DERIVATION OF CONTEXT-FREE STOCHASTIC L-GRAMMAR RULES FOR PROMOTER SEQUENCE MODELING USING SUPPORT VECTOR MACHINE

Robertas Damaševičius

***Abstract**: Formal grammars can used for describing complex repeatable structures such as DNA sequences. In this paper, we describe the structural composition of DNA sequences using a context-free stochastic L-grammar. L-grammars are a special class of parallel grammars that can model the growth of living organisms, e.g. plant development, and model the morphology of a variety of organisms. We believe that parallel grammars also can be used for modeling genetic mechanisms and sequences such as promoters. Promoters are short regulatory DNA sequences located upstream of a gene. Detection of promoters in DNA sequences is important for successful gene prediction. Promoters can be recognized by certain patterns that are conserved within a species, but there are many exceptions which makes the promoter recognition a complex problem. We replace the problem of promoter recognition by induction of context-free stochastic L-grammar rules, which are later used for the structural analysis of promoter sequences. L-grammar rules are derived automatically from the drosophila and vertebrate promoter datasets using a genetic programming technique and their fitness is evaluated using a Support Vector Machine (SVM) classifier. The artificial promoter sequences generated using the derived L-grammar rules are analyzed and compared with natural promoter sequences.*

***Keywords**: stochastic context-free L-grammar, DNA modeling, machine learning, data mining, bioinformatics.*

***ACM Classification Keywords**: F.4.2 Grammars and Other Rewriting Systems; I.2.6 Knowledge acquisition; I.5 Pattern recognition; J.3 Life and medical sciences.*

***Conference**: The paper is selected from International Conference "Intelligent Information and Engineering Systems" INFOS 2008, Varna, Bulgaria, June-July 2008*

Introduction

Promoters are short regulatory DNA sequences that precede the beginnings of genes. They are common both in prokaryotic and eukaryotic genomes. It is important to distinguish between promoter and non-promoter sequences, because this distinction allows identifying starting locations of genes in uncharacterized DNA sequences. Though there are conserved patterns in the promoter sequences of a species, there are many exceptions which make the promoter recognition a complex problem. Although many complex machine learning methods have been proposed so far for promoter recognition, the problem is still open [Sobha Rani et al., 2007].

The crucial obstacle in finding mammalian promoters is that they usually do not share extensive sequence similarity even when they are functionally correlated, which prevents detection by sequence similarity-based search methods such as BLAST or FASTA. Mammalian promoters can be seen as small structures of coding regions with few functional elements (exons) interspersed in a larger sequence with no known biological function (introns) [Werner, 2003].

Modern promoter recognition tools use machine learning techniques such as Naive Bayes, Decision Trees, Hidden Markov Models, Neural Networks or Support Vector Machine (SVM) [Monteiro, 2005]. Best machine-learning based promoter recognition methods allow achieving up to 98% accuracy [Ranawana and Palade, 2005], however they do not provide any insight on the internal structure of the promoter sequences. Analysis of some promoters suggests that promoter sequences may have a modular structure. Identification of promoter structure may enhance our understanding of gene regulation mechanisms and genome evolution process.

Formal grammars can provide a means of describing complex repeatable structures such as DNA. The structure of promoter sequences can be described using a formal context-free grammar such as *L-grammar* (or *L-system*) [Lindenmayer, 1968], and the problem of promoter recognition can be replaced by grammar induction [Unold,

2007]. Several authors consider grammar induction in the context of bioinformatics as well as L-systems. The possibility of discovering the rewrite rule for L-systems and the state transition rules for cellular automata using genetic programming techniques is discussed in [Koza, 1993]. So-called semi-Lindenmayer systems are considered for the study of protein formation in [Marcus, 1974]. An algorithm to construct a short context-free grammar (also called program or description) that generates a given sequence is proposed in [Jimenéz-Montano, 1984]. The inference of regular language grammar rules based on n-grams and minimization of the Kullback-Leibler divergence is described in [Infante-Lopez and de Rijke, 2004]. O'Neill et al. [2004] generate regular expressions for promoter recognition problem using Grammatical Swarm technique. Each individual swarm particle represents choices of program construction rules, where these rules are specified using a Backus-Naur Form (BNF) grammar. Denise et al. [2003] generate genomic sequences and structures according to a given probability distribution and syntactical (grammatical) parameters. The method is applied for generating basic RNA secondary structures. Stochastic context-free grammars constructed from sample sets of sequences are also considered for modeling RNA sequences [Grate et al., 1994; Sakakibara et al., 1994]. The usefulness of Chomsky-like grammar representations for learning members of biological sequence families is analyzed in [Muggleton et al., 2000].

