}} aA \overset{(k-1)*(2)}{\underset{G_1}{\Longrightarrow}} a^k A \overset{(3)}{\underset{G_1}{\Longrightarrow}} a^k bB \overset{(k-2)*(4)}{\underset{G_1}{\Longrightarrow}} a^k b^{k-1} B \overset{(5)}{\underset{G_1}{\Longrightarrow}} a^k b^k$

We can see that, for $k = 1$, the grammar G_1 allows to use the first (1) derivation (starting from the starting symbol S_1). This leads to ascribing the value of 1 to l_{ab}. Then, as $k = 1$, we can use only the last production (6) which leads us to the terminal symbol b, so it results in the word $ab = a^1 b^1$.

For the $k \geq 2$, after using the first (1) production only once, the grammar G_1 leads to multiple ($k - 1$ times) usage of the second (2) production. As a result we get a word $a^k A$, so it means the grammar has to supplement the b terminals. Thus the third (3) production is used only once, then we use $(k - 2)$ times the fourth production achieving the $a^k b^{k-1}$ word. As the fourth production decreases the l_{ab} attribute resulting in $l_{ab} = 1$ value, the only production that can be now used is the fifth (5) one, resulting in the terminal symbol b. The result of such derivation is the word $a^k b^k$.

(b) $L(G_1) \subset L_1$

Let us take $\alpha \in L(G_1), \alpha \in \Sigma_1^*$. To show that $\alpha \in L_1$ let us construct the tree of possible derivations in G_1. The tree has been presented in the Figure 1.

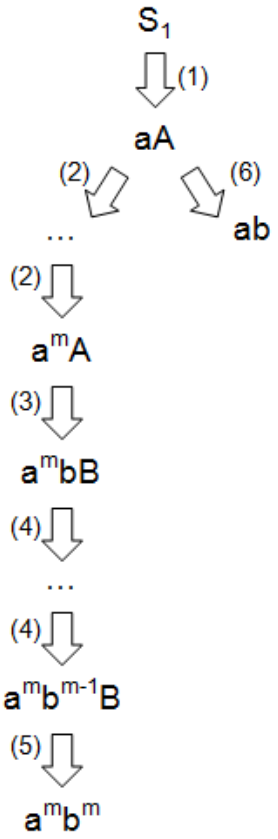

Figure 1. The tree of possible derivations in grammar G_1.

The first and the only production that can be applied at the beginning of the derivation process is the first (1) production of the grammar G_1 leading to a word in the form aA. Then, the grammar allows to use the second (2) or the sixth (6) production only. The sixth (6) production leads to a word ab ending up the process of the grammar's derivation. The second (2) production allows to apply it multiple times increasing the number of a terminal symbol. Afterwards, because of the constraints provided by π production applicability predicates, the only production that can be applied is the third (3) one. Next, the fourth (4) production is applied (optionally multiple times, depending on the amount of a terminal symbols added to the word during the derivation process) which then leads to the fifth (5) production. The process results in the word $a^m b^m$.

As we can see the grammar G_1 generates a language that consists of the words in the form $a^m b^m$, $m \geq 1$, which is the same form as the words of the language L_1. □

$qrs\_lack := FALSE$,
$potent\_last\_before\_qrs\_lack := 0$,

1. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(0)} \longrightarrow \mathbf{PR}\, X^{(1)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

