JMLR: Workshop and Conference Proceedings 21:210–219, 2012 The 11th ICGI

Fuzzy Grammar-based Prediction of Amyloidogenic Regions

Olgierd Unold OLGIERD.UNOLD@PWR.WROC.PL
Institute of Computer Engineering, Control and Robotics, Wroclaw University of Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland

Editors: Jeffrey Heinz, Colin de la Higuera and Tim Oates

Abstract

In this paper, we address the problem of predicting the location of amyloidogenic regions in proteins. The language of protein sequence can be described by using a formal system such as fuzzy context-free grammar, and the problem of amyloidogenic region recognition can be replaced by fuzzy grammar induction. The induced fuzzy grammar achieved 70.6% accuracy and 96.7% specificity on a recently published amyloidogenic dataset. Our results are comparable to other methods dedicated to recognize amyloid proteins.

Keywords: Grammatical Inference, Grammar-based Classifier System, Fuzzy Grammar, Learning Classifier System, Amyloid Proteins

1. Introduction

Fuzzy languages and grammars have been introduced in (Lee and Zadeh, 1969). Contrary to the crisp language theory, the fuzzy language theory enables us to deal with errors in the input of the parser or the recognizer. Fuzzifying context-free languages (CFLs) is a significant step towards robustness in parsing CFLs. We refer the reader to (Mordeson and Mailk, 2002) for more details of fuzzy languages and fuzzy automata.

In this paper, we are interested in inducing a fuzzy context-free grammar (fCFG) that accepts a CFL given a finite number of positive and negative examples drawn from that language. Relatively few efforts have been made to learn fCFGs or fuzzy finite automata that recognize fCFG (Carter and Kremer, 2006; Molina-Lozano and Vallejo-Clemente, 2008; Wen and Min, 2006; Mozhiwen, 2006; Unold, 2010).

A proposed novel flexible approach based on a learning method, called fuzzy Grammar-based Classifier System (fGCS), is fuzzified version of Grammar-based Classifier System (GCS) – a model of Learning Classifer System (LCS) introduced in (Unold, 2005a), in which knowledge about the solved problem is represented by CFG in Chomsky Normal Form (CNF) productions.

Fuzzy grammar-based approach seems to be an ideal tool for parsing very often noisy (or erroneous) biological sequences, including amyloid proteins. Amyloidosis is a group of diseases – such as Alzheimer's, Huntington's disease, type 2 diabetes – in which a protein, called amyloid, builds up in the organs and tissues. Symptoms of amyloidosis depend on the organs it affects. Short sequences, named *hotspots*, play a key role in conversion of proteins from their soluble state into fibrillar, beta-structured aggregates.

Over the last few years, various computational methods have been developed to detect these *hotspots* in proteins, like AmylPred (Frousios et al., 2009), Pafig (J.Tian et al.,

2009), FoldAmyloid (Garbuzynskiy et al., 2010), and Waltz (Maurer-Stroh et al., 2010) (for available software dedicated to this task see (Hamodrakas, 2011)).

In this paper, a new classifier of amyloidosis based on an evolved fuzzy context-free grammar is proposed, and compared in terms of accuracy, sensitivity, and specificity with existing methods.

The remainder of this paper is organized as follows. Section 2 describes the GCS - original Grammar-based Classifier System that works with a crisp CFG. Section 3 introduces the new fGCS - extended system prepared to work with noisy input and fuzzy CFG. Section 4 illustrates experiments with the problem of amyloidogenic regions prediction. Finally, in Section 5, we present some concluding remarks.

2. The GCS

GCS is a new kind of Learning Classifier System (Holland, 1976) – a rule-based evolutionary learning classifier system, in which each classifier implements a partial task to the target problem. A typical goal of LCS is to evolve a population of classifiers to represent a complete solution to the target problem.

The GCS was widely described in works (Unold, 2005a,b, 2007, 2008). The system operates similar to the classic LCS but differs from them in (i) representation of classifiers population, (ii) scheme of classifiers' matching to the environmental state, (iii) methods of exploring new classifiers.