The aim of this paper is 1) to describe automatic derivation of stochastic L-grammar rules from promoter datasets using Support Vector Machine (SVM), and 2) to apply the derived L-grammar rules for structural analysis of the promoter sequences.

Modeling of DNA Sequences Using L-Grammar

If we treat the genome as a language, we can generalize the structural information contained in biological sequences and investigate it using formal language theory methods [Fernau, 2003]. Some aspects of formal languages are similar to biological processes in general and genetic mechanisms in particular, e.g.:

- *Pure grammars* do not differentiate between *terminal* and *non-terminal* symbols, so that all words generated from the grammar rules are put into the generated language. This notion is well-motivated biologically, because all symbols in a DNA sequence have similar role.
- *Erasing rule* $A \to \varepsilon$ models the death of a cell (which has been in a state A before its disappearance) or a delete mutation in a DNA sequence.
- *Chain rule* $A \to B$ reflects a change of a state of the corresponding cell or a single nucleotide mutation in DNA.
- *Repetition rule* $A \to AA$ models highly repetitive DNA sequences such as tandem repeats.
- *Growing rule* $A \to BC$ models the split of a cell being in a state A, into two children being in state B and C, or growth of a DNA sequence.
- *Stochastic rule* $p : A \to B$ models the random mutations of DNA sequences.

From the biological point of view, the components of any biological organism evolve simultaneously, so we cannot expect that biological processes could be modeled using a sequential approach. It is more likely that the cells that reproduce simultaneously would be modeled by a mechanism that is based on the same behavioral principles. This makes a special class of grammars, called *parallel grammars* [Fernau, 2003], particularly interesting for research of biological sequences such as DNA. L-systems are a special class of parallel grammars that can model the growth of living organisms, e.g. plant development, but also able to model the morphology of a variety of organisms [Prusinkiewicz and Lindenmayer, 1990]. They produce sentences that can be interpreted graphically to produce images of fractals or organisms. L-grammars have been applied also for modeling DNA sequences [Searls, 1993; Yokomori and Kobayashi, 1998; Mihalache and Salomaa, 2001; McGowan, 2002; Gheorge and Mitrana, 2004].

The essential difference between sequential grammars such as BNF and L-systems is that in sequential grammars the production rules are applied sequentially, one at a time; whereas in L-systems they are applied in parallel and may simultaneously replace all letters in a given word. Also, in L-systems there is no distinction

between the terminal and non-terminal symbols. All symbols that appear in the grammar are valid in the final string, and any symbol in the alphabet can head a rule [Prusinkiewicz and Hanan, 1989]. The recursive nature of the L-system rules leads to self-similarity and fractal-like forms which is also a property of DNA sequences [Abramson et al., 1999].

For modeling DNA sequences we use a *context-free stochastic L-grammar*, which is defined as a tuple:

$$G = \{V, \omega, R, P\} \tag{1}$$

where:

$V = \{A, C, G, T\}$ is a set of symbols containing elements (nucleotide types, in our case) that can be replaced, i.e. the alphabet.

$\omega = V^K$ is a K -length string of symbols that define the initial state of the system.

$R \subseteq V^1 \times V^L$ is a finite set of production rules defining the way symbols can be replaced with combinations of other symbols. A rule consists of two strings - the *predecessor* (an individual symbol from V) and the *successor* (L -length string composed from V elements). Each production rule refers only to an individual symbol and not to its neighbors.

P is a set of probabilities $p_j \in [0,1]$ that a production rule $r_j \in R$ will be applied.

Machine Learning Using Support Vector Machine

Support Vector Machines (SVM) [Vapnik, 1998] are one of the most popular tools in bioinformatics for supervised classification of genetic data (biosequences, protein structure data, microarray gene expression, etc.). SVM is a structural risk minimization-based method for creating binary classification functions from a set of labeled training data. SVM requires that each data instance is represented as a vector of real numbers in *feature space*. Hence, if there are categorical attributes, we first have to convert them into numeric data. First, SVM implicitly maps the training data into a (usually higher-dimensional) feature space. A *hyperplane* (decision surface) is then constructed in this feature space that bisects the two categories and maximizes the margin of separation between itself and those points lying nearest to it (the *support vectors*). This decision surface can then be used as a basis for classifying vectors of unknown classification.