2. $\pi = TRUE, \quad X^{(1)} \longrightarrow \mathbf{rs}\, X^{(2)}, \quad CM = none,$
3. $\pi = TRUE, \quad X^{(1)} \longrightarrow \mathbf{R}\, X^{(3)}, \quad CM = none,$
4. $\pi = TRUE, \quad X^{(2)} \longrightarrow \mathbf{ST+}\, X^{(4)}, \quad CM = none,$
5. $\pi = TRUE, \quad X^{(2)} \longrightarrow \mathbf{ST0}\, X^{(5)}, \quad CM = none,$
6. $\pi = TRUE, \quad X^{(3)} \longrightarrow \mathbf{ST0}\, X^{(6)}, \quad CM = none,$
7. $\pi = TRUE, \quad X^{(4)} \longrightarrow \mathbf{T+}\, X^{(7)}, \quad CM = none,$
8. $\pi = TRUE, \quad X^{(5)} \longrightarrow \mathbf{T+}\, X^{(8)}, \quad CM = none,$
9. $\pi = TRUE, \quad X^{(6)} \longrightarrow \mathbf{T+}\, X^{(9)}, \quad CM = none,$
10. $\pi = TRUE, \quad X^{(7)} \longrightarrow \mathbf{TP}\, X^{(10)}, \quad CM = none,$
11. $\pi = TRUE, \quad X^{(8)} \longrightarrow \mathbf{TP}\, X^{(11)}, \quad CM = none,$
12. $\pi = TRUE, \quad X^{(9)} \longrightarrow \mathbf{TP}\, X^{(12)}, \quad CM = none,$
13. $\pi = TRUE, \quad X^{(10)} \longrightarrow \mathbf{PP}\, X^{(13)}, \quad CM = (qrs\_lack := TRUE),$
14. $\pi = TRUE, \quad X^{(11)} \longrightarrow \mathbf{PP}\, X^{(14)}, \quad CM = (qrs\_lack := TRUE),$
15. $\pi = TRUE, \quad X^{(12)} \longrightarrow \mathbf{PP}\, X^{(15)}, \quad CM = (qrs\_lack := TRUE),$

16. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(13)} \longrightarrow \mathbf{PR}\, X^{(16)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

17. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(14)} \longrightarrow \mathbf{PR}\, X^{(17)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

18. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(15)} \longrightarrow \mathbf{PR}\, X^{(18)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

19. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(10)} \longrightarrow \mathbf{PR}\, X^{(16)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

20. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(11)} \longrightarrow \mathbf{PR}\, X^{(17)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

21. $\pi = ((qrs\_lack \implies l_{PR} < potent\_last\_before\_qrs\_lack), \quad X^{(12)} \longrightarrow \mathbf{PR}\, X^{(18)},$
$CM = (potent\_last\_before\_qrs\_lack := l_{PR};\ \ qrs\_lack := FALSE)$,

22. $\pi = TRUE, \quad X^{(16)} \longrightarrow \mathbf{rs}\, X^{(2)}, \quad CM = none,$
23. $\pi = TRUE, \quad X^{(17)} \longrightarrow \mathbf{rs}\, X^{(2)}, \quad CM = none,$
24. $\pi = TRUE, \quad X^{(18} \longrightarrow \mathbf{R}\, X^{(3)}, \quad CM = none.$

Table 1. DPAR grammar for second-degree atrioventricular block type Mobitz I.

4. Example of DPAR grammar

In this section, let us show an example of the DPAR grammar constructed for the second-degree atrioventricular block type Mobitz I. As we can see, for some productions, there is an auxiliary variable dynamically programmed and then used for a comparison in the following productions as a production applicability predicate.

The formal DPAR grammar for second-degree atrioventricular block type Mobitz I has been show in the Table 1.

5. Conclusions

As described in the introduction, a proper diagnostic justification is a very important ability during medical studies and the beginning of professional development of a physician.

We can distinguish two main kinds of information processing in medicine: analytic and automatic [12]. Automatic processing assumes that a diagnostic justification is achieved with the use of pattern recognition-like method - a new case that is to be diagnosed is compared with known cases [12]. This mechanism is similar to the standard pattern recognition method used in computer science. However, this kind of diagnosis is rather used by experienced physicians. For medicine students or novice doctors, it is more typical to analyze and interpret a case based on a biomedical knowledge. For ECG analysis, a chart is interpreted based on its structure and numerical parameters. This scheme corresponds with syntactic pattern recognition methods in computer science and seems to be more suitable for teaching medical students ECG interpretation and diagnosis.

