From the Institute for Theoretical Computer Science of the
University of Lübeck

Director: Prof. Dr. Rüdiger Reischuk

Search and Learning in the Immune System: Models of Immune Surveillance and Negative Selection

Dissertation
for Fulfillment of Requirements
for the Doctoral Degree
of the University of Lübeck

From the Departments of
Computer Science and Engineering

Submitted by
Johannes Textor
from Göttingen

Lübeck, July 2011

1. Berichterstatter:	Prof. Dr. Rüdiger Reischuk
2. Berichterstatter:	Prof. Dr. Jürgen Westermann
Tag der mündlichen Prüfung:	27. September 2011

Zum Druck genehmigt. Lübeck, den 24. Mai 2012

Abstract. The immune system is crucially important for survival and development of the human species, and many fundamental processes in this highly complex system are not yet fully understood. In recent years, technological advances like two-photon microscopy have yielded substantial new information, which partially contradicts established immunological assumptions. Computational models and simulations play an increasingly important role in extracting new insights and hypotheses from these data. This thesis investigates models of two important immunological processes using techniques from theoretical computer science. A main emphasis lies on obtaining qualitative and quantitative predictions from these models, which we then go on to test against published experimental data within the context of contemporary immunological research questions. The two objectives of this approach are (1) a deeper understanding of the fundamental properties of the investigated models, and (2) new immunological insights.

First, we investigate T cell immune surveillance. Because the number of T cells that can detect a specific pathogen is very low, T cells continuously migrate *within* secondary lymphoid organs (SLOs), and travel around once daily *between* different SLOs. Whilst the inter-organ circulation has been investigated since the 1960s, it only recently became possible to observe lymphocyte migration directly in tissue. For a mathematical model that integrates these two scales, we derive the *optimal restart time:* how frequently should a T cell travel between different SLOs to ensure that localized infections are detected as quickly as possible? We validate our result using several experimental datasets, and show that the predicted optimal restart time indeed coincides with physiological values. This finding supports the hypothesis that T cell migration is near-optimal for timely antigen detection.

Second, we investigate negative selection of T cells in the thymus. Negative selection generates a T cell population that tolerates the components of its host organism (self), and attacks only foreign substances (nonself). We re-consider an existing string-based model called the *negative selection algorithm* (NSA) from the perspective of algorithmic learning theory. We discuss relations between the NSA and established learning theoretical models like inductive inference and version space learning, and show how the algorithm employs inexact pattern matching to generalize beyond input data. Moreover, we develop techniques based on string processing and data compression methods to generate predictions from the NSA in a computationally efficient manner (i.e., in polynomial time), which facilitates the analysis of large proteomic datasets in a matter of minutes. By these means, we show that a NSA based on the so-called *r-contiguous matching rule* is capable of predicting the recognition of HIV peptides by CD8 T cells. This work represents a starting point for a broader learning theoretical treatment of the immune system, which, despite being one of the two important cognitive systems in the human body, has received little attention from learning theorists so far.

Contents

Chapter 1

General Introduction

Immunology and theoretical computer science are in the author's biased opinion the two most fascinating areas of contemporary science. This thesis aims at employing tools of theoretical computer science for addressing important problems in modern immunology. Needless to say, these two disciplines could hardly differ more in their methods, styles, and cultures. The structural organization of this thesis reflects and honors these differences and is, as a result, perhaps somewhat unusual. In this introductory chapter, we explain the reasoning behind this design, which reflects the scientific method of computational biology.

The thesis consists of two independent parts. In each part, we propose a model of an important phenomenon in the immune system; then we investigate fundamental properties of the model using techniques akin to, say, the analysis of a probabilistic algorithm; and finally, we apply the model in the context of a current immunological research question by making quantitative and qualitative *predictions*, which we *validate* against published experimental data. The philosophy behind this approach, which is the foundation of mathematical and computational biology, is nicely summarized in Murray's classical textbook [110, p. xii]:

> "The art of good modelling relies on (i) a sound understanding and appreciation of the underlying problem; (ii) a realistic mathematical representation of the important biological phenomena; (iii) finding useful solutions, preferably quantitative; and what is crucially important; (iv) a biological interpretation of the mathematical results in terms of insights and predictions. The mathematics is dictated by the biology and not vice versa. Sometimes the mathematics can be very simple. Useful mathematical biology research is not judged by mathematical standards but by different and no less demanding ones."

In all due respect, we will take one of the above rules very liberally: The analysis of our models will sometimes go beyond the extent necessary for addressing the motivating biological questions. The reasons for this choice are two-fold. First, it is the author's firm belief that a prerequisite for applying a formal model to a biological question is a thorough understanding of the model's basic properties. Murray

presumably did not emphasize this aspect as his book mainly discusses models based on differential equations – these are rooted in a sophisticated, centuries-old theory, and can be considered more "standard" than the models used here which are of discrete, stochastic, and algorithmic nature. Second, formal models of biological processes also serve as a source of inspiration for computer science. Such inspiration has lead, for example, to the development of neural networks [13], genetic algorithms [73], and more recently, artificial immune systems [35]. In fact, the model analyzed in the second part of this thesis was originally intended as an immune-inspired algorithm for protecting computers and networks. Some of our results have important implications for immune-inspired computing. Most notably, our analysis in the second part solves as a by-product a problem that the field of artificial immune systems had struggled with for several years.