Consider an input space X with input vectors $x_i \in X$, a target space $Y = \{1, -1\}$ with $y_i \in Y$ and a training set $T = \{(x_1, y_1), ..., (x_N, y_N)\}$. In the SVM classification, separation of the two data classes $Y = \{1, -1\}$ is done by the *maximum margin* hyperplane, i.e. the hyperplane that maximizes the distance to the closest data points and guarantees the best generalization on new examples. In order to classify a new point x_j , the classification function $g(x_j)$ is used:

$$g(x_j) = \operatorname{sgn}\left(\sum_{x_i \in SV} \alpha_i y_i K(x_i, x_j) + b \right) \tag{2}$$

where: SV are the support vectors, $K(x_i, x_j)$ is the kernel function, α_i are weights, and b is the offset parameter.

If $g(x_j) = +1$, x_j belongs to the *Positive* class, if $g(x_j) = -1$, x_j belongs to the *Negative* class, if $g(x_j) = 0$, x_j lies on the decision boundary and can not be classified.

Here we have a binary classification problem in which the outcomes are labeled either as positive (*P*) or negative (*N*) class. There are four possible outcomes from a binary classifier:

- *True positive* (*TP*) – the outcome from a prediction is *P* and the actual value is *P.*
- *False positive* (*FP*) – the outcome from a prediction is *P* and the actual value is *N.*

– *True negative* (*TN*) – both the prediction outcome and the actual value are *N.*

– *False negative* (*FN*) – the prediction outcome is *N* while the actual value is *P*.

The efficiency of the classification function $g(x_j)$ can be evaluated using several different metrics such as Sensitivity, Specificity, F-measure or Mathew's Correlation Coefficient. Here we use a simple *Accuracy* (ACC) metric, which is a measure of how well a binary classification test correctly identifies or excludes a condition.

$$ACC = \frac{n_{TP} + n_{TN}}{n_P + n_N} \cdot 100\% \quad (3)$$

where n_i is the number of i cases in the classification results.

The best possible classification method would yield 100% accuracy (all TPs and no FPs are found).

Grammar Rule Inference

Derivation of formal grammar rules, also known as *grammatical induction* or *grammar inference,* refers to the process of inducing a formal grammar (usually in the form of production rules) from a set of observations using machine learning techniques. The result of grammar inference is a model that reflects the characteristics of the observed objects. Here, for inference of L-grammar rules we use a "*trial-and-error*" method [Duda, 2001]. The method suggests successively guessing grammar rules (productions) and testing them against positive and negative observations. The best ruleset is then mutated randomly following the ideas of genetic programming until no improvement in accuracy is found within a certain number of iterations.

The rules of the L-system grammar are applied iteratively and simultaneously starting from the initial string. Let us denote $L(G)$ all those strings over V that can be generated by starting with the start string ω and then applying the production rules in R with probabilities P. We describe the grammar inference problem as a classical optimization problem as follows. Determine G, where V is constant, from a given set of input sequences $X = V^N$ such as to achieve the best classification $c(X')$:

$$c(X') = \max \sum_{X'} g(x_j) \quad (4)$$

where: $x_j, x_j \in X' \subset L(G)$ are strings produced by production rules R with probabilities P, and $g(x_j)$ is the classification function trained on a set of input sequences $x_i \in X$.

Case Study: Derivation of L-Grammar Rules from Promoter Datasets

Datasets

We use the following datasets:

1) The 2002 collection of data of drosophila (*D. melanogaster*) core promoter regions [Drosophila]. The test file contains 6500 examples (1842 promoters, 1799 introns, and 2859 coding sequences), each 300 bp length.

2) The collection of data of human and additional eukaryotic (vertebrate) promoter regions. The promoters were extracted from the Eukaryotic Promoter Database rel. 50; the negative set contains coding and non-coding sequences from the 1998 GENIE data set [Human]. The test file contains 5800 examples (565 promoters, 4345 introns, and 890 coding sequences), each 300 bp length.

The positive set consists of promoter sequences and the negative set consists of introns and coding sequences.