In this paper we have presented the enhanced DPAR grammar which initially was designed as PAR grammar in [4] and then developed into first version of DPAR grammar in [5]. We have also described its detailed formal model along with its power properties. Eventually, we showed an example of DPAR grammar application for *Second-degree atrioventricular block type Mobitz I.*

6. References

[1] Belferte G., De Mori R., Ferraris F., *A contribution to the automatic processing of electrocardiograms using syntactic methods.* IEEE Trans. Biomed. Eng., 1979, 26, pp. 125-136.

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

[3] Cianciolo A.T., Williams R.G., Klamen D.L., Roberts N.K., *Biomedical knowledge, clinical cognition and diagnostic justification: a structural equation model.* Medical Education, 2013, 47, pp. 309-316.

[4] Flasiński M., Flasiński P., Konduracka E., *On the use of programmed automata for verification of ECG diagnoses.* Advances in Intelligent Systems and Computing 226, Springer, Berlin-Heidelberg-New York, 2013, pp. 591–599.

[5] Flasiński P., *Syntactic pattern recognition of ECG for diagnostic justification.* Machine Graphics & Vision International Journal, 2014, 24, 3/4, pp. 43–55.

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

[7] Gilhooly K.J., McGeorge P., Hunter J. et al., *Biomedical knowledge in diagnostic thinking: the case of electrocardiogram (ECG) interpretation.* European Journal of Cognitive Psychology, 1997, 9, pp. 199–223.

[8] Gonzales R.C., Thomason M.G., *Syntactic Pattern Recognition: An Introduction.* Addison-Wesley, Reading, 1978.

[9] Horowitz S.L., *A syntactic algorithm for peak detection in waveforms with applications to cardiography.* Comm. ACM, 1975, 18, pp. 281–285.

[10] Kolykhalov I.V., Iznak, A.F., Chayanov N.V. et al., *EEG mapping in diagnostic assessment of patients with mild dementia.* European Neuropsychopharmacology, 1997, 7, pp. 248.

[11] Koski A., Juhola M., Meriste M., *Syntactic recognition of ECG signals by attributed finite automata.* Pattern Recognition, 1995, 28, pp. 1927–1940.

[12] McLaughlin K.J., *The Contribution of Analytic Information Processing to Diagnostic Performance in Medicine*, Ph.D. Thesis, Erasmus University Rotterdam, 2007.

[13] Papakonstantinou G., Skordalakis E., Gritzali F., *An attribute grammar for QRS detection.* Pattern Recognition, 1986, 19, pp. 297–303.

[14] Patel V., Groen G., *Knowledge based solution strategies in medical reasoning.* Cognitive Science, 1986, 10, pp. 91–116.

[15] Pavlidis T., *Structural Pattern Recognition.* Springer, New York, 1997.

[16] Piętka E., *Feature extraction in computerized approach to the ECG analysis.* Pattern Recognition 1991, 24, pp. 139–146.

[17] Skordalakis E., *Syntactic ECG processing: a review.* Pattern Recognition 1986, 19, pp. 305–313, .

[18] Stallmann F.W., Pipberger H.V., *Automatic recognition of electrocardiographic waves by digital computer.* Circ. Res., 1961, 9, pp. 1138–1143.

[19] Tadeusiewicz R., Ogiela M.R., *Medical Image Understanding Technology.* Springer, Berlin-Heidelberg-New York, 2004.

[20] Trahanias P., Skordalakis E., *Syntactic pattern recognition of the ECG.* IEEE Trans. Patt. Analysis Mach. Intell., 1990, 12, pp. 648–657.

[21] Tumer M.B., Belfore L.A., Ropella K.M., *A syntactic methodology for automatic diagnosis by analysis of continuous time measurements using hierarchical signal representations.* IEEE Trans. on Syst. Man Cybern., 2003, 33, pp. 951–965.

[22] Udupa J., Murthy I.S.N., *Syntactic approach to ECG rhythm analysis.* IEEE Trans. Biomed. Eng., 1980, 27, pp. 370–375.