Deriving a Concise Description of Non-Self Patterns in an Artificial Immune System

Sławomir T. Wierzchoń

Institute of Computer Science
Polish Academy of Sciences
ul. Ordona 21
01-267 Warsaw, Poland

and

Department of Computer Science
Białystok University of Technology
Białystok, Poland
E-mail: stw@ipipan.waw.pl

Abstract: From a computer science perspective the immune system is a complex, self organizing and highly distributed system that has no centralized control and uses learning and memory when solving particular tasks. The learning process does not require negative examples and the acquired knowledge is represented in explicit form. The main actors of the immune systems are lymphocytes equipped with a set of receptors recognizing intruders, or pathogens (i.e. viruses, bacteria, etc.). Because the receptors on a surface of a single lymphocyte are of identical structure, and they recognize only a narrow class of pathogens, we can treat them as a single receptor from an abstract point of view. In this paper we focus on a binary AIS in which all the information is represented by the bit strings of fixed length. A receptor activates if it matches a bit string, and as the match rule we take the k-contiguous bits rule: two strings match if they have the same bits in at least k contiguous positions. With such a method of receptor activation we study the discriminative power of the receptors repertoire. First, a method for calculating the number of strings recognized by a single receptor is proposed, and next we investigate how this number decreases when the size of the repertoire increases. Second, we give a method enabling to determine the number of strings which cannot be detected by an „ideal" repertoire in the presence of a set of self strings (i.e. strings representing normal behavior of a system). These results are of importance when constructing an optimal receptors repertoire.

Keywords: Binary Immune System, Schemas, Binary Receptors, Detection Probability, Lower Bounds on Failure Probability, Maximal Detectability

1. Introduction

Artificial Immune Systems (AISs) are computer systems exploiting the natural immune system metaphor: protect an organism against invaders. Hence, a natural field of applications of AIS is computer security - see (Kephart, 1994) for exhaustive discussion of this topic. But the notion of invader can be extended: for instance a fault occurring in a system disturbs patterns of its regular functioning. Thus fault, or anomaly detection is another field of applications (Dasgupta, 1999). We can extend the notion of invader even further. A problem with unknown solution is something that „invades" our mind. Like a specialized antibody produced by an organism to cope with antigen (invader), a proper solution deactivates the problem; hence a problem to be solved can be treated as an antigen and the solution of this problem as an antibody. AIS have found interesting and successful applications in machine learning (Hunt and Cooke, 1996), information retrieval (Hunt, Cooke and Holstein, 1995), binary patterns recognition (Smith, Forrest and Perelson, 1993), operation research (Hart, Ross, and Nelson, 1998) or numeric optimization (Bersini and Varel, 1990). Recently edited volume (Dasgupta, 1998) reviews ideas and applications of AIS while Perelson and Weisbuch (1997) give an exhaustive overview of basic facts concerning immunology, and analytical methods used in the field.

Developing AIS's computer scientists are attracted by the next key features of natural immune system, or NIS (cf. (Hofmeyr, 1995) for details):

- *Distributed detection*: the receptors used by the NIS are highly distributed and are not subjected to centralized control. Hence, the NIS resembles what we a multiagen system.
- *Imperfect detection*: no perfect matching of antigen by antibody is requested. Partial detection increases flexibility of the system. This is very important since in a human organism there is about 10^6 self patterns and about 10^{16} non-self patterns.
- *Anomaly detection*: the NIS correctly identifies never seen pathogens.
- *Adaptability*: NIS can learn the structures of the pathogens and remember those structures; facing already „seen" pathogen it quickly remembers its structure. The „memory" of the NIS is effectively managed: rarely used information is forgotten.
- *Self organization*: the memory cells are organized into so-called idiotypic network (a hypotheses stated by Jerne, 1974) that changes in time. The NIS resembles Kohonen neural networks in this aspect. The idea of self-organization is especially attractive when building learning systems, cf. (Hunt and Cooke, 1996).

- *No need for negative examples*: in many machine-learning systems we vast the time to collect appropriate negative examples. The NIS doesn't need such examples, and it correctly recognizes non-self patterns.
- *Explicit symbolic representation*: All the knowledge acquired by the NIS is represented in a fixed form forced by the structure of the receptors on the surface of the lymphocytes.
- *Uniqueness*: the NIS of each individual is unique, and the system solves problems in a unique way.

The basic building blocks of the NIS are white blood cells called lymphocytes. The size of an organism determines the number of lymphocytes: mice have of the order of 10^8 lymphocytes, while humans have of the order of 10^{12}. There are two major classes of lymphocytes: B-lymphocytes, or B-cells, produced in the bone marrow in the course of so-called clonal selection (described later), and T-lymphocytes, or T-cells, processed in the thymus. Roughly (but not quite precisely) speaking B lymphocytes are related to humoral immunity: they secrete antibodies. Among the B-cells are „memory cells". They live relatively long and „remembering" foreign proteins they constantly restimulate the immune response of the organism. On the other hand, T-lymphocytes are concerned with cellular immunity: they function by interacting with other cells. T-lymphocytes divide into CD4[1] lymphocytes or helper T-cells, and CD8 lymphocytes, called cytoxic or killer T-cells, that eliminate intracellular pathogens. Helper T-cells generally activate B-cells promoting their growth and differentiation into an antibody-secreting state. Activated B-cells cut protein antigens into smaller parts (peptides) and present them to killer T-cells. These last cells are responsible for killing virally infected cells and cells that appear abnormal.

A lymphocyte has about 10^5 receptors, which are of the same structure. In the case of B-cells, the receptor is an immunoglobulin (antibody) molecule embedded in the membrane of the cell, while in the case of T cells the receptor is simply called the T-cell receptor, or TCR. These receptors are constructed from inherited gene segments (libraries) and they come into being in the process of random recombination of segments from different libraries. The process relies upon random selection of a genetic component from each of the libraries. There are many possible combinations of the available components, so the immune system can generate a large number of antibodies even though the libraries contain a limited amount of genetic information. Additionally the libraries evolve in time. Hightower, Forrest and Perelson (1995) used this idea to simulate the ability of organising the complex structure of the antibody libraries via a genetic algorithm. This extends, in a sense, earlier work of Smith, Forrest, and Perelson (1993) and provides an interesting perspective for building pattern recognition systems. On the other hand, it can be used to improve the performance of genetic algorithms,

[1] CD is a shorthand of *cluster of differentiation*.

concretely their exploration aspect. According to Schema Theorem (Holland, 1976) the algorithm assigns exponentially increasing number of trials to the observed best parts of the search space. Treating its population individuals as libraries and equating their fitness to the fitness of (randomly generated) antibodies we obtain the population with partially expressed fitness. More precisely, the phenotype of an individual (i.e. expressed antibody molecules) does not completely represent its genotype (the total collection of gene segments in the library). Hence, best parts of the search space discovered in one cycle are rather different from best parts identified in the next cycle as random segment selection allows the segments to be temporarily hidden from selection stage of the genetic algorithm.

Clonal selection is another mechanism guaranteeing large diversity of the receptors. When a cell is activated by binding to pathogens, it secretes a soluble form of its receptors and, simultaneously, it clones itself. Clones are not perfect, but they are subjected to somatic mutation (characterized by high mutation rate) which result with children having slightly different receptors than the parent. These new B-cells can also bind to pathogens and if they have a high affinity (or simply "similarity") to the pathogens they in turn will be activated and cloned. The rate of cloning a cell is proportional to its "fitness" to the problem: fittest cells replicate the most. The somatic mutation guarantees sufficient variation of the set of clones, while selection is provided by competition for pathogens. This mechanism was employed by (Hunt and Cooke, 1996) to create a learning system, and by (Bersini and Varela, 1990) in solving optimization problems.