The two parts of this thesis follow a common structure that reflects the general approach outlined above. Each part is subdivided into two chapters. The first chapter starts out with a short, gentle introduction to the immunological subject at hand, continues with a definition of the proposed model, and then gives a detailed account of the model's formal properties. These chapters are written more in the style of a *computer science* research work; they will assume no particularly deep background knowledge in immunology, but familiarity with probability and complexity theory is presupposed. The analysis concludes with a short epilogue outlining potential uses in computer science, which we do not, however, explore in detail. The second chapter will then state and investigate a contemporary research question in immunology, and the model will be used within the scope of this investigation as a tool to make qualitative and quantitative predictions, which ultimately lead to new insight into the problem at hand. These chapters are written in the usual language and style of a *biological* research work: They follow the classical structure of empirical science papers (introduction, results, discussion, methods), and the main text adopts a higher level of immunological terminology, while mathematical formalisms are avoided to the largest possible extent, and are presented in a separate methods section instead.

It is the author's hope that addressing both intended audiences of this thesis in their own language will help both to understand and appreciate our results. However, this thesis cannot provide the entire breadth of background information that would be needed to enable all readers to follow every technical detail. For example, standard experimental techniques like adoptive transfer or two-photon microscopy will not be explained, nor will an in-depth introduction to computational complexity theory be provided. Even so, the biology chapters should enable all readers to convince themselves that the proposed models are indeed firmly settled in relevant questions of modern immunology; similarly, we hope that the computer science chapters will convey at least a basic idea of our key techniques and arguments to a broader readership. We alert those who will indeed read through *all* of these chapters that there is some overlap especially between the introductory parts. This overlap is intentional, and meant to ensure that each chapter is sufficiently self-contained to be read on its own, with the other chapters being consulted for

reference as appropriate.

1.1 Overview of Part I: T Cell Immune Surveillance

T cells are constantly motile cells that protect the body by migrating and circulating through different tissues in search of foreign intruders. The major routes of T cell circulation have been established since the late 1960s through a series of classical experiments, which demonstrated that the cells complete one round of circulation roughly once per day [158]. The migration of T cells was thought to be mostly controlled by chemokines, chemical substances that guide cell migration to and within different tissues [102]. More recently, it became possible to observe T cell migration directly *within* tissue by means of two-photon microscopy [146]. Surprisingly, this technique revealed that T cell migration in tissue is mostly random. In light of this new information, we designed a model of T cell circulation that comprises two scales: macroscopic circulation *between* organs, and microscopic migration *within* organs (Chapter 2). The model provides new insight into how these two scales jointly facilitate timely detection of infections, including those that are localized to small regions of the body (Chapter 3).

The modeling idea arose from a collaboration with Jürgen Westermann. A first joint paper on the subject was published at the *6th International Conference on Artificial Immune Systems (ICARIS 2007)* [148], where it received the Best Student Paper Award. From this initial paper, two different lines of work emerged. First, we initiated a collaboration with the group of Ulrich von Andrian at Harvard Medical School in order to develop more sensible methods for analyzing two-photon microscopy data, which would put the conclusion that T cell migration within lymph nodes is random on a firmer basis. This work is not part of the present thesis, but has been published in the *Proceedings of the National Academy of Sciences of the United States of America* [147]. The second line of work sought to integrate two-photon data with classical data from cell migration experiments, and this goal lead to the development of the model that is presented in this thesis. Several stages of this work were presented as it progressed, including a poster at the meeting on *Lymphocyte Kinetics in Health and Disease* organized by the Infectious Disease Research Network (IDRN) in London, 2008; an invited talk at the annual meeting of the Society for Mathematical Biology in Vancouver, 2009; and an invited talk at the Institute for Theoretical Biology and Bioinformatics, University of Utrecht. The talk at Utrecht lead to a 6-week research visit to the institute in autumn 2010, funded by the Deutscher Akademischer Austauschdienst (DAAD), during which the biological predictions and conclusions presented in Chapter 3 were developed under supervision of Rob J. de Boer and Joost B. Beltman. The results were presented in a selected talk at the IDRN meeting on *T Lymphocyte Dynamics in Acute and Chronic Viral Infection* in London, 2011, and a journal publication is being prepared. The theoretical results presented in Chapter 2, which were developed under supervision of Rüdiger Reischuk, were presented at ICARIS 2011 [122].

1.2 Overview of Part II: Thymic Negative Selection

Thymic negative selection generates a population of T cells capable of recognizing and acting against foreign substances, while tolerating their host organism's domestic substances. Essentially, the body first generates a large random pool of T cells, and then kills those that turn out to be auto-aggressive. Stephanie Forrest and co-workers [55] proposed an algorithmic model of this process, which can be understood as a classifier that learns from only negative examples. In Chapter 4, we will formalize this algorithm, investigate the basic mechanisms by which it learns, and develop methods to compute the classification outcome efficiently. The algorithmic improvements facilitate new ways of generating quantitative predictions, which will be made and tested in Chapter 5. We find that the negative selection algorithm is among the best existing mechanistic (i.e., hypothesis-based) predictors for determining which parts of HIV can be recognized by the immune system's CD8 T cells.