2.1. Representation

Population of classifiers has a form of a context-free grammar rule set in a Chomsky Normal Form (CNF). This is not a limitation because every context-free grammar can be transformed into equivalent CNF. Chomsky Normal Form allows only production rules in the form of $A \rightarrow a$ or $A \rightarrow BC$, where A, B, C are the non-terminal symbols and a is a terminal symbol. The first rule is an instance of *terminal rewriting rule*. These are not affected by the genetic algorithm (GA), and are generated automatically as the system meets unknown (new) terminal symbol. Left hand side of the rule (A) plays a role of classifier's action while the right side a classifier's (a or BC) condition. System evolves only one grammar according to the so-called Michigan approach. In this approach, each classifier – or grammar rule in GCS – is subject of the GA's operations. All classifiers (rules) form a population of evolving individuals. In each cycle, a fitness calculating algorithm computes a value (an adaptation) of each classifier and a discovery component operates only on a single classifier.

2.2. Matching

Automatic learning CFG is realized with so-called grammatical inference from text, understood here as a sequence of symbols over an alphabet (Gold, 1967). According to this technique system learns to use a training set that in this case consists of sentences both syntactically correct and incorrect. Grammar which accepts correct sentences and rejects incorrect ones canis able to classify so far unseen sentences from a testing set. Cocke-Younger-Kasami (CYK) parser, which operates in $\Theta(n^3 \cdot |G|)$ time (where n is the length

of the parsed string and $|G|$ is the size of the CNF grammar G) (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970), is used to parse sentences from the sets.

The environment of CFG is substituted by an array of CYK parser (note that the classifier system can be regarded as an artificial animal acting in/adapting to an environment). The classifier system matches the rules according to the current environmental state (state of parsing) and generates an action (or set of actions in GCS) pushing the parsing process toward the complete derivation of the sentence analyzed.

2.3. Discovery component

The discovery component in GCS is extended in comparison with standard LCS. In some cases, a *covering* procedure may occur, adding some useful rules to the system. It adds productions that allow continuing of parsing in the current state of the system. This feature utilizes for instance the fact that accepting 2-length sentences requires separate, designated rule in grammar in CNF.

Crowding technique preserves diversity in the population and also – which is vital in GCS – extends preservation of the dependencies between rules by replacing rules by the similar ones.

Apart from the *covering* a GA also explores the space searching for new, better rules. First GCS implementation used a simple rule fitness calculation algorithm which appreciated the ones commonly used in correct recognitions. Later implementations introduced the "fertility" technique, which made the rule fitness dependant on the amount of the descendant rules (in the sentence derivation tree). In both techniques classifiers used in parsing positive examples gain highest fitness values, unused classifiers are placed in the middle while the classifiers that parse negative examples gain lowest possible fitness values.

3. The fGCS

The architecture of the fGCS is based on the GCS system. The CYK table is the environment in which the fuzzy GCS operates. The learning (training) process is divided into the cycles (see Algorithm 1). During every cycle evolved (fuzzy) grammar is tested against every example of the training set (lines 4-8), then new rules (productions) are evolved or existing ones are modified (line 13), and another cycle begins. The fGCS differs from the GCS in such a way as crisp CFL differs from fuzzy CFL.

A fuzzy formal language is a formal language where each sequence of symbols has a degree of membership to the language. A fuzzy CFG

$$G = (V, T, P^T, P^N, S, \omega, \otimes, \oplus) \tag{1}$$

consists of a set of variables V, a set of terminals T, a set of terminal productions P^T, a set of non-terminal productions P^N, start symbol S, ω a set of weights defined over the production rules P, $\otimes$ denotes a t-norm, and $\oplus$, a t-conorm. Productions are of the form $A \xrightarrow{\omega} \alpha$ where $A \in V$, $\alpha \in (T \cup (V \times V))$ and $\omega \in [0, 1]$. The empty word is denoted by λ. The fuzzy language $L(G)$ generated by this fuzzy grammar is $\left\{(w, \mu_L(w))|w \in T^*, S \xrightarrow{*} w\right\}$, $\mu_L(w)$ represents the degree of membership of the word w to the language L and is obtained by applying the t-norm $\otimes$ to the weights of all productions involved in the generation of

w. Should the grammar be ambiguous, and a word w can be rechable from S by different sequences of productions, then t-conorm $\oplus$ will be used to calculate the final degree of membership from the degrees of membership obtained through different sequences of productions. A λ-free (fuzzy) context-free grammar G is in CNF iff $P \subseteq V \times [0,1] \times (T \cup (V \times V))$.