Description of L-grammar rule generation

The L-grammar rule generation process is performed in two stages as follows:

1) Derivation of a learned classifier. The promoter dataset is used for training a SVM classifier. As a classifier, we use SVM^{light}, which is an implementation of SVM in C [SVMlight]. Nucleotide sequences are mapped onto a feature space using *orthogonal encoding*, where nucleotides in a DNA sequence are viewed as unordered

categorical values and represented by the 4-dimensional orthogonal binary vectors: $\{A \to 0001, C \to 0010, G \to 0100, T \to 1000\}$. For training, the linear kernel function is used. The result is a model of a learned classifier that can separate promoter and non-promoter sequences.

2) L-grammar rule induction. The classifier model is used for evaluating the fitness of the generation of L-grammar rules. Rules are generated by L-grammar rule generator using a directed random search method: the best ruleset so far is mutated randomly and used to generate 1000 of 300 bp length L-system strings. The mutation parameters are the start string, successor strings and production rule probabilities. The number of rules as well as the predecessor strings is fixed. There are four rules for each type of nucleotide: A-rule, C-rule, G-rule and T-rule. The generated strings are encoded as 4-dimensional orthogonal binary vectors, fed to the trained SVM classifier and the accuracy of the classification is obtained. If the accuracy of the mutated ruleset is better than the accuracy of the previous best ruleset, the mutated ruleset is saved as the best ruleset; otherwise the previous best ruleset is retained. The process is continued until the required accuracy is achieved.

The structure of the L-grammar induction system is summarized in Fig. 1.

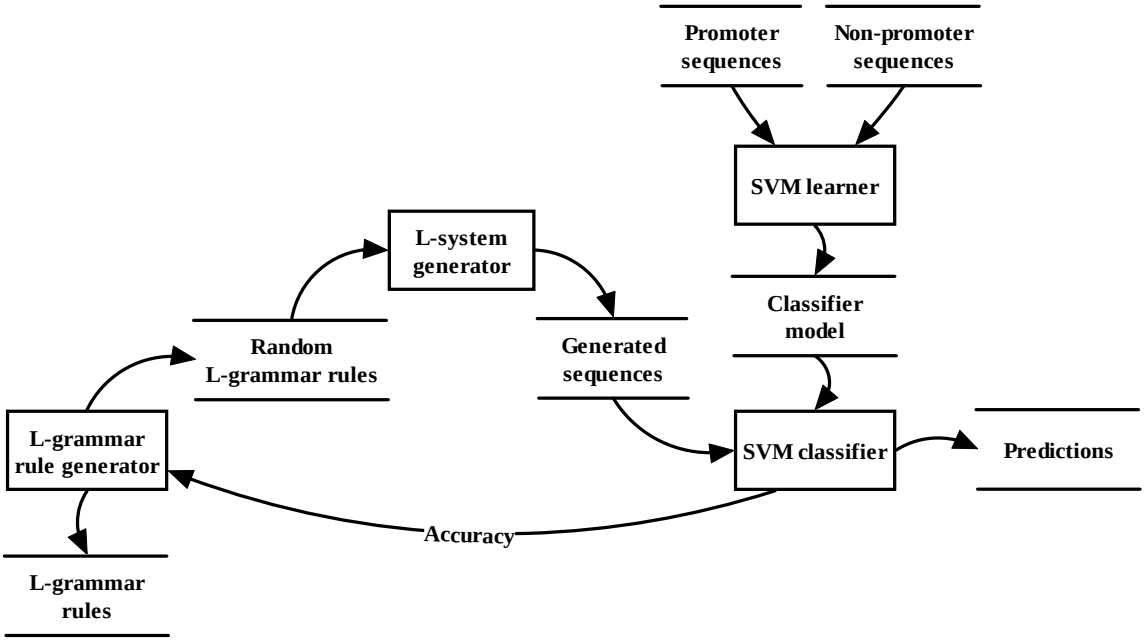

Fig. 1. Structure of the L-grammar induction system

Results

The classification was done in two stages: (1) SVM was trained using promoter vs. non-promoter sequences, and the trained classifier was used to classify (2a) random sequences labeled as promoters, and (2b) artificially generated (from the induced L-grammar rules) sequences labeled as promoters. The classification results using the Accuracy metric (see Eq. 3) are presented in Table 1.

We can see that artificially generated sequences can be classified vs. non-promoter sequences almost as good as real (natural) promoters. That suggests that induced L-grammar rules indeed capture some essential dataset properties of promoter sequences. The results are worse for the vertebrate dataset, because vertebrate promoters have more complex patterns with more irregularities and exceptions.