The work presented in Chapter 4 builds upon joint work with Michael Elberfeld, with whom the author published a paper at ICARIS 2009 on the first polynomial time string-based negative selection algorithm [42]. Because this solved a problem that had been open for many years [139], and disproved a conjecture that polynomial time negative selection algorithms do not exist unless P=NP [149], our work received the Best Paper Award at the conference, and we were invited to submit an extended version to the journal *Theoretical Computer Science*. This extended version [43] contains completely redeveloped algorithms which run in linear rather than cubic time. These algorithms were designed and implemented by the author, and their correctness proofs (Lemma 4.62, Fact 4.63, Corollary 4.64, Lemma 4.65, and Theorem 4.66) were joint work with Michael Elberfeld. All other results in Section 4.7 were obtained by the author alone. Independently, the author generalized the results of the ICARIS paper to arbitrary string-based pattern classes and recast the algorithm in the formal framework of algorithmic learning theory. These results were developed under the supervision of Maciej Liśkiewicz, and published at the *12th Genetic and Evolutionary Computation Conference* (GECCO 2010), where the paper was one of three nominees for the Best Paper Award of the Genetics Based Machine Learning track [99]. For this thesis, the results of that paper were mostly rewritten and merged together with some new results into Sections 4.5 and 4.6. Finally, the ideas for the biological validation of the model presented in Chapter 5 were developed and refined in discussions with Can Keşmir and Jorg Calis during the author's visit to Utrecht. These preliminary results have not been published prior to this thesis.

Part I

T Cell Immune Surveillance

Chapter 2

Optimal Restart in a Stochastic Model of Immune Surveillance

Summary. *T cells perpetually circulate through the body in search of infections, using the blood and lymph vasculatory systems as "highways" to travel between different organs and tissues. The need to detect infections as quickly as possible gives rise to an optimization problem: When should a T cell stop searching for antigen in the current region, and invest the time to travel to another one? Here we investigate this question using a simple stochastic model of T cell immune surveillance. Using techniques akin to the analysis of randomized algorithms, we derive the optimal behavior of a single searcher, which presents a surprisingly rich phenomenology. We reveal a gradual transition between two characteristic regions of the search parameter space, and show that the optimum is relatively stable to parameter perturbations. Finally, we discuss potential applications of our results for the analysis of stochastic search heuristics, for instance Las Vegas algorithms with expensive restarts or agent-based intrusion detection systems.*

2.1 Introduction

The T cells of the vertebrate immune system play a key role in detecting and clearing infections. Each T cell is equipped with a highly specific receptor that detects any given pathogen with a very low probability ($\approx 10^{-5} - 10^{-6}$ [15, 108, 113]); conversely, the number of T cells in the body that can act against a given pathogen is very low (estimates in mice range from 80-1200 [113]), and thus it is not possible to provide a complete T cell repertoire for maximum protection in every region of the body. To compensate for this, T cells circulate between different *lymph nodes*. These small organs are distributed strategically across the body (Figure 2.1), and provide hubs where the T cells can efficiently screen the interstitial fluids and proteins of the surrounding tissue for signs of infection. T cells reach lymph nodes from the blood, remain within a lymph node for several hours, and then egress to the lymphatic vessels which ultimately drain back into the blood. One round of

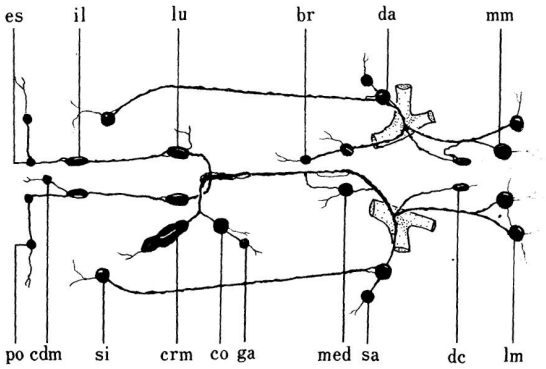


Figure 2.1: Diagram of the lymph system in mice (Figure taken from Kawashima et al. [84]). Left: tail side (caudal), right: head side (cranial). Lymph nodes are placed strategically across the body to provide distributed protection against localized infections; note especially the large cluster of *cranial mesenteric* (crm) lymph nodes near the gut, which is prone to many infections.

recirculation takes approximately 24 hours [158]. Because lymph node entry and egress are via two different vasculatory systems, the sequence of lymph nodes a T cell visits is essentially random.