After the initial grammar has been generated, the weights are assigned to the production rules (line 2 of Algorithm 1). The initial weights are set to random numbers, except the terminal rules which obtain the value 1 (note that terminal rules are not a subject of discovering component). Setting the initial values is necessary in order to use different t-norms (in this paper only one t-norm and one t-conorm is used).

Algorithm 1: Training fGCS

Input: A learning set $D = (D_+, D_-)$
Output: A fuzzy grammar $G_{ind} = (V, T, P^T, P^N, S, \omega, \otimes, \oplus)$

1 **Method:**
2 Initialize G_{ind};
3 **while** *a stopping criterion is not met* **do**
4 **foreach** $d \in D$ **do**
5 find all derivations of d by calling CYK algorithm;
6 calculate $\mu_L(d)$;
7 calculate a label of d;
8 **end**
9 **foreach** $p \in P^N$ **do**
10 calculate the weight w of p;
11 **end**
12 assess an induced fuzzy grammar G_{ind};
13 find new two non-terminal productions using a GA;
14 **end**
15 **return** G_{ind};
16 **EndMethod.**

The final degree of membership of each sentence is worked out from the degrees of memberships obtained through different sequences of productions (t-norm *min*) in all possible derivation trees (t-conorm *max*) (line 6 of Algorithm 1).

To explain in more detail this step let us consider the exemplary fuzzy grammar

$G_f = (\{A, B, C, D\}, \{a, b, c\}, \{A \xrightarrow{1} a, B \xrightarrow{1} b, C \xrightarrow{1} c, S \xrightarrow{0.5} AD, S \xrightarrow{0.8} DC, D \xrightarrow{0.3} BC, D \xrightarrow{0.6} AB\}, S, \{1, 1, 1, 0.5, 0.8, 0.3, 0.6\}, min, max)$.

Note that the string *abc* can be derived from this grammar by two parse trees (Figure 2). Let $D(w, G)$ be the set of all possible derivations of the word w from S in the grammar G. If G is unambiguous the set $D(w, G)$ contains only one derivation for each word w. In this example, we have two derivations for the word *abc*, $D(abc, G_f) = \{d_1, d_2\}$, where:

$d_1 = S \to AD \to aD \to aBC \to abc$,
$d_2 = S \to DC \to ABC \to abC \to abc$.

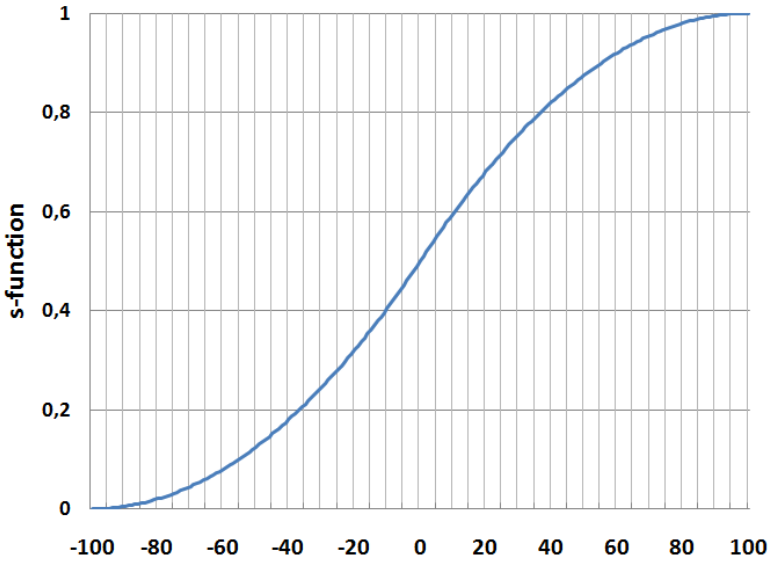

Figure 1: A membership *s-function* used to calculate weights of non-terminal productions; $a = -100, b = 0, c = 100$

To obtain the final degree of membership of the word *abc*, the following computations are used: $\mu_L(abc) = max(min(0.5, 1, 0.3, 1, 1), min(0.8, 0.6, 1, 1, 1)) = max(0.3, 0.6) = 0.6$

Figure 2: Fuzzy derivation trees for the input string *abc*

To obtain a crisp label of the sample, the fuzzy value needs to be defuzzified. The threshold was set to 0.5, and each sentence with a membership over this threshold is counted as a positive sentence, and with a membership below or equal to the treshold as a negative one (line 7).