Clonal selection (which operates with individual) and stochastic gene selection (operating on genes that determine the specificity of antibodies) are two main mechanisms providing an exponential number of combinations. Potentially the NIS can produce 10^{15} different receptors, although an estimated number of receptors present in a body at any given time varies between 10^{8} - 10^{12}. Recognition in the NIS occurs at the molecular level and is based on the complementarity in shape between the binding site of the receptor and an epitope (a portion of the antigen). It is important to notice that since all the receptors on the surface of a single lymphocyte have the same structure, the lymphocyte can be formally treated as a single detector. It can only recognize a narrow class of structurally related epitopes.

The most popular among biologists theory is that helper T-cells are responsible for making the discrimination among self and non-self patterns. Thus, in this paper we focus on the properties of abstract helper T-cells. To detect invaders (e.g. anomalies in a system functioning) effectively we should have efficient means for generating sufficiently rich repertoire of receptors being a counterpart of T-cells receptors. The method of generating such detectors hardly depends on the rule triggering them and the method of representing genetic information.

In the rest of the paper we concentrate on a so-called binary immune system introduced in 1987 by Farmer, Packard and Perelson (cf. Section 2). Instead of a genetic alphabet with four symbols (**A**denine, **C**ytosine, **T**hymine, and **G**uanine) the model uses a binary alphabet. Both receptors and intruders are represented as binary strings of fixed length. As a rule activating a single receptor the *k*-contiguous rule is introduced in Sect. 2.1. It was proposed by (Percus, Percus and Perelson, 1993) and is widely used in anomaly detection problems - cf. (Dasgupta, 1999) or (Dasgupta and Forrest, 1996). The rule says that a receptor detects antigen if both the strings have the same bits in at least k contiguous positions. Section 2.2 is a brief overview of existing algorithms for receptors generation. Since the process of antigen recognition can be viewed as a form of template matching, in Section 2.3 the notion of templates is introduced and some elementary properties of the templates are presented.

In Section 3 the discriminative power of a single receptor is determined. Contrary to the statistical analysis, available in the literature - e.g. (Percus, Percus and Perelson, 1993) - we use deterministic approach. It allows precisely compute the number of strings detected by a single receptor. The means introduced here are also of use when the effectiveness of a set of n cooperating receptors is investigated (Section 4). This analysis gives a hint on how to construct an effective (of minimal size) repertoire of receptors and allows determine, in advance, the number of strings recognized by a given repertoire. Even if such an effective repertoire has been constructed there are still some strings that cannot be detected. D'haeseeler (1995) explained this phenomena by the existence of so-called holes, i.e. strings consisting of the templates from which the self strings were constructed. But the set of non-recognizable strings is even larger. In Section 5 this problem is studied in depth and a method enabling to compute the maximal number of strings that can be recognized by the optimal set of receptors is presented. This section ends with the verbal description of the algorithm for the optimal repertoire construction. Its details are given in (Wierzchoń, 2000).

2 Preliminaries

Let U be the set of all binary strings of length l; obviously $|U|$, the cardinality of U, equals 2^l. Let $S \subseteq U$ be a proper subset of U, called self strings, which represent e.g. regular states of a system. The strings from the set U-S are referred to as non-self strings. The problem relies upon constructing a set of detectors, denoted R, such that each $r \in R$ doesn't recognize any self string, and at least one receptor activates when meeting a non-string representing abnormal state of the system. This way of detecting abnormal states was proposed by Forrest *et. al.* (1994) under the name negative selection method. It has a number of interesting features distinguishing it from other methods. The most important, among them, are:

1. No prior knowledge of anomaly is requested.
2. Detection is probabilistic and tuneable: instead of constructing a set of detectors recognizing all non-self strings (so-called complete repertoire) a smaller set of detectors is generated. It recognizes all but a small fraction P_f of non-self strings in exchange for a smaller set of detectors.
3. Detection is local: only small sections of data are checked and when a detector does find an anomaly it can be localized to the string that the detector is checking.
4. Detection is distributable: small sections of the protected system can be checked separately and no communication among detectors is needed until an anomaly is detected.

Observe that the strings from R can be loosely treated as a concise description of a notion N described by the strings belonging to the set U-S. Denoting by R^* the set of strings detected by the receptors in R, the problem can be stated as follows: knowing the description on non-N, given by a set $S \subseteq U$, find a subset $R \subseteq U$-S of minimal cardinality such that $R^* = U$-S. Here, typically, the cardinality of S is relatively small in comparison with the cardinality of U. This is in contrast to the earlier work of Smith, Forrest, and Perelson (1993) where the set S was not taken into account explicitly and the number of antigens was relatively small.

To implement the algorithm of identifying the set R we should define in general: receptors representation (binary in our case), the method of their activation (so-called matching rule), and the method of receptors generation. These topics are discussed below.

2.1 Matching rules

There is no unique receptors activation method. Perhaps a simpler one is Hamming matching: two strings x and y match under the rule if they have different bits in at least k positions, where $1 \le k \le l$, i.e.

$$\texttt{match}_H(x,y) \text{ iff } d_H(x,y) \ge k$$

where d_H stands for the Hamming distance. It is easy to observe that $\texttt{match}_H(x,y)$ is symmetric and irreflexive since $d_H(x,y) = d_H(y,x)$ and $d_H(x,x) = 0$.

The number of strings that are exactly d bits from an arbitrary string $x \in U$ is the number of ways to choose d coordinates from a total of l coordinates, and is therefore given by the binomial coefficient (consult e.g. (Kanerva 1988))

$$|\{y \in U: d_H(x,y) = d\}| = \binom{l}{d}$$

Hence, the total number of strings recognized by a single receptor $r \in R$ under the Hamming match with threshold k, $D_H(l,k)$, equals

$$D_H(l,k) = \sum_{i=k}^{l} \binom{l}{i}$$

Knowing this number we easily find $p_H(l,k)$ - the probability that two random strings match at least k bits: $p_H(l,k) = 2^{-l} \cdot D_H(l,k)$.

Other rules based on Hamming match are reviewed in (Hunt and Cooke, 1996).
In this paper we will focus on so-called k-contiguous bits rule introduced by Percus *et. al*. (1993) as a plausible abstraction of receptor binding in the immune system. Two strings, x and y match under the rule if x and y have the same bits in at least k contiguous positions. Suppose for instance that $l = 6$, $k = 3$ and assume that the strings r (receptor) and x_1, x_2 are of the form $r = 100110$, $x_1 = 001100$ and $x_2 = 000100$. Then $\mathsf{match}_C(r,x_1)$ = FALSE while $\mathsf{match}_C(r,x_2)$ = TRUE, so x_1 is a self pattern and x_2 is an antigen (anomaly). The rule can be imagined as moving a window of width k over two tested strings:

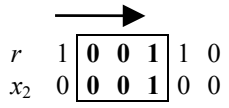

Figure 1. Matching under the k-contiguous rule: move from left to right a window of length k over the receptor (r) and tested (x) strings. If the two substrings within the window are identical, receptor activates.

Contrary to the Hamming match this rule is symmetric and reflexive and hence induces a tolerance relation over U-S. The discriminative power of this rule is investigated in the next sections.

2.2 Existing algorithms for receptors generation

Given a binary immune system with the set S of self strings we should generate the set R containing as small as possible receptors which recognize all the antibodies from U - S. Such a set R is referred to as the complete repertoire. Usually, to reduce its cardinality we are satisfied with a subset of R recognizing all but a small fraction P_f of non-self strings. D'haeseler (1995) proved that in general it is impossible to construct a set R recognizing all non-self strings. The set U - S contains so-called holes i.e. strings constructed from the templates (defined in

Section 2.3) of S which are not members of S. Wierzchoń (2000) has shown that U - S contains additional number of non-recognizable strings: they contain some templates characterizing S and cannot be complemented with templates characterizing strings from U - S.