The lymph node *residence time*, i.e., the time a T cell should stay within each lymph node, is subject to a trade-off: When the cell exits prematurely from the current node, it risks missing an infection that might be present there. On the other hand, if it remains too long in the current node, it might miss an infection that is present somewhere else. In this chapter, we ask the following question: what is the most appropriate time for the cell to return to the circulation? To investigate this question mathematically, we will first need a simple but still reasonably realistic model consisting of two components: (1) A model of T cell recirculation *between* lymph nodes; and (2) a model of antigen scanning *within* lymph nodes.

2.1.1 Organization of this Chapter

This chapter is structured as follows. In the upcoming Section 2.2, we motivate and define the two components of our model. Thereafter, in Section 2.3, we investigate the fundamental properties of the *hitting time*, i.e., the time a circulating cell needs to locate an infection, and provide both exact and asymptotic expressions for the *optimal residence time* that minimizes the expected hitting time. In Section 2.4 we briefly discuss how the hitting time is reduced by several independent searchers. Section 2.5 wraps the analysis up by discussing some implications for our biological applications of the model in the next chapter. As an epilogue, we sketch in Section 2.6 how our work relates to the analysis of randomized search heuristics. This is meant to provide a starting point for future work that might explore such

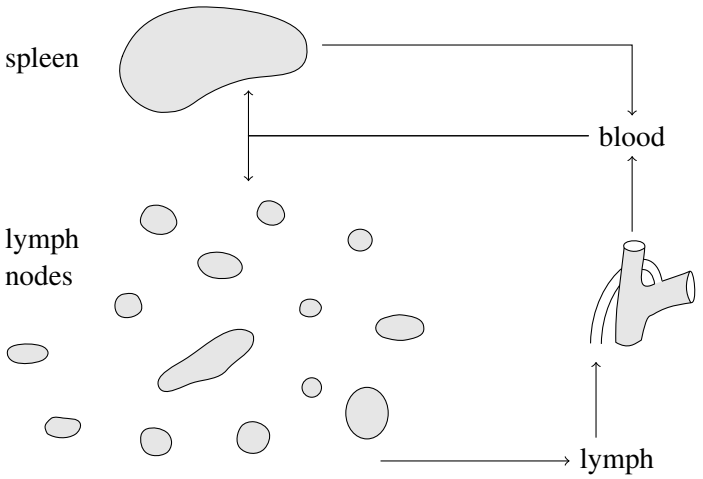


Figure 2.2: Diagram of the major recirculation pathways for immune surveillance. Lymphocytes circulate via the two vasculatory systems of the body: the blood and the lymphatic system. The spleen, which is the biggest lymphatic organ and harbors about 14% of all lymphocytes in humans [159], functions as a central guardian against infections that spread via the blood, and lymphocytes enter and leave the spleen mostly via the blood. In lymph nodes, on the other hand, lymphocytes enter from the blood and exit to the lymphatic system. The lymph drains back into the blood, mostly via the largest lymphatic vessel, the *thoracic duct*, which ends in a junction with the left subclavian vein.

connections in more detail.

2.2 Definition of Our Model

Consider an indexed set of compartments, which we call *bags* (these represent the lymph nodes). Each bag may either be a *good* bag (an infected one) or a *bad* bag (an uninfected one). Let n_{good} denote the number of good bags, and n be the number of all bags. Then $\rho := n_{\text{good}}/n$ gives the fraction of good bags.

We define a discrete-time search process with parameters[1] $\rho, \alpha \in (0,1)$ and $R, T \in \mathbb{N}$ as follows. The searcher starts in a bag chosen uniformly at random, which is a good bag with probability ρ. Each time step, the searcher throws a biased coin. If the present bag is a good one, the success probability of the coin flip is α; otherwise, it is 0. After R time steps, the searcher moves to a new bag chosen uniformly at random, and the transit to the new bag takes T time steps. This process is iterated infinitely often. Let the random variable

$$H \in \{0, \ldots, R-1\} \cup \{T+R, \ldots, T+2R-1\} \cup \ldots$$

denote the the time of the first successful coin flip. We will show that despite its simplicity, the model gives rise to a surprisingly rich phenomenology. Specifically, we derive the following results:

- Expectation $\mathrm{E}[H]$ and variance $\mathrm{Var}[H]$ of the hitting time H (Proposition 2.1);
- Asymptotic expressions for $E[H]$ for large and small R (Propositions 2.2 and 2.3);
- The optimal residence time R_{opt} for given T, ρ, and α (Proposition 2.4); and
- Asymptotic expressions for R_{opt} for two characteristic regions of the search parameter space (Propositions 2.6 and 2.7).

Note that we represent the search for antigen in lymph nodes simply as drawing with replacement. This choice is motivated by the finding that T cells apparently screen lymph nodes by migrating along random paths [106, 145, 156]. Because the lymph node environment is three-dimensional, the random walk search is well approximated by drawing with replacement. For the interested reader, we provide some detail below on how this approximation can be justified via well-known results from random walk theory. Our analysis will start thereafter in Section 2.3.

[1] The parameter ρ is mnemonic for *recruitment* (to good bags a.k.a. infected lymph nodes), and α stands for *activation* (within infected lymph nodes).