In addition to this specification of a fGCS, a method for assigning a final weights to the productions is required. This is accomplished by applying – after each cycle of learning

(lines 9-11) – the fuzzy membership *s-function* (see Fig. 1) to the difference between a number of rules uses in positive sentences (n_p) and a number of rules uses in negative sentences (n_n). The *s-function* is given as

$$sfunction(x,a,b,c) = \begin{cases} 0, & for \quad x \leq a \\ 2\left(\frac{x-a}{c-a}\right)^2, & for \quad a < x \leq b \\ 1-2\left(\frac{x-c}{c-a}\right)^2, & for \quad b < x \leq c \\ 1, & for \quad x > c \end{cases} \tag{2}$$

In (2), x is the value of $n_p - n_n$, and a, b, c are the parameters describing a shape of *s-function*.

Having parsed all samples from a learning set, the induced grammar can be assessed (line 12). The fitness function counts the number of sentences parsed in a proper way i.e. labelled according to the input set. At the end of each cycle of learning a genetic algorithm explores the space of non-terminal productions searching for new, better rules (line 13).

Algorithm 2: Testing fGCS

Input: A tested fuzzy grammar $G_{test} = (V, T, P^T, P^N, S, \omega, \otimes, \oplus)$,
a testing set $D = (D_+, D_-)$
Output: An assessed fuzzy grammar G_{test}

1 **Method:**
2 **foreach** $d \in D$ **do**
3 find all derivations of d by calling CYK algorithm;
4 calculate $\mu_L(d)$;
5 calculate a label of d;
6 **end**
7 assess a tested fuzzy grammar G_{test};
8 **return**;
9 **EndMethod.**

To evaluate the evolved grammar, a testing (validation) phase on an unknown testing set must be performed (Algorithm 2). This phase tests if the grammar parses the sentences in the training set one by one agreeably to their input labels (lines 2-6). In line 7 of Algorithm 2 some statistical measures for an evolved fuzzy grammar are calculated, like Specificity, Sensitivity, and Accuracy. Specificity is a measure of the incidence of the negative results in testing all the negative (i.e. non-amyloid) sequences, i.e., (True Negatives/(False Positives + True Negatives)) x 100. Sensitivity is a measure of the incidence of positive results in testing all the positive (amyloid) sequences, i.e., (True Positives/(True Positives + False Negatives)) x 100. Accuracy is measured by the number of correct results, the sum of true positives and true negatives, in relation to the number of tests carried out, i.e. ((True Positives + True Negatives)/Total) x 100. True Positives are correctly recognized positive examples, True Negatives – correctly recognized negatives, False Negatives – positives recognized as negatives, and False Positives – negatives recognized as positives.

4. Experimental results

fGCS was tested among several different previously proposed methods dedicated to recognizing amyloid proteins. Here we briefly list the methods that we used to compare with our approach.

Waltz Waltz is a web-based tool that uses a position-specific scoring matrix (PSSM) to determine amyloid-forming sequences (Maurer-Stroh et al., 2010). Tool is available at http://waltz.switchlab.org/ (last accessed March 14, 2012).

AmylPred Method utilizes five different and independently published methods, to form a consensus prediction of amyloidogenic regions in proteins (Frousios et al., 2009). Tool is available at http://biophysics.biol.uoa.gr/AMYLPRED/input.html (last accessed March 14, 2012).

Pafig Prediction of amyloid fibril forming segments (Pafig) is based on support vector machines, to identify hexapeptides associated with amyloid fibrillar aggregates (J.Tian et al., 2009). Tool can be downloaded from http://www.mobioinfor.cn/pafig/index.htm (last accessed March 14, 2012).

FoldAmyloid In the method expected probability of hydrogen bond formation and expected packing density of residues are simultaneously used to detect amyloidogenic regions in a protein sequence (Garbuzynskiy et al., 2010). Tool is available at http://bioinfo.protres.ru/fold-amyloid/oga.cgi (last accessed March 14, 2012).