A naive solution to the problem of receptor repertoire construction is to generate randomly candidate strings and then test them to see if they match any self string. If a match is found, the candidate is rejected. This process is repeated until the desired number of receptors is generated. This algorithm resembles the way in which B-cells in the immune system are recruited in the bone marrow. It is ineffective, however, because the receptors grow exponentially with the size of $|S|$. Denoting P_m the probability that two random strings match at least k contiguous positions and P_f - the probability that the set of receptors fail to detect an antigen, the time complexity of this algorithm is $O(-\ln(P_f)\cdot|S|/(P_m\cdot(1-P_m)^{|S|})$ and the space complexity is $O(l\cdot|S|)$, consult (D'haeseleer, Forrest, and Helman, 1996).

Helman and Forrest (1994), proposed a more efficient algorithm which runs in linear time with the size of self (and receptors). It consists of two stages. First, the set of templates (defined below) from which receptors can be constructed is identified, and numbering of the templates is established. Next, this numbering is used to construct randomly the receptors. Unfortunately, this way we obtain many redundant receptors. D'haeseleer (1995) proposed a greedy algorithm based on the same principles which generates better coverage of the string space by placing detectors as far apart as possible. However, its space complexity of this algorithm is of order $O((l - k)^2\cdot 2^k$.

Further simplification of this idea has been proposed by (Wierzchoń, 1999), who observed that the templates used to construct receptors form binary trees. Identifying common subtrees in these trees we can substantially reduce the number of receptors.

2.3 Templates

Moving the window of width k over the self strings, see Fig. 1, we can split each of them into (l-k+1) substrings of length k. These substrings induce so-called templates introduced by (Hellman and Forrest, 1994) to build receptors. Since each receptor does not recognize any self string, $s \in S$, it is obvious that it cannot contain any template recognized in a self string.

To be more precise, let w be a binary string of length k (k is the threshold value). We will consider strings of length l over the alphabet {0,1,*} where * stands for „irrelevant". By a template $t_{i,w}$ of order k, we understand a string (of length l), whose substring of length k taken from position i equals w, and all the remaining positions of the template are filled by the star symbol. For instance, when l = 6, k

= 3, and w = 010 then $t_{1,w}$ = 010***, $t_{2,w}$ = *010**, $t_{3,w}$ = **010*, and $t_{4,w}$ = ***010. A self string s = 001101 splits into four templates: $t_{1,001}$ = 001***, $t_{2,011}$ = *011**, $t_{3,110}$ = **110*, and $t_{4,101}$ = ***101. In genetic algorithms terminology a template of order k is a schema (Holland, 1975) of order[2] k in which all the significant bits are contiguous.

The set of all possible templates, denoted T, contains $(l - k + 1)\cdot 2^k$ different elements. We split T into two disjoint subsets: T_S consisting of all the templates contained in at least one self string and the set of remaining templates, T_N, used to construct receptor strings. Typically T_S is a low fraction of T. Following D'haeseleer (1995) we will naively[3] represent the set T as the matrix T with 2^k rows and $(l - k + 1)$ columns: T[w,i] = 0 if $t_{i,w} \in T_S$ and T[w,i] = 1 if $t_{i,w} \in T_N$.

Example 1: Let l = 6, k = 3. Given the set of ten self strings, shown in the leftmost column of Table 1, the sets T_S (of self templates) and T_N (of non-self templates) are represented by the table T consisting of 8 rows and 4 columns shown below.

Table 1. Matrix T representing the set T_S (of self) and T_N (of non-self) templates

S
001110
001101
001111
010001
010101
011100
011111
100001
110001
110100

no	w	T[w,1]	T[w,2]	T[w,3]	T[w,4]
0	000	1	0	0	1
1	001	0	1	1	0
2	010	0	1	0	1
3	011	0	0	1	1
4	100	0	0	1	0
5	101	1	0	1	0
6	110	0	1	0	0
7	111	1	0	0	0

3. Discriminative Power of a Receptor

Consider a single receptor $r = b_1 b_2,...,b_l$ where $b_i \in \{0,1\}$ denotes bit value at i-th position, i = 1,...,l. We are interested in finding the number $D(l, k)$ of unique

[2] The order of a schema is defined as the number of relevant positions in this schema. For instance if x_1 = 0000**** and x_2 = 00000*** then $order(x_1)$ = 4 and $order(x_2)$=5.

[3] Linked lists or sparse arrays are more efficient representations. We use the matrix representation for its illustrative power only.

strings from U detected (by means of the k-contiguous-bits rule) by the receptor r. Obviously this number depends on the receptor length and the threshold value only. To find $D(l, k)$ we will represent all the templates $t_{i,w}$ constituing a given receptor by the set of schemas forming a partition of the set of all detected strings. In other words, if $X = \{x_1,...,x_m\}$ is the set of schemas generated by the receptor and u is an antibody detected by r then u is an instance[4] of exactly one schema $x_i \in X$. We restrict to the special class of schemas, however: a schema derived from a template $t_{i,w}$ has first $(k+i-1)$ positions meaningful and remaining $(l-k-i+1)$ positions are filled in by the star symbol. Hence, such a schema covers $2^{l-k-i+1}$ strings. To find the number $D(l, k)$ we consider two cases: a simpler one when $k \geq (l/2)$ and more complicated case when $k < (l/2)$.

3. 1 The threshold value $k \geq (l/2)$

In this case the number $D(l, k)$ can be found by counting the number of schemas generated by the templates $t_{i,w}$, $i = 1,...,l-k+1$. The template $t_{1,u}$, where $u = b_1b_2,...,b_k$, detects strings that agree with the schema $b_1b_2...b_k*...*$ containing $(l-k)$ star symbols. The template $t_{2,v}$, where $v = b_2b_3,...,b_{k+1}$, detects strings agreeing with the schema $*b_2b_3...b_{k+1}*...*$ containing $(l-k-1)$ stars. According to our convention this schema divides into two schemas: $b_1b_2b_3...b_{k+1}*...*$ and $(1-b_1)b_2b_3...b_{k+1}*...*$, where $(1-b_1)$ stands for the complement of b_1. The first schema is an instance of the schema induced by the template $t_{1,u}$; hence only second schema is *fresh*, i.e. it recognizes new strings. Similarly, the template $t_{3,w}$, where $w = b_3b_4,...,b_{k+1}b_{k+2}$ splits into four schemas: $b_1b_2...b_kb_{k+1}b_{k+2}$ *...*, $(1-b_1)b_2...b_kb_{k+1}b_{k+2}$ *...*, $b_1(1-b_2)...b_kb_{k+1}b_{k+2}$ *...*, and $(1-b_1)(1-b_2)...b_kb_{k+1}b_{k+2}$ *...*. The first schema is an instance of the schema generated by the first template and the second schema is an instance of a schema generated by the second template. Thus only third and fourth schemas are fresh. To list all fresh schemas generated by all the templates contained in a receptor r, we proceed as follows.

a) Put on a first position of a list the schema induced by the first template.
b) Let current length of the list, c_1, equals 1
c) For any template $t_{i,w}$ (i=2,..., $l-k$+1) do the following
d) For $j = 1$ to c_{i-1} copy j-th schema from the list to $(c+j)$-th position and replace $(i-1)$-th bit in the schema by its complement, $b_{i-1} \leftarrow 1-b_{i-1}$
e) Modify current length of the list: $c_i \leftarrow c_{i-1} + c_{i-1} = 2^{i-1}$.

Table 2 shows the result of this procedure for four initial templates.