2.2.1 Modeling Antigen Search in Lymph Nodes

Two-photon experiments indicate that the search of T cells for antigen is essentially a 3D random walk through the lymph node tissue [106, 145, 156]. It is not yet entirely clear how this random walk comes about, but presumably the network of resident *fibroblastic reticular cells* within the lymph node plays a role in guiding the cell movement [5, 6, 109]. As they move along, T cells encounter so-called *dendritic cells* (DCs), which present them protein samples from the surrounding tissues[2]. If a DC presents a sample that the T cell recognizes as foreign, the two cells establish a long-lasting contact that ultimately leads to activation of the T cell, and thus potentially initiates an immune response. It was estimated, again using two-photon microscopy, that a T cell scans several DCs per hour and needs on average 8 hours to establish a successful contact to a DC [105]. In our model, we approximate this scanning process by drawing with replacement with a constant success probability α, with the two-photon estimate corresponding to $\alpha = 1/8h$.

To justify this approximation, consider a random walk in the lattice $\mathbb{Z}^3$, and let p_n denote the probability that the site reached by the random walk in its n-th step has been visited before. Let S_n denote the number of different sites covered within the first n steps, and r_n the probability that the random walk returns to its starting point at least once within the first n steps. By reversing the random walk in time, it is clear that $p_n = r_n$, and it is known that

$$r_\infty := \lim_{n\to\infty} r_n = \sup_{n\in\mathbb{N}} r_n = 1 - \frac{1}{u(3)} = 0.3405373\ldots\,, \tag{2.1}$$

which is Polya's random walk constant. In this expression, $u(3)$ stands for the integral [60]

$$u(3) = \frac{3}{(2\pi)^3}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\frac{dx\,dy\,dz}{3-\cos x-\cos y-\cos z}\,. \tag{2.2}$$

Thus, we can approximate S_n as follows:

$$\mathrm{E}[S_n] \approx (1-r_\infty)\,n\,. \tag{2.3}$$

Every time a node is visited, we therefore have a chance of at least $1-r_\infty = 66\%$ that it is a node we have not visited before. Now assume that targets are distributed on the lattice such that every vertex is a target with probability c. We are interested in the *first hitting time*, i.e. the number of steps it takes a random walk to hit a target for the first time, denoted by $H = H(c)$. Its expectation can be evaluated as follows:

$$\begin{aligned}
\mathrm{E}[H] &= \sum_{j=1}^{\infty} j\cdot\Pr[H=j] = \sum_{j=1}^{\infty} j\cdot(\Pr[H>j-1]-\Pr[H>j]) && (2.4)\\
&= \sum_{j=0}^{\infty}\Pr[H>j] = \sum_{j=0}^{\infty}\mathrm{E}[(1-c)^{S_j}] && (2.5)\\
& && (2.6)
\end{aligned}$$

[2] Antigen presentation will be explained in more detail in Chapter 4.

For $c \ll 1$, i.e. when $(1-c)^n$ decreases very slowly, the last expression can be approximated by

$$\mathrm{E}[H] \approx \sum_{j=0}^{\infty} (1-c)^{\mathrm{E}[S_j]} \approx \sum_{j=0}^{\infty} (1-c)^{(1-r_\infty)\,j} \tag{2.7}$$

$$= \frac{1}{1-(1-c)^{(1-r_\infty)}} = \frac{1}{1-(1-c)^{0.659462670\ldots}} \,. \tag{2.8}$$

This approximation, called the *Rosenstock approximation* [157], is known to give good estimates for $c < 0.05$. In our case, c would denote the probability for each DC to activate the crawling T cell; given that perhaps 150 DCs are contacted per hour [11], and that the cell searches some hours before it is activated, $c < 0.05$ is a reasonable assumption in our case.

Let $t_\infty = 1 - r_\infty$. Using a binomial series expansion, we obtain the following for the step-wise success probability of the random walk:

$$\frac{1}{\mathrm{E}[H]} \approx 1-(1-c)^{t_\infty} = 1 - \sum_{k=0}^{t_\infty} \binom{T_\infty}{k} (-c)^k = c\,t_\infty - O(c^2) \,. \tag{2.9}$$

Comparing this random walk search to a systematic search of the lattice vertex by vertex, which is equivalent to drawing with replacement, we see that both methods essentially give rise to a geometric hitting time distribution. For small c, the systematic search outperforms the random walk in terms of expected hitting time by a factor of at most 1.5. This efficiency is remarkable because the random walk does not need any memory nor sense of orientation and is thus simple to perform. On the other hand, it is hard to imagine how a cell could screen a tissue region systematically. However, at least three spatial dimensions are needed to make the random walk search competitive with the systematic search – the random walk search strategy would be far less effective in a one- or two-dimensional environment: A random walk in less than three dimensions is *recurrent* (it returns to the origin with probability 1), and thus the fraction of newly discovered sites converges to 0 with time.

Even though our argument above is based on random walk on a lattice, the same reasoning would hold for random walks on more complex topologies – averaged over a large amount of steps, any random walk is eventually subject to the central limit theorem and can then be approximated by a simpler random walk. Indeed, a quite detailed, spatially explicit model of the T cell random walk by Beltman et al. [11] has also predicted that 2/3 of the contacts between T cells and DCs should be unique.