The comparison was performed using a recently published Amyloidogenic dataset, composed by 116 hexapeptides known to induce amyloidosis and by 162 hexapeptides that do not induce amyloidosis (Maurer-Stroh et al., 2010). The mentioned above methods were tested against the whole Amyloidogenic dataset (i.e. 282 hexapeptides), whereas fGCS was evaluated and tested against the same dataset using 10-fold cross validation method. A k-fold cross validation (k-fold CV) is a well-established statistical method of evaluating a learner, combining training and validation phases (Mosteller and Turkey, 1968). In k-fold CV the data is partioned into k folds, and next subsequently k iterations of learning and testing are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k-1$ folds are used for learning. To reduce bias in evaluating the performance of fGCS, we calculate the average of the classification accuracy, sensitivity, and specificity of the 10 runs of 10-fold cross validation. The experiments were conducted using the *fgcs* application written in Java NetBeans IDE 7.0 by (Kmiecinski, 2011).

Table 1 summarizes the performances of the five compared methods: AmylPred (Frousios et al., 2009), Pafig (J.Tian et al., 2009), FoldAmyloid (Garbuzynskiy et al., 2010) – version with a scale *expected number of contacts 8Å* named FoldAmylod1, version with a scale *bone bone donors* named FoldAmyloid2, version with a scale *hybrid* named FoldAmyloid3, version with a scale *bone bone acceptors* named FoldAmyloid4, version with a scale *tripple hybrid* named FoldAmyloid5 (the scale represents either expected packing density or the probability of formation of hydrogen bonds), Waltz (Maurer-Stroh et al., 2010) – version with ph=2.6 and threshold=79 named Waltz, version with ph=2.6 and threshold=92 named Waltz2, version with ph=2.6 and threshold=97 named Waltz3, and proposed fuzzy classificator. fGCS was run with inter alia following settings: non-terminal rules' population 100, crossover probability 20%, mutation probability 80%.

Note that fGCS model gained the highest Specificity among tested methods. The Specificity (called also true-negative rate) of 96.7%, shows that, in 96.7% of all cases of actual non-amyloids, fGCS made the right prediction. On the other hand, the 0.342 Sensitivity (called true-positive rate) value obtained tells us that fGCS rightly predicted only 34.3% of the actual amyloid cases, which is the lowest result among all methods. The Accuracy of 70.6% placed the fuzzy approach around the middle of results of the tested methods. It is worth remembering, that the model – as opposed to the compared methods – is general-purpose and was used essentially without any modifications in a specific application area. We believe that the results obtained by fGCS can be improved by (1) tuning parameters of the method, (2) using specific t-(co)norms and membership functions, (3) and introducing domain knowledge.

Table 1: Comparison of the induced fuzzy Grammar-based Classifier System (fGCS) with different methods. After each execution of fGCS inside 10-fold CV, four numbers were calculated: True Positives (correctly recognized amyloids), True Negatives (correctly recognized non-amyloids), False Negatives (amyloids recognized as non-amyloids), and False Positives (non-amyloids recognized as amyloids). Then the average of these numbers was found, and Specificity, Sensitivity, and Accuracy were calculated. The table is arranged in order of decreasing Accuracy.

Method	Specificity	Sensitivity	Accuracy
Waltz2	79.6	77.6	78.8
AmylPred	88.3	58.6	75.9
Waltz3	93.8	49.1	75.2
Pafig	57.4	89.7	70.8
fGCS	96.7	34.3	**70.6**
FoldAmyloid3	64.2	74.1	68.3
Waltz1	46.3	97.4	67.6
FoldAmyloid1	62.3	74.1	67.3
FoldAmyloid5	59.9	77.6	67.3
FoldAmyloid2	69.7	60.3	65.8
FoldAmyloid4	59.3	69.8	63.7

5. Conclusions

We have introduced the fuzzy grammar approach for recognizing amyloidogenic regions in proteins. The algorithm efficiently represents and manipulates fuzzy context-free grammar using rule-based evolutionary learning classifier system. We find that general-oriented method gives results comparable to the computational methods dedicated to detecting these hotspots in proteins. fGCS achieved 70.6% accuracy in the testing set, and proved to be extremely specific (96.7%), and as such can be used with a high reliability to recognize non-amyloid proteins.

Note that many problems both in natural language processing and bioinformatics can be described by a fuzzy grammar due to imprecision and fuzziness. The analysed sequences

are very often not only noisy, but also erroneous, and in such cases fuzzy formalism seems to be indispensable.

Further work is ongoing to investigate the use of different t-(co)norms and membership functions. An interesting question is whether the additional knowledge could be introduced. In (Unold and Jaworski, 2010), we showed that learning of (crisp) context-free grammar supported by a partial knowledge of the structure improves the efficiency of the whole process. We hope that the manual input of some fuzzy rules describing well known properties of proteins into the initial grammar can significantly speed up an induction. Moreover, such fixed rules can protect GA-based algorithm against pre-mature convergence.