Observe that each template $t_{i,w}$, (i=2,..., $l-k$+1), divides into 2^{i-1} schemas (because it contains i-1 leading star symbols) and only half of them is fresh. Thus i-th template ($i \geq 2$) generates 2^{i-2} new schemas and each of them covers $2^{l-k-i+1}$

[4] That is, if x = 00000*** then e.g. u=00000101 is an instance of x.

different strings (since each schema contains l-k-i+1 star symbols). In summary, the first template covers 2^{l-k} strings and any other template $t_{i,w}$, i = 2,...,l-k+1, covers $2^{i-2}\cdot 2^{l-k-i+1} = 2^{l-k-i+1}$ different strings. The total number of strings recognized by a receptor equals

$$D(l,k) = 2^{l-k} + (l-k)\cdot 2^{l-k-1} = 2^{l-k-1}\cdot(l-k+2) \quad (1)$$

Table 2. Fresh (unique) schemas generated by four initial templates

Temp-late	substring generating template	Schemas generated by the template
$T_{1,u}$	$u=b_1b_2,...,b_k$	b_1 b_2 b_3 b_4 ... b_k * * * * ... *
$T_{2,v}$	$v=b_2,...,b_{k+1}$	1-b_1 b_2 b_3 b_4 ... b_k b_{k+1} * * * ... *
$T_{3,w}$	$w=b_3,...,b_{k+2}$	b_1 1-b_2 b_3 b_4 ... b_k b_{k+1} b_{k+2} * * ... *
		1-b_1 1-b_2 b_3 b_4 ... b_k b_{k+1} b_{k+2} * * ... *
$T_{4,x}$	$x=b_4,...,b_{k+3}$	b_1 b_2 1-b_3 b_4 ... b_k b_{k+1} b_{k+2} b_{k+3} * ... *
		1-b_1 b_2 1-b_3 b_4 ... b_k b_{k+1} b_{k+2} b_{k+3} * ... *
		b_1 1-b_2 1-b_3 b_4 ... b_k b_{k+1} b_{k+2} b_{k+3} * ... *
		1-b_1 1-b_2 1-b_3 b_4 ... b_k b_{k+1} b_{k+2} b_{k+3} * ... *

3. 2 The threshold value $k < (l/2)$

This case is more complicated and in fact we must consider two situations: (a) k is close to (l/2), and (b) k is close to 1.

3.2.1 The threshold value is close to (l/2)

The procedure described in previous subsection pretty works for i = 1,...,k+1. Suppose now $i = k+2$ and $k < l/2$. Then by step (d) of our procedure we must change (k+1)-th bit in all schemas belonging to the current list. But the first schema from the list has star symbol on this position. It means that we must insert empty string on (c+1)-th position since this schema has been exhausted by the first template (see Table 3, first row in the block corresponding to the template $t_{5,000}$). Now if $i = k+3$, both the first and second schema from the list has star symbol on (k+2)-th position. Further, current list contains empty string already introduced when the template $t_{k+2,w}$ was converted into fresh schemas. Thus, when developing the template $t_{k+3,w}$ we must insert 2+1 empty strings to the list. In general, the number of empty strings introduced by i-th template, α_i, equals

$$\alpha_i = 2^{i-k-2} + \beta_i,\ i = k+2,...,l-k+1 \quad (2a)$$

where

$$\beta_j = \alpha_{k+2} + ... + \alpha_{j-1} \tag{2b}$$

is the number of empty strings already introduced when previous templates have been developed. Obviously, if $i < k+2$ then no empty strings occur on the list and $\alpha_i = 0$. Table 4 shows the values of α_i and β_i for $i = k+2,\ldots,k+10$.

Table 3. Fresh schemas induced by the receptor 00000000 with threshold $k = 3$

template	schema		template	schema
$t_{1,000}$	000*****		$t_{6,000}$	empty
$t_{2,000}$	1000****			empty
$t_{3,000}$	01000***			01001000
	11000***			11001000
$t_{4,000}$	001000**			00101000
	101000**			10101000
	011000**			01101000
	111000**			11101000
$t_{5,000}$	empty			empty
	1001000*			10011000
	0101000*			01011000
	1101000*			11011000
	0011000*			00111000
	1011000*			10111000
	0111000*			01111000
	1111000*			11111000

Empty strings reduce the number of strings recognized by the receptor. The number, *red*, which decreases $D(l,k)$ is computed according to the formula

$$red = \sum_{j=k+2}^{l-k+1} \alpha_j \cdot 2^{l-k-j+1} \tag{3}$$

3.2.1 The threshold value is close to 1

When k is relatively small, the number of reduced strings must be slightly modified. To explain this consider Table 5, where $l = 8$, $k = 2$ (note that the schemas generated by the template $t_{7,00}$ are displayed in two last columns). The number of redundant schemas for the templates $t_{4,00},\ldots, t_{6,00}$ can be computed

according to the equations (2a) and (2b). When converting the template $t_{7,00}$ we must copy, according to step (d) of our procedure, $c_6 = 2^5$ positions from the current list and replace 6-th bit in each receptor by its complement. To find the number *red*, as described in previous subsection, the first $2^{7-k-2} = 2^3$ positions added to the current list must be empty and next we must count empty strings already introduced when previous templates have been converted into schemata.
Observe however, that the first empty string occurs on 5-th position in our list – it's a result of conversion $t_{k+2} = t_{4,00}$ template into fresh schemas. Hence the equation (2b) will count this empty strings two times. To avoid such multiple counting define $j^* \in [k+2,\ldots i-1]$ to be maximal integer such that

$$2^{i-k-2} > 1+c_{j^*-1} = 1+2^{j^*-2} \tag{4}$$

Table 4. Values of α_i and β_i defined in equations (2a), (2b)

	α_i	β_I
k+2	1	0
k+3	3	1
k+4	8	4
k+5	20	12
k+6	48	32
k+7	112	80
k+8	256	192
k+9	578	448
k+10	1280	1024

If such an integer exists, then reduce the number *red* by the value β_{j^*+1}. Let us explain this modification. When a template $t_{i,w}$ is transformed into schemas, the list length is $c_{i-1} = 2^{i-2}$, and the first new schema is inserted into position c_{i-1} +1. This explains right hand side of equation (4). As observed in subsection 3.2.1 a template $t_{i,w}$, $i \geq k+2$, introduces 2^{i-k-2} empty strings to the list. So we must test if the newly created empty strings cover existing strings. This explains the idea underlying equation (3). The number of covered strings is $\alpha_{j^*} = 2^{j^*-k-2} + \beta_{j^*} = \beta_{j^*+1}$. Hence when such a j^* has been identified we must subtract β_{j^*+1} schemas from *rel*. Example 1 below illustrates how to modify the values of α_i.

Table 5. Fresh schemas induced by the receptor 00000000 with $k = 2$. Schemas induced by the template $t_{7,00}$ are shown in two last columns.

template	schema	template	Schema	template	Schema	schema
$t_{1,00}$	00******	$t_{6,00}$	Empty	$t_{7,00}$	Empty	empty
$t_{2,00}$	100*****		Empty		Empty	empty
$t_{3,00}$	0100****		Empty		Empty	empty
	1100****		Empty		Empty	empty
$t_{4,00}$	Empty		Empty		Empty	empty
	10100***		1010100*		Empty	10101100
	01100***		0110100*		Empty	01101100
	11100***		1110100*		Empty	11101100
$t_{5,00}$	empty		Empty		Empty	empty
	empty		Empty		Empty	empty
	010100**		0101100*		01010100	01011100
	110100**		1101100*		11010100	11011100
	empty		Empty		Empty	empty
	101100**		1011100*		10110100	10111100
	011100**		0111100*		01110100	01111100
	111100**		1111100*		11110100	11111100

Example 2. In both cases presented below we assume that the receptor consists of l 0's.

(a) If $l = 8$ and $k = 2$, then j^* must be not less than $k+2 = 4$. When developing the template $t_{6,00}$ we have $2^{6-2-2} < 1 + 2^{4-2}$. But for the template $t_{7,00}$ we have $2^{7-2-2} > 1 + 2^{4-2}$ and $2^{7-2-2} < 1 + 2^{5-2}$ what correctly shows that we must reduce *red* by $\beta_{4+1} = \beta_{(k+2)+1} = 1$ schemata.

(b) Let $l = 13$ and $k = 2$, then $j^* \geq 4$. For the template $t_{8,00}$ we have $j^* = 5$ and $\beta_{5+1} = \beta_{(k+2)+2} = 4$ while for the template $t_{9,00}$ we have $j^* = 6$ and $\beta_{6+1} = \beta_{(k+2)+3} = 12$.