2.3 Minimizing the Hitting Time

In this section, we investigate some fundamental properties of our model. We start out by stating the elementary moments of the hitting time distribution.

Proposition 2.1. *Let H denote the first hitting time of a search process according to Section 2.2 with parameters R, T, α, and ρ. Then H has expectation*

$$E[H] = \frac{1-\alpha}{\alpha} - \frac{1-\zeta}{\zeta} R + \left(\frac{1-\xi}{\xi}\right) (R+T) + 1 \tag{2.10}$$

and variance

$$Var[H] = \frac{1-\alpha}{\alpha^2} - \frac{1-\zeta}{\zeta^2} R^2 + \left(\frac{1-\xi}{\xi^2}\right) (R+T)^2 \tag{2.11}$$

where $\zeta = 1-(1-\alpha)^R$ and $\xi = \rho\zeta$.

Proof. Let us call a sequence of searching a bag (which takes time R) and transiting to the next bag (which takes time T) a *phase*. Let U be a random variable denoting the number of unsuccessful phases before the searcher finds a target in a good bag in phase $U+1$, and let S be the number of samples drawn in phase $U+1$ before the target is found. Then the hitting time is given by

$$H = (T+R)\,U + S + 1\ . \tag{2.12}$$

Since U and S are stochastically independent, due to the linearity of expectation and variance we get

$$\begin{aligned} \mathrm{E}[H] &= (R+T)\,\mathrm{E}[U] + \mathrm{E}[S] + 1 \text{ and} && (2.13)\\ \mathrm{Var}[H] &= (R+T)^2\,\mathrm{Var}[U] + \mathrm{Var}[S]\ . && (2.14) \end{aligned}$$

U is geometrically distributed with parameter ξ, hence

$$\begin{aligned} \mathrm{E}[U] &= \frac{1-\xi}{\xi} \text{ and} && (2.15)\\ \mathrm{Var}[U] &= \frac{1-\xi}{\xi^2}\ . && (2.16) \end{aligned}$$

S on the other hand has a geometric distribution that is "truncated" to the finite support $\{0,\ldots,R-1\}$. With some algebra, it can be verified that

$$\begin{aligned} \mathrm{E}[S] &= \frac{1}{\zeta}\sum_{k=0}^{R-1} k\,(1-\alpha)^k\,\alpha = \frac{1-\alpha}{\alpha} - \frac{1-\zeta}{\zeta} R\ , && (2.17)\\ \mathrm{Var}[S] &= \left(\frac{1}{\zeta}\sum_{k=0}^{R-1} k^2\,(1-\alpha)^k\,\alpha\right) - \mathrm{E}[S]^2 = \frac{1-\alpha}{\alpha^2} - \frac{1-\zeta}{\zeta^2} R^2\ . && (2.18) \end{aligned}$$

Putting S and U together, we obtain the result. □

The above equations for expectation and variance are reminiscent of the fact that our search process is a combination of two sampling processes with replacement – the global search for a good bag, and the local search for a target in a good bag.

2.3.1 Asymptotics of the Expected Hitting Time

To understand the dependencies of the expected hitting time, we first analyze its asymptotics for large and small R. For $R \geq M\,\alpha^{-1}$ with $M \gg 1$, the term $(1-\alpha)^R \leq e^{-M}$ becomes very small, and thus ζ is close to 1. This results in the following asymptotics.

Proposition 2.2. *Let H, R, T, ρ, α be defined as above, and fix a large constant $M \gg 1$ such that $R \geq M/\alpha$. Then*

$$E[H] \;\approx\; \frac{1-\alpha}{\alpha} + \frac{1-\rho}{\rho}(R+T) \;\in\; \Theta(R)\,. \tag{2.19}$$

Hence, spending significantly more time searching a bag than the expected hitting time α^{-1} for a good bag increases the overall hitting time linearly. On the other hand, if R becomes too small, we get the following.

Proposition 2.3. *Let H, R, T, ρ, α be defined as above, and fix some small nonzero $\varepsilon \ll 1$ such that $R \leq \varepsilon\,\alpha^{-1}$. Then*

$$E[H] \;\approx\; \frac{1}{\rho\,\alpha} + \frac{T}{\rho\,\alpha}\,R^{-1} - T \;\in\; \Theta(1/R)\,. \tag{2.20}$$

Proof. Since $R \leq \varepsilon\,\alpha^{-1}$, one can use the approximation $(1-\alpha)^R = 1 - R\,\alpha + O((R\,\alpha)^2)$. This implies $\zeta \approx \alpha R$ and $\xi = \rho\,\alpha R$, which upon insertion into Equation 2.10 gives the result. □

The $\Theta(1/R)$ asymptotics for small R (i.e., halving an already small R almost doubles the number of phases until a global hit occurs) can be intuitively explained by noting that most of the time is spent in transit between bags, since the success probability within a bag ($\approx R\,\rho^{-1}\,\alpha^{-1}$) is very low (Figure 2.3).