References

P. Carter and S.C. Kremer. Fuzzy grammar induction from large corpora. In *2006 IEEE International Conference on Fuzzy Systems*, pages 2083–2089, 2006.

J. Cocke and J.T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.

K.K. Frousios, V.A. Iconomidou, C.M. Karletidi, and S.J. Hamodrakas. Amyloidogenic deteminants are usually not buried. *BMC Structural Biology*, 9:44, 2009.

S.O. Garbuzynskiy, M.Y. Lobanov, and O.V. Galzitskaya. An introduction to variable and feature selection. *Bioinformatics*, 26:326–332, 2010.

E. Gold. Language identification in the limit. *Information Control*, 10:447–474, 1967.

S.J. Hamodrakas. Protein aggregation and amyloid fibril formation prediction software from primary sequence: Towards controlling the formation of bacterial inclusion bodies. *FEBS Journal*, 278(14):2428–2435, 2011.

J. Holland. *Adaptation. In Progress in theoretical biology.* Plenum, New York, 1976.

J.Tian, N. Wu, J. Guo, and Y. Fan. Prediction of amyloid fibril-forming segments based on a support vector machine. *BMC Bioinformatics*, 10 (Suppl 1):S45, 2009. doi: doi: 10.1186/1471-2105-10-S1-S45.

T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report afcrl-65-758, Air Force Cambridge Research Lab, Bedford, MA, 1965.

M. Kmiecinski. Prediction of patterns from biosequences using fuzzy grammars. Master's thesis, Wroclaw University of Technology, Wroclaw, 2011. in Polish.

E.T. Lee and L.A. Zadeh. Note on fuzzy languages. *Inform. Sci.*, 1:421–434, 1969.

S. Maurer-Stroh, M. Debulpaep, N. Kuemmerer, M. Lopez de la Paz, I.C. Martins, J. Reumers, K.L. Morris, A. Copland, L. Serpell, L. Serrano, J.W. Schymkowitz, and F. Rousseau. Exploring the sequence determinants of amyloid structure using position-specific scoring matrices. *Nat Methods*, 7:237–242, 2010.

H. Molina-Lozano and E.E. Vallejo-Clemente. Dna sequence analysis using fuzzy grammars. In *2008 IEEE International Conference on Fuzzy Systems*, pages 1915–1921, 2008.

J.N. Mordeson and D.S. Mailk. *Fuzzy Automata and Languages: Theory and Applications*. Chapman and Hall, 2002.

F. Mosteller and J.W. Turkey. *Data analysis, including statistics*. Handbook of Social Psychology. Addison-Wesley, Reading, MA, 1968.

W. Mozhiwen. An evolution strategy for the induction of fuzzy finite-state automata. *Journal of Mathematics and Statistics*, 2(2):386–390, 2006.

O. Unold. Context-free grammar induction with grammar-based classifier system. *Archives of Control Science*, 15 (LI)(4):681–690, 2005a.

O. Unold. Playing a toy-grammar with gcs. In J. Mira and J.R. Álvarez, editors, *IWINAC 2005*, number 3562 in LNCS, pages 87–98, 2005b.

O. Unold. Grammar-based classifier system for recognition of promoter regions. In B. Beliczynski and et al., editors, *ICANNGA 2007*, number 4431 in LNCS, Part I, pages 798–805, 2007.

O. Unold. Inducing regular languages using grammar-based classifier system. In A. Clark, F. Coste, and L. Miclet, editors, *ICGI 2008*, number 5278 in LNCS (LNAI), pages 298–300, 2008.

O. Unold. Learning fuzzy context-free grammar - a preliminary report. In J.M. Sempere and P. Garcia, editors, *ICGI 2010*, number 6339 in LNCS (LNAI), pages 309–312, 2010.

O. Unold and M. Jaworski. Learning context-free grammar using improved tabular representation. *Appl. Soft Comput. J.*, 10(1):44–52, 2010.

M.Z. Wen and W. Min. Fuzzy automata induction using construction method. *Journal of Mathematics and Statistics*, 2(2):395–400, 2006.

D. Younger. Recognition and parsing of context-free languages in time n3. Technical report, University of Hawaii, Department of Computer Science, 1967.