Another interesting case is that of $k = 1$. Assume, as previously, that r consists of l 0's. The first empty string occurs when the template $t_{3,0}$ is developed into schemas (Since this template divides into two fresh schemas, it enters only one fresh schema in this situation. The next template enters two empty strings and copies the already existing empty strings. It means that among four fresh schemata three of them are empty and this template introduces only one fresh schema. For the fifth template (entering 8 fresh schemata) the inequality (3) is satisfied with $j^* = 3$ what means that - according to (2a), (2b) - $\alpha_5 = \alpha_{k+4} = 8$ schemas are redundant and - according to (3) this number must be reduced by $\beta_{j^*+1} = \beta_4 = \beta_{1+3} = 1$. In summary, fifth template enters only one fresh schema again. Continuing this line of reasoning we state that each template enters exactly one fresh schema of the form

1...10*..*. For i-th template the corresponding schema contains (i - 1) 1's, one symbol '0' at i-th position, and (l - i) star symbols. Hence the number of strings recognized by a receptor of length l with threshold value k = 1 equals $D(l, 1) = \sum_{i=1}^{l} 2^{l-i} = 2^l\text{-}1$.

This last case can be redefined as: Find the cardinality of the set $O = \{u \in U: d_H(x,y) \le l - 1\}$. Equivalently, noting that the set O is a circle in Hamming space with radius l-1 and center at $x \in U$, the number $D(l, 1)$ is just its area, cf. (Kanerva, 1988). From this observation it follows also that $D(l, 0) = 2^l$.

3.3 General recipe

A procedure `FindStringsNumber`, given below, summarizes our considerations. To save time and space, the values of β_i are stored in the table `btab`. Since $\beta_i = 0$ for $i = 1,\ldots, k+2$, we can store relevant values only, that is we renumber (k+2)-th position as 0, (k+3)-th position as 1, an so on.

Knowing $D(l,k)$ we can compute $p(l,k)$, the probability that a randomly chosen string $u \in U$ matches with a receptor (i.e. that u is an antigen): $p(l,k) = D(l,k)/2^l$. When $k \ge (l/2)$ then

$$p(l,k) = 2^{l-k-1}\cdot(l - k + 2)/2^l = 2^{-k}\cdot[(l - k)/2 + 1] \qquad (5)$$

This formula was derived by Perelson using binomial distribution and with additional requirement that $2^{-k} << 1$. From our analysis it follows that (4) is valid when $k \ge (l/2)$ and no additional requirement is requested. Figure 2 shows how the number of strings recognized by a single receptor changes for different values of l and k.

```
Function FindStringsNumber
Input: l, k.
Output: D = the number of strings recognized by a receptor.
Auxiliary variables and arrays:
i, j: integer; {counters}
α, β, red,tmp: integer;
btab: array [0..l-k-2] of integer;
begin
    D = 2^(l-k-1)·(l-k+2);
    if (k ≤ (l div 2)) then
    begin
        α:=0; β:=0; red:=0;
        for i:=k+2 to (l-k+1) do
```

```
        begin
            α:=2^(i-k-2);
            for j:=k+2 to (i-1) do {identify j* and β_(j*+1)}
            if (α>(1+2^(j-2))) then sum:=btab[j-k-2];
            α:=α+β-sum;
            β:=β+α;
            btab[i-r-2]:=b;
            red:=red+α*2^(l-i-k+1);
        end;
        D := D – red;
    end;
    return D;
end.
```

The reasoning presented here easily extends to the case when strings over an alphabet consisting of m symbols are considered. For instance when $k \geq (l/2)$ we observe that each template $t_{i,w}$ introduces $(m\text{-}1)/m$ fresh schemata. Hence the total number os strings recognized by a single receptor equals

$$D_m(l,k) = m^{l-k} + (l\text{-}k)\cdot(m\text{-}1)\cdot m^{l-k-1} = m^{l-k-1}\cdot[(l\text{-}k)\cdot(m\text{-}1) + m]$$

Dividing this number by m^l we obtain the formula given in (Percus, Percus, Perelson, 1993)

4. Reduction of the Discriminative Power

Although a single receptor can distinguish $D(l,k)$ unique strings from the universe U, its discriminative power radically changes when it cooperates with another receptors. To be more illustrative consider two receptors `000000` and `001100`, and assume that the threshold $k = 3$. Using the method described in previous section we easily state that both the receptors recognize 38 unique strings and not $2\cdot D(6,3) = 40$ strings. On the other hand, the ensemble consisting of two receptors `000000` and `100001` recognizes only 28 receptors.

To explain this phenomena let us write down all the schemas induced by the templates representing corresponding strings. In the first case (case (a) in Table 6) we see that the schema `000100`, belonging to the template $t_{4,100}$ of the string `001100` is absorbed by the schema `000***` representing the template $t_{1,000}$. Similarly the schema `001000` is absorbed by the schema `001***`. Hence, instead of 16 fresh schemas we have only 14 fresh schemas. In the second case

(case (b) in Table 6) we have only 12 fresh schemas covering 28 different strings, that is each receptor recognizes 14 strings in average.

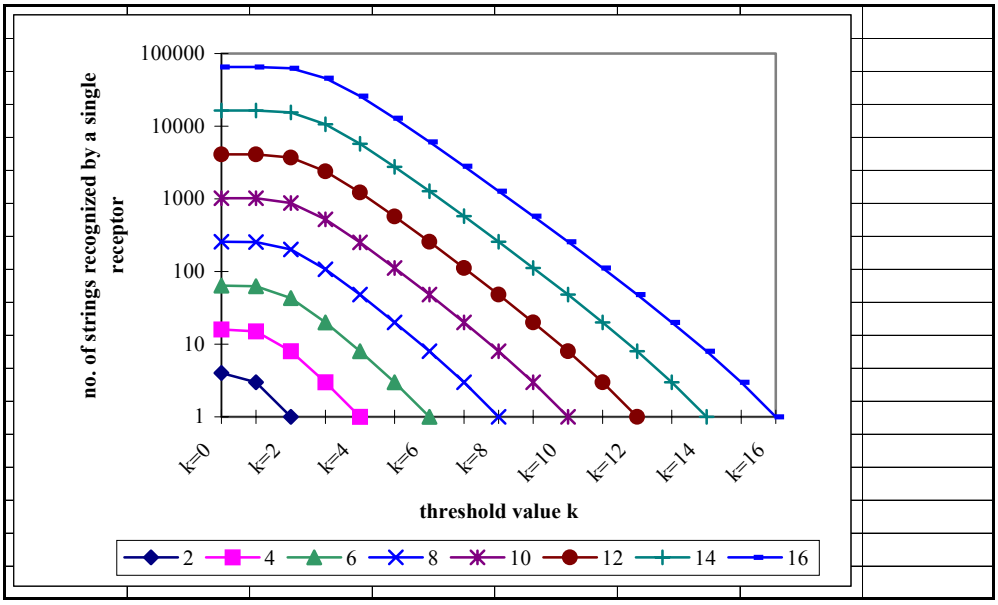

Figure 2. Number of strings detected by a single receptor

In general, to estimate the average number of strings recognized by a set of n receptors we should use statistical approach. Assuming that the receptors are chosen independently we can define $p_f(l,k,n)$, the failure probability

$$p_f(l,k,n) = (1 - p(l,k))^n \approx e^{-n \cdot p(l,k)} \tag{6}$$

This last approximation is valid for large values of n and small values of $p(l,k)$. The average number of strings detected by n receptors is determined by the formula

$$d(l,k,n) = (1 - p_f(l,k,n)) \cdot 2^l \tag{7}$$

and the average number strings detected by a single receptor among the ensemble of cardinality n equals $d_{avg}(l,k,n) = d(l,k,n)/n$. Figure 3 shows how this number varies for different values of l, k and n. The parameters l and k were chosen such $D(l, k)$ is fixed and equals 48. Each receptor can be treated as a „ball" in its Hamming space. Increasing l we increase the „volume" of the space, and the larger the space, the balls have more places and can freely move without losing

theirs independence.