2.3.2 The Optimal Residence Time

Let T, ρ and α be given. What is the optimal choice for R, i.e., the one that minimizes $\mathrm{E}[H]$? Let us denote this value by R_{opt}. It is given by the following proposition.

Proposition 2.4. *Let H, R, T, ρ, α be defined as above, Consider T, ρ, α as constants, and $E[H]$ as a function of R. Then $E[H]$ is minimized by*

$$R_{opt} = W_{-1}\left(-\frac{(1-\alpha)^{\frac{T}{1-\rho}}}{e}\right)\frac{1}{\ln(1-\alpha)} - \frac{T}{1-\rho} + \frac{1}{\ln(1-\alpha)} \tag{2.21}$$

where W_{-1} is the non-principal branch of the Lambert W function [31].

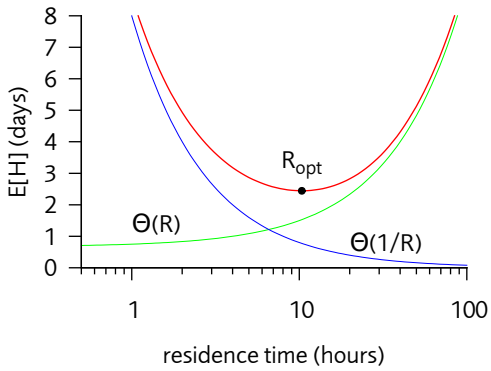


Figure 2.3: The expected hitting time $\mathrm{E}[H]$ as per Equation 2.10 and its asymptotics as per Propositions 2.2 and 2.3 as functions of the residence time R for the parameters $\rho = 1/3, T = 8h, \alpha = 1/8h$. Note the characteristic basin-like shape.

Proof. We have to solve $\frac{d}{dR}\mathrm{E}[H] = 0$, which is equivalent to

$$0 = \frac{d}{dR}\left[\frac{1-\zeta}{\zeta}R + \frac{1-\xi}{\xi}R + \frac{1}{\xi}T\right] = \frac{d}{dR}\left[\left(\frac{1}{\rho}-1\right)\frac{R+T/(1-\rho)}{1-(1-\alpha)^R}\right] . \quad (2.22)$$

Now, differentiating we get

$$0 = 1-(1-\alpha)^R + (R+T/(1-\rho))\,(1-\alpha)^R \ln(1-\alpha) \quad (2.23)$$

$$\Leftrightarrow \left(R + \frac{T}{1-\rho} - \frac{1}{\ln(1-\alpha)}\right)(1-\alpha)^R = -\frac{1}{\ln(1-\alpha)} . \quad (2.24)$$

This is a transcendental equation and thus cannot be solved for R using only standard algebra. A tool for solving equations of this type, which arise in many applications [31], is the *Lambert W function* defined by

$$x\,e^x = y \;\Leftrightarrow\; x = W(y) . \quad (2.25)$$

Using this function, we can solve equations of the type $(x+b)\,a^x = c$ as follows:

$$\begin{aligned} & (x+b)\,a^x = c \\ \Leftrightarrow\; & (x+b)\,a^{x+b} = a^b c \\ \Leftrightarrow\; & (x+b)\ln a\, e^{(x+b)\ln a} = a^b c \ln a \\ \Leftrightarrow\; & \hat{x}e^{\hat{x}} = \hat{y} \\ \Leftrightarrow\; & \hat{x} = W(\hat{y}) \\ \Leftrightarrow\; & x = \frac{W(\hat{y})}{\ln a} - b . \end{aligned} \quad (2.26)$$

Inserting

$$a = (1-\alpha)\;;\; b = \frac{T}{1-\rho} - \frac{1}{\ln(1-\alpha)}\;;\; c = -\frac{1}{\ln(1-\alpha)}$$

in our case gives

$$\hat{y} = a^b c \ln a = -(1-\alpha)^{\frac{T}{1-\rho}}/e \ .$$

Because $-1/e < \hat{y} < 0$, the two branches W_0 and W_{-1} of the Lambert W function both solve the equation. In our case, the non-principal branch W_{-1} is the meaningful one because it maps to the interval $(-\infty,-1)$, while W_0 maps to $[-1,0]$. Inserting $\hat{y}, a, b$, and c into Equation 2.26 yields the claimed expression. □

2.3.3 Asymptotics of the Optimal Residence Time

The exact solution given by Proposition 2.4 for the optimal residence time is rather complex and yields little insight into the dependencies between R_{opt} and the parameters T, ρ, and α. Thus, we now turn our attention to two important regions in the parameter space for which more illustrative asymptotic forms of R_{opt} can be derived. For simplicity, we assume that α is moderately small (e.g., like our estimation $\alpha = 1/8$ for T cell activation in lymph nodes); a similar analysis is possible without this assumption, but the asymptotic formulae become more complicated.

We will show the existence of two quite different parameter regimes. The switching point between them is given by the following trade-off.