The derived stochastic L-grammar rules are presented in Fig. 2, a) and Fig. 2, b) for drosophila and vertebrate promoter sequences, respectively. Note that in drosophila grammar rules C and in vertebrate grammar rules T symbols are completely missing from the right (successor) side of production rules.

Table 1. Classification results

Classified sequences	No. of sequences	Classification accuracy	
		Drosophila dataset	Vertebrate dataset
Promoters vs. non-promoters	6500/5800	99.74%	94.67%
Random sequences	1000	3.3%	1.2%
Artificially generated sequences	1000	93.40%	92.10%

a)
```
Variables: A, C, G, T
Start:     AAACTAAT
Rules:     0.85: (A -> TATA),
           0.94: (C -> TA),
           0.91: (G -> TAG),
           0.10: (T -> TGA)
```

b)
```
Variables: A, C, G, T
Start:     A
Rules:     0.50: (A -> CGGAA),
           0.86: (C -> CCCCG),
           0.19: (G -> ACGG),
           0.50: (T -> AA)
```

Fig. 2. Stochastic L-grammar rules for generating
a) drosophila, and b) vertebrate promoter-like sequences

Evaluation

After analyzing the induced promoter sequence production rules, we can conclude that these rules can fairly good characterize the subsequences that are typical for promoters. The drosophila promoter production rules feature the TATA successor string, which matches the TATA-box (TATAA or TATAAA) typical for the drosophila promoters. Other subsequences typical for the promoter sequences are produced by the successive application of the production rules, e.g., the *Pribnow* box (TATAAT) is produced by two successive applications of A-rules.

The vertebrate promoters are typically more complex than the drosophila promoters, because there are many TATA-less promoters, which are characterized by other more complex subsequences. These subsequences also can be produced from the induced grammar rules, e.g., the CACG subsequence characteristic to the *E-box* (CACGTG) is produced by the successive application of the A-rule and G-rule. The AACC subsequence characteristic to the *Y-box* (GGGTAACCGA) is produced by the successive application of the G-rule, A-rule, and C-rule. Human promoter sequences are also characterized by the occurrence of DPE (*downstream promoter element*), a distinct 7-nucleotide subsequence (A/G)G(A/T)CGTG that is similar to the derived G-rule.

Conclusion

The advantage of the machine learning method combined with formal grammar is that additionally to recognition which does not provide any structural information, the modular structure of the analyzed genetic sequences can be identified. The structural analysis of the derived L-grammar production rules allows identifying common promoter sequence elements (so called “boxes”) such as TATA-box, Pribnow box, E-box, Y-box and DPE.

The classification results of the artificial promoter sequences generated using the derived L-grammar rules are almost as accurate as that of the natural promoters, thus showing a great deal of similarity between the discriminating features of both types of sequences.

Bibliography

[Abramson et al., 1999] Abramson, G., Cerdeira, H.A., Bruschi, C.: Fractal properties of DNA walks. Biosystems 49(1): 63-70 (1999)

[Denise et al., 2003] Denise, A., Ponty, Y., Termier, M.: Random generation of structured genomic sequences. Proc. of Int. Conf. on Research in Computational Molecular Biology (RECOMB'03), Berlin, Germany, 10-13 April (2003)

[Drosophila] Berkeley Drosophila Genome Project. Drosophila promoter dataset. Available at: http://www.fruitfly.org/seq_tools/datasets/Drosophila/promoter/

[Duda, 2001] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2 ed.), John Wiley & Sons (2001)

[Fernau, 2003] Fernau, H.: Parallel Grammars: A Phenomenology. Grammars 6(1): 25-87 (2003)

[Gheorghe and Mitrana, 2004] Gheorghe, M., Mitrana, V.: A formal language-based approach in biology. Comparative and Functional Genomics 5: 91–94 (2004)

[Grate et al., 1994] Grate, L., Herbster, M., Hughey, R., Haussler, D.: RNA modelling using Gibbs sampling and stochastic context-free grammars. Proc. of the 2nd Int. Conf. on Intelligent Systems for Molecular Biology. AAAI/MIT Press (1994)

[Human] Berkeley Drosophila Genome Project. Human promoter dataset. Available at: http://www.fruitfly.org/seq_tools/datasets/Human/promoter/

[Infante-Lopez and de Rijke, 2004] Infante-Lopez, G., de Rijke, M.: Alternative approaches for generating bodies of grammar rules. Proc. of 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004), Barcelona, Spain, 21-26 July, 454-461 (2004)