Table 6. Overlapping schemas in the ensemble of two strings

Case (a)				case (b)		
Schemas induced by the string 000000		Schemas induced by the string 001100		Schemas induced by the string 000000		Schemas induced by the string 100001
100***		001***		000***		100***
1000**		1011**		~~1000**~~		~~0000**~~
01000*		01110*		01000*		~~11000*~~
11000*		11110*		11000*		~~01000*~~
~~001000~~		~~000100~~		001000		101001
101000		100100		101000		001001
011000		010100		011000		111001
111000		110100		111000		011001

Formula (7) almost perfectly agrees with empirical data. Figure 4 shows the theoretical curve compared with real data. The plot was averaged over 200 runs, and average of these runs is almost identical (hence, not shown) with theoretical values. However, we see that apart from mean values there are two extreme lines: upper one shows the best results achieved in these runs, while the lower line shows the worst results. It is interesting to contrast the number of strings recognized by a receptor with the average number of fresh schemas included in the receptor. Figure 4 shows how this number decreases when the size of the set of receptors increases (again the plot was averaged over 200 runs). A careful examination of both the plots shows that choosing receptors that contain as most as possible different templates, we can increase their discriminative power.

These considerations give a hint on how to construct a repertoire of receptors. Namely, to achieve maximal discrimination power of the receptors we should choose them in such a way that each receptor enters maximal number of different schemas. To achieve this receptors must be build from diverse templates. This problem is discussed in the next section.

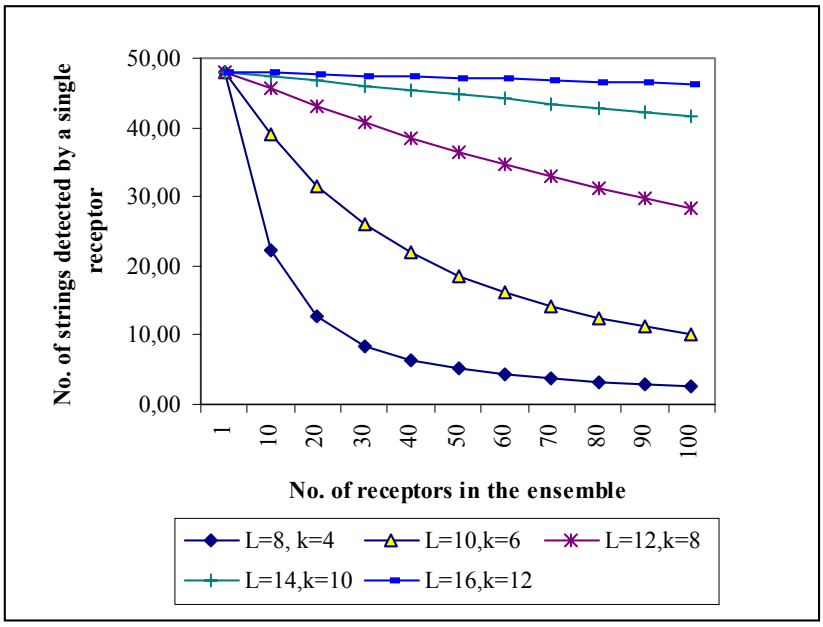

Figure 3. Reduction of the discriminative power of a single receptor

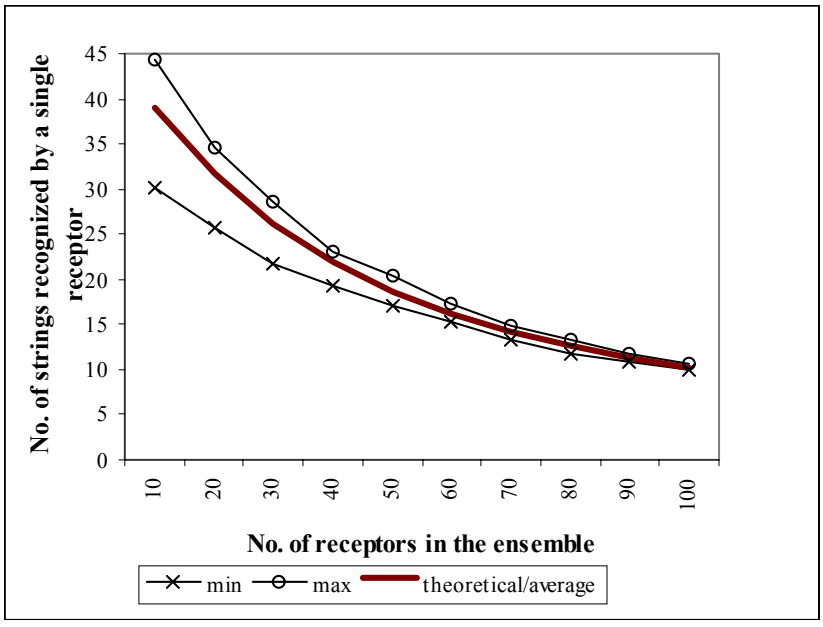

Figure 4. Average number of strings recognized by n receptors (l=10, k=6).

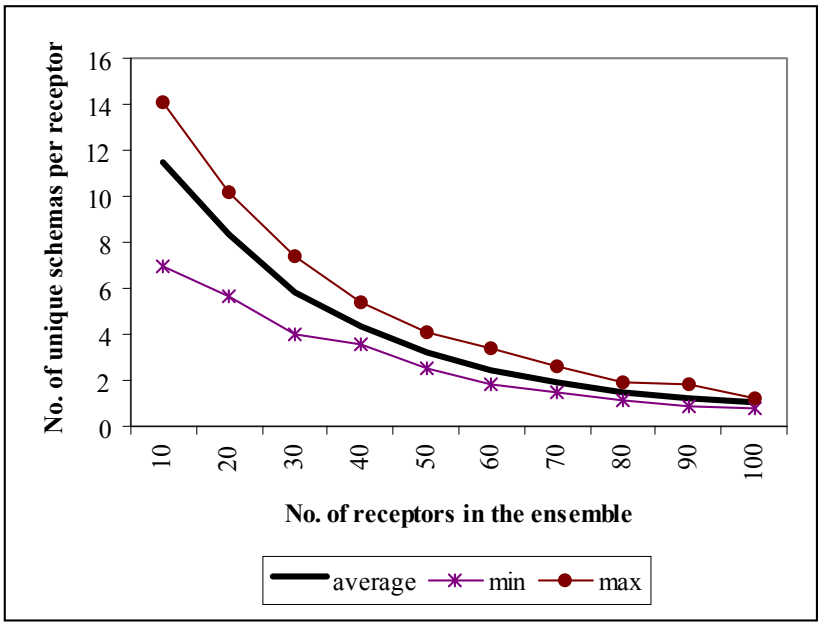

Figure 5. Average number of schemas included in a single receptor (l=10, k=6).

5. Lower Bound for the Fault Probability

The fault probability $p_f(l,k,n)$ defined in formula (6) applies to the case when $S = \varnothing$. When $S \neq \varnothing$ it is not possible, in general, to construct detectors recognizing all the strings from the set U-S. It is hard to define the lower bound analytically, but it is relatively easy to treat the problem numerically. There are two sources of non-detectability, which will be discussed below.

The first source are so-called holes defined by D'haeseleer (1995). Intuitively by hole we understand any string $u \in U$-S build up from the templates belonging to the set T_S only. There is a simple procedure, suggested by Wierzchoń (1999), to count the number of holes. Before recalling it let us introduce some useful notions, however.

Let w be a substring of length k. Denote by $(\rightarrow w)$ the substring obtained by deleting the first bit from w, and by $(w\leftarrow)$ the substring resulted from deletion of last bit from w. The symbol $(\rightarrow w)+b$ denotes the string $(\rightarrow w)$ appended with b, where $b \in \{0,1\}$, and similarly $b+(w\leftarrow)$ denotes b appended with $(w\leftarrow)$; obviously in both cases the length of new strings is again k.

Wierzchoń (1999) observed further that the self strings can be represented as binary trees whose nodes correspond to the templates: a template $t_{1,w}$ is said to be the root of a tree. In general, given a template $t_{i,w} \in T_S$, $1 \le i \le (l\text{-}k)$, we call te template $t_{i+1,(\rightarrow w)+0} \in T_S$ the left child of $t_{i,w}$, and the template $t_{i+1,(\rightarrow w)+1} \in T_S$ the right child of $t_{i,w}$. Surely, if $i + 1 = l\text{-}k + 1$ then the corresponding child is just a leaf of the binary tree. By analogy, given a template $t_{i,w} \in T_S$, $2 \le i \le (l\text{-}k+1$, we call te template $t_{i\text{-}1,(w\leftarrow)+0} \in T_S$ the left child of $t_{i,w}$, and the template $t_{i\text{-}1,(w\leftarrow)+1} \in T_S$ the right child of $t_{i,w}$.

Figure 6 shows binary trees representing the set of self strings from Example 1. The trees are drawn in a compact way: common subtrees were identified and joined together. For instance the leftmost structure represents two binary trees with the roots $t_{1,001}$ and $t_{1,011}$; both the trees have common subtrees: one rooted with $t_{3,110}$, and second rooted with $t_{3,111}$. Now any path from the root to a leaf represents a single string. Careful examination of the structure shows that we can reconstruct 15 strings instead of the original $10 = |S|$ strings. It means that we can construct 5 additional strings from the templates belonging to the set T_S. These additional strings are just holes. For instance the leftmost structure encodes four strings: `001100`, `001101`, `001110` and `001111`. Two of them are holes: `001100`, `001110`.

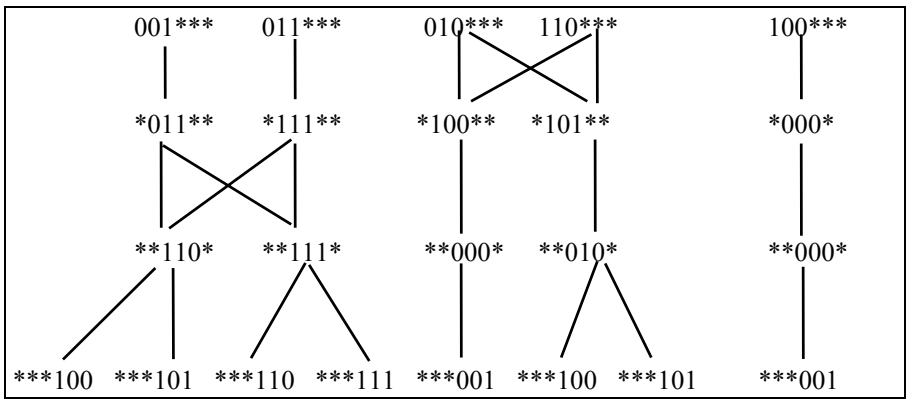

Figure 6. Graphical representation of the templates from the set T_S

We are ready now to define a simple procedure counting the number of strings induced by the set S. Suppose we move from the leaves toward the root of a tree. A node $t_{l\text{-}k,w} \in T_S$ has at most two children: $t_{l\text{-}k+1,(\rightarrow w)+0}$ and $t_{l\text{-}k+1,(\rightarrow w)+1}$. If this is the case, it means that we can construct two self strings: one ends with bit 0 and the second ends with 1. Suppose the values of the table T (defined in Section 2.3)

were changed such that T[w,i] = 1 - T[w,i], i.e. T[w,i] = 1 iff $t_{i,w} \in T_S$. Hence the values of the table T should be updated according to the rule:

$$T[w,i] = T[(\rightarrow w)+0,i+1] + T[(\rightarrow w)+1,i+1] \text{ , } i = (l-k),...,1$$

provided that $t_{i+1,(\rightarrow w)+0}$ and $t_{i+1,(\rightarrow w)+1}$ are members of T_S. Summing up all the entries in the first column of the table T we find the number N_S of all possible strings that can be constructed from the templates belonging to T_S. Now, N_S - $|S|$ is the number of holes.

Let us focus now on the second source of non-detectability. Suppose we construct receptors from the template belonging to T_N. Consider the template $t_{1,000}$ from Table 1 of Example 1. Its left child $t_{2,000}$ belongs to the set T_S and its right children $t_{2,001}$ belongs to the set T_N; hence the initial four bits of possible receptor are 0001. Now the template $t_{2,001}$ has only one valid (i.e. belonging to T_N) child: $t_{3,011}$. But both the children of this last template, i.e. $t_{4,110}$ and $t_{4,111}$ belong to T_S what means that $t_{1,000}$ cannot generate a valid receptor. Wierzchoń (2000) proposed a simple procedure for finding the set $T_R \subseteq T_N$ from which we can build receptors. Roughly speaking for each template $t_{i,w} \in T_S$ we check its parents: if a parent, say $t_{i-1,v}$ is not a member of T_S, but both its children are members of T_S, we move $t_{i-1,v}$ to the set T_S. Similarly, we check the children of $t_{i,w} \in T_S$: if a child, say $t_{i+1,r}$ is not a member of T_S, but both its parents are members of T_S, we move $t_{i+1,r}$ to the set T_S. Let us call such a procedure `FindIneffective`. Table 7 presents modified (by the `FindIneffective` procedure) Table 1 from Example 1.

Table 7. Matrix T representing modified set T_S (added templates are in bold) and the set T_R of templates from which receptors can be generated

S
001110
001101
001111
010001
010101
011100
011111
100001
110001
110100

no	w	T[w,1]	T[w,2]	T[w,3]	T[w,4]
0	000	**0**	0	0	1
1	001	0	**0**	**0**	0
2	010	0	1	0	1
3	011	0	0	**0**	1
4	100	0	0	1	0
5	101	1	0	1	0
6	110	0	1	0	0
7	111	1	0	0	0

Now, counting the number of strings induced by the modified set $T_S' = T - T_R$ we determine the whole number of nondetectable strings. It consists of: the number of self strings, the number of holes, and the number of additional nondetectable strings. In our example we find that the number of strings induced by T_S' is 28 what means that the set of receptors is able to recognize only $2^6 - 28 = 36$ strings. Applying the method for identifying holes to the set T_R we find that it is possible to construct 6 different receptors shown in Figure 7.

Observe however that the three receptors chosen such that at least one of them contains the template $t_{1,101}$ and the template $t_{1,111}$ has the same discriminative power as the full set of six receptors. Suppose for instance that we decided to choose the receptors `101010`, `101011` and `111000`. Then the first and second receptors recognize templates $t_{1,101}$, $t_{2,010}$, $t_{3,101}$, $t_{4,010}$ and $t_{4,011}$ while the third detector recognizes remaining templates $t_{1,111}$, $t_{2,110}$, $t_{3,100}$, $t_{4,000}$. The general rule for constructing nonredundant receptors is such that they must cover all possible paths from roots to the leaves, and the number of these paths must be as small as possible. This problem is discussed in (Wierzchoń, 2000).