[Jiménez-Montaño, 1984] Jiménez-Montaño, M.A.: On the Syntactic Structure of Protein Sequences and the Concept of Grammar Complexity. Bull. Math. Biol. 46: 641-659 (1984)

[Koza, 1993] Koza, J.R.: Discovery of Rewrite Rules in Lindenmayer Systems and State Transition Rules in Cellular Automata via Genetic Programming. Symposium on Pattern Formation (SPF-93), February 13, Claremont, CA (1993)

[Lindenmayer, 1968] Lindenmayer, A.: Mathematical models for cellular interactions in development. Journal of Theoretical Biology 18: 280-315 (1968)

[Marcus, 1974] Marcus, S.: Linguistic structures and generative devices in molecular genetics. Cahiers Ling Theor. Appl. 1: 77–104 (1974)

[McGowan, 2002] McGowan, J.F.: Nanometer Scale Lindenmayer Systems. Proc. of SPIE Vol. 4807 (2002)

[Mihalache and Salomaa, 2001] Mihalache, V., Salomaa, A.: Lindenmayer and DNA: Watson-Crick D0L Systems. In G. Paun, G. Rozenberg, A. Salomaa (Eds.), Current Trends in Theoretical Computer Science, 740-751 (2001)

[Monteiro, 2005] Monteiro, M.I., de Souto, M., Gonçalves, L., Agnez-Lima, L.F.: Machine Learning Techniques for Predicting Bacillus subtilis Promoters. Advances in Bioinformatics and Computational Biology. LNCS 3594. Springer (2005)

[Muggleton, 2001] Muggleton, S.H., Bryant, C.H., Srinivasan, A., Whittaker, A., Topp, S., Rawlings, C.: Are grammatical representations useful for learning from biological sequence data? - a case study. Journal of Computational Biology 8(5), 493-522 (2001)

[O'Neill et al., 2004] O'Neill, M., Brabazon, A., Adley, C.: The Automatic Generation of Programs for Classification Problems with Grammatical Swarm. Proc. of Congress on Evolutionary Computation CEC 2004, Portland, USA, 104-110 (2004)

[Prusinkiewicz and Hanan, 1989] Prusinkiewicz, P., Hanan, J.: Lindenmayer Systems, Fractals, and Plants (Lecture Notes in Biomathematics). Springer-Verlag (1989)

[Prusinkiewicz and Lindenmayer, 1990] Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer-Verlag: New York (1990)

[Ranawana and Palade, 2005] Ranawana, R., Palade, V.: A neural network based multiclassifier system for gene identification in DNA sequences. Journal of Neural Computing Applications 14: 122–131 (2005)

[Sakakibara et al., 1994] Sakakibara, Y., Brown, M., Hughey, R., Mian, I.S., Sjoelander, K., Underwood, R., Haussler, D.: Stochastic context-free grammars for tRNA modelling. Nucleic Acids Res 25: 5112–5120 (1994)

[Searls, 1993] Searls, D.B.: The computational linguistics of biological sequences. In Hunter, L. (ed.): Artificial Intelligence and Molecular Biology, 47–120. AAAI/MIT Press (1993)

[Sobha Rani et al., 2007] Sobha Rani, T., Durga Bhavani, S., Bapi, R.S.: Analysis of E.coli promoter recognition problem in dinucleotide feature space. Bioinformatics 23(5):582-588 (2007)

[SVMlight] SVMlight. Available: http://svmlight.joachims.org/

[Unold, 2007] Unold, O.: Grammar-Based Classifier System for Recognition of Promoter Regions. In Adaptive and Natural Computing Algorithms, LNCS Vol. 4431. Springer Berlin/Heidelberg (2007)

[Vapnik, 1998] Vapnik, V.: Statistical Learning Theory. Wiley-Interscience, New York (1998)

[Werner, 2003] Werner, T.: The state of the art of mammalian promoter recognition. Briefings in Bioinformatics 4(1):22-30 (2003)

[Yokomori and Kobayashi, 1998] Yokomori, T., Kobayashi, S.: Learning local languages and their application to DNA sequence analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence 10(20): 1067-1079 (1998)

Authors' Information

Robertas Damaševičius *– Assoc. Prof.; Software Engineering Department, Kaunas University of Technology, Studentų 50, LT-51368, Kaunas, Lithuania; e-mail: robertas.damasevicius@ktu.lt*