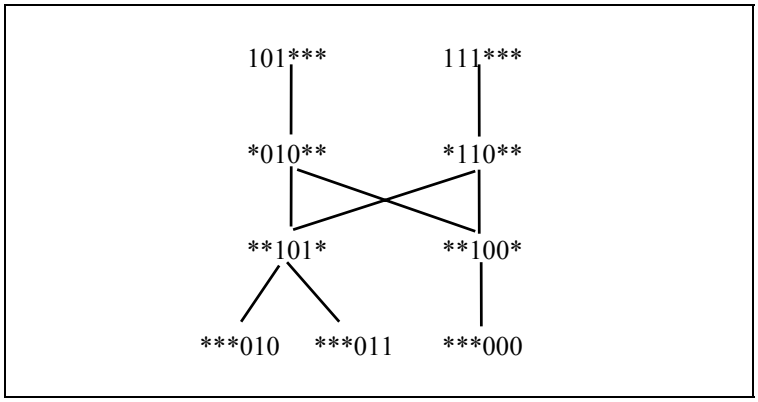

Figure 7. Graphical representation of the set of receptors to be constructed for the set T_R given in Table 7

It is interesting that the number of strings that can be detected by the complete repertoire of receptors hardly depends on the structure of the set T_S. For instance, replacing the first string of S in Example 1 by the string `001100` we are able to construct four detectors recognizing 51 strings (i.e. the number of holes is 4).

6. Summary and Conclusions

The problem of generating receptors recognizing strings from the set U-S can be treated in broader sense as a problem of inducing concise description of a notion N represented in the DNF form: each string $u \in U$-S is an elementary conjunct and the whole set U-S is the disjunction of these conjuncts. Knowing non-N, i.e. the set S, we are looking for a short (i.e. including minimal number of conjuncts) description R such that its extension R^* (expressed in terms of the matching rule) equals just U-S. The results of last section show that in general R^* is only a proper subset of U-S.

To verify our ideas we conducted a number of simulations (not reported here) and the number of holes, the total number of unrecognizable strings and the number of receptors recognizing all the remaining strings were computed by using the ideas from last section. Table 8 below presents exemplary results when S consists of 200 randomly chosen strings of length 20. Varying the threshold k we observe that the number of unrecognizable strings decreases while the number of receptors increases. This number has been compared with theoretical value dictated by the equation (6). The failure probability has been computed as the ratio of unrecognized strings to 2^l. It is interesting to observe that for $k \leq 6$ it is not possible to construct any detector. For $k = 7$ the theoretical estimate is 1,13 but since $D(20,7) = 61008$ and the number of stings that can be detected equals $2^{20} - 979584 = 68992$ it is obvious that we need at least two receptors. When k increases, the number of receptors increases approximately as the power of two. Indeed $D(l,k+1)/D(l,k) = (l-k+1)/[2\cdot(l-k+2)]$ that is the number of strings recognized by a receptor decreases approximately twice if the threshold increases by one. It is interesting however, that the method presented in Section 5 allows generate only half of the theoretical number of receptors (for larger values of k).

Table 8. Comparison of theoretical (c) and empirical (d) number of detectors needed to recognize maximal number of non-self strings. Row (a) shows number of holes and (b) shows total number of unrecognizable strings (it counts self strings as well).

k	7	8	9	10	11	12	13
(a)	122721	15349	2257	634	201	82	22
(b)	979584	107777	6999	1214	463	308	227
(c)	1.13	82.22	392.23	1150.53	2872.73	6658.34	15356.68
(d)	2	84	388	874	1868	3903	7995
(c/d)	1.7614	1.0216	0.9892	0.7596	.6503	0.5862	0.5206

It is also important to observe that for small values of k the number of holes is much lesser than the number of strings that cannot be recognized and when k increases these two numbers became almost identical. In our case when $k = 13$ the number of holes is 22 and number of additional unrecognizable strings equals 227 - 200 - 22 = 5. When $k = 14$ there is only 6 holes and 2 additional unrecognizable strings and for $k = 15$ the number of holes is 4 and there is no additional unrecognizable strings (apart of self strings of course).

In summary, the methods described in Sections 3-5 allow to:

- Count the number of holes.
- Count the number of additional strings that cannot be recognized by any set of receptors (for given threshold k). This allows correctly determine the lower bound for the failure probability.
- Count (in advance) the number of strings that can be recognized by a given repertoire of receptors.
- Generate a minimal set of receptors recognizing maximal subset of strings from the set U-S.

There is one more interesting remark. Suppose we use the random algorithm described in Section 2.2 and suppose we have found, say r receptors. Let T_r be the set of templates contained in these receptors. A new receptor can be added to the existing repertoire if (1) it does not match self strings, and (2) it enters as much as possible new templates to the T_r.

References

Bersini, H., and Varela, F.J. (1990) Hints for adaptative problem solving gleaned from immune networks. In: H.P. Schwefel and H. Muhlenbein (eds.) *Parallel Problem Solving from Nature*, LNCS 496, Springer-verlag, pp. 343-354

Dasgupta, D. (1998) *Artificial Immune Systems and Their Applications*. Springer-Verlag: Berlin Heidelberg

Dasgupta, D. (1999) Immunity-based intrusion detection systems: A general framework. In: *Proc. of the 22-nd National Information Systems Security Conference*, October 18-21.

Dasgupta, D., and Forrest, S. (1996) Novelty detection in time series data using ideas from immunology. In: *ISCA 5th International Conference on Intelligent Systems*, Reno, Nevada (also in (Dasgupta, 1998), pp. 262-267)

D'haeseleer (1995) Further efficient algorithm for generating antibody strings, Technical Report CS-95-03, The University of New Mexico, Albuquerque, NM

D'haeseleer, P., Forrest, S., and Helman, P. (1996). An immunological approach to change detection: algorithms, analysis, and implications. In *Proc. of IEEE Symposium on Research in Security and Privacy*, Oakland, CA

Forrest, S., Perelson, A.S., Allen, L., and Cherukuri, R. (1994). Self-non-self discrimination in a computer. In *Proc. of 1994 IEEE Symposium on Research in Security and Privacy,* Los Alamitos, CA: IEEE Computer Society Press

Hart, E., Ross, P., and Nelson, J. (1998) Producing robust schedules via an artificial immune system. In: *IEEE World Congress on Computational Intelligence, International Conference on Evolutionary Computing*

Hightower, R., Forrest, S., and Perelson, A.S. (1995) The evolution of emergent organization in immune system gene libraries. In: L.J. Eshelman (ed.) *Proc. of the 6-th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, CA, pp. 344-350

Hofmeyr, S.A. (1995) An overview of the immune systems. Technical Report, The University of New Mexico, Albuquerque, NM.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press: ANN Arbour.

Hunt, J.E., and Cooke, D.E. (1996) Learning using an artificial immune system. *Journal of Network and Computer Applications*, 19, 189-212

Hunt, J.E., Cooke, D.E., and Holstein, H. (1995) Case memory and retrieval based on the immune system. In: W. Weloso and A. Aamodt (eds.) *Case-Based Reasoning Research development*, LNAI 1010, Springer-Verlag, pp. 205-216

Jerne, N.K. (1974) Towards a network theory of the immune system. *Ann. Immunol*. (*Inst. Pasteur*), 125C: 373-389

Kanerva, P. (1988) *Sparse Distributed Memory*. MIT Press: Cambridge

Kephart, J.O. (1994). A biologically inspired immune system for computers. In: R.A. Brooks and P. Maes (eds.), *Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems*, Cambridge, MA, pp. 130-139.

Percus, J.K., Percus, O.E., and Perelson, A.S. (1993). Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-non-self discrimination. *Proc. Natl. Acad. Sci. USA*, 90: 1691-1695

Perelson, A.S., and Weisbuch, G. (1997) Immunology for physicists. *Reviews of Modern Physics*, 69: 1219-1265

Smith, R.E., Forrest, S., and Perelson, A.S. (1993) Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation* 1(2): 127-149

Wierzchoń, S.T. (1999) Generating antibody string in an artificial immune system. ICS PAS Report No. 892, Institute of Computer Science, Polish Academy of Sciences.

Wierzchoń, S.T. (2000) Generating optimal repertoire of antibody strings in an artificial immune system. In: M. Kłopotek, M. Michalewicz and S.T. Wierzchoń (eds.) *Proc. of 8-th Symposium on Intelligent Information Systems*, Physica-Verlag, pp. 119-133