Definition 2.5. *Let H, T, ρ, α be defined as above, and let α be moderately small such that $\ln(1-\alpha) \approx -\alpha$. For*

$$\frac{1}{\alpha} \ll \frac{T}{1-\rho} \ , \tag{2.27}$$

we call H transit dominated. *Otherwise, if*

$$\frac{1}{\alpha} \gg \frac{T}{1-\rho} \ , \tag{2.28}$$

then we call H locally dominated.

Note that the parameter ρ plays hardly any role in defining these two parameter regimes as in the interesting cases, ρ is typically rather small (otherwise the search problem would not be very different from a simple local search in one bag). Thus, surprisingly, *the transit time T is more important than the difficulty of the global search problem*, which can be measured by $1/\rho$ rather than $1/(1-\rho)$.

The upcoming two propositions yield interesting differences between transit dominated and locally dominated settings.

Proposition 2.6. *Let H be transit dominated by T, ρ, α. Then*

$$R_{opt} \approx \frac{\ln T - \ln(1-\rho) + \ln \alpha}{\alpha} \ . \tag{2.29}$$

Proof. We use the following power series expansion for $W_{-1}(y)$, which converges quickly for $1/e \ll y < 0$ [31]:

$$W_{-1}(y) = \lambda_1 - \lambda_2 + \sum_{k=0}^{\infty}\sum_{m=1}^{\infty} c_{km}\frac{\lambda_2^m}{\lambda_1^{m+k}} \tag{2.30}$$

$$= \lambda_1 - \lambda_2 + \sum_{k=0}^{t-1}\sum_{m=1}^{t-k} c_{km}\frac{\lambda_2^m}{\lambda_1^{m+k}} + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^{t+1}\right), \tag{2.31}$$

where $\lambda_1 := \ln(-y)$, and $\lambda_2 := \ln(-\lambda_1) = \ln(-\ln(-y))$. The c_{km} are constants that are not important for our analysis, since we asymptotically approximate W_{-1} for $y \to 0$ by truncating the sum terms of the power series ($t = 0$). For our $\hat{y}$ defined in the proof of Proposition 2.4 this results in $\lambda_1 = \frac{T}{1-\rho}\ln(1-\alpha) - 1$ and

$$W_{-1}\left(-\frac{(1-\alpha)^{\frac{T}{1-\rho}}}{e}\right) = \lambda_1 - \lambda_2 + O\left(\frac{\lambda_2}{\lambda_1}\right) \tag{2.32}$$

$$= \frac{T\ln(1-\alpha)}{1-\rho} - 1 - \ln\left(1 - \frac{T\ln(1-\alpha)}{1-\rho}\right) + O\left(\frac{\lambda_2}{\lambda_1}\right). \tag{2.33}$$

Inserting this asymptotic expansion into the closed form for R_{opt} given by Proposition 2.4, some terms cancel out and we arrive at

$$R_{\text{opt}} = -\ln\left(1 - \frac{T\ \ln(1-\alpha)}{1-\rho}\right)\frac{1}{\ln(1-\alpha)} \tag{2.34}$$

$$+ O\left(\frac{\lambda_2}{\lambda_1\ln(1-\alpha)}\right). \tag{2.35}$$

In the region where $\frac{T}{1-\rho} \gg -\ln(1-\alpha) \approx \alpha$, the argument $1 - \frac{T\ \ln(1-\alpha)}{1-\rho}$ of the first logarithm is much larger than 1 and can be replaced by $\frac{T}{1-\rho}\ (-\ln(1-\alpha))$. This gives the approximation

$$R_{\text{opt}} \approx \frac{\ln T - \ln(1-\rho) + \ln(-\ln(1-\alpha))}{-\ln(1-\alpha)}. \tag{2.36}$$

which is valid for any α in a transit dominated setting. Substituting $-\ln(1-\alpha)$ for α yields the claimed expression. □

To understand why this approximation eventually breaks down for $\alpha \to 0$, we look more closely at the O-term of Equation (2.35) for R_{opt}:

$$R_{\text{opt}} = -\ln\left(1 - \frac{T\ \ln(1-\alpha)}{1-\rho}\right)\frac{1}{\ln(1-\alpha)} + O\left(\frac{\ln\left(1 - \frac{T\ \ln(1-\alpha)}{1-\rho}\right)}{\frac{T\ \ln(1-\alpha)^2}{1-\rho} - \ln(1-\alpha)}\right) \tag{2.37}$$

Applying De l'Hôpital's Rule it can be shown that R_{opt} *without* the O-term approaches a constant value for $\alpha \to 0$, whereas the O-term starts to dominate. This limit takes us to the locally dominated regime, which we examine next.

Proposition 2.7. *Let H be locally dominated by T, ρ, α. Then we have*

$$R_{opt} \approx \sqrt{\frac{2T}{(1-\rho)\alpha}}\,. \tag{2.38}$$

Proof. In the transit dominated regime, the argument of W_{-1} in the closed form of R_{opt} (Proposition 2.4) is close to $-1/e$, the branch point of the W function. Near this branch point, the power series used in the proof of the previous result is no longer useful (Figure 2.4A). From the results of Corless et al. [31] one can derive an alternative power series expansion for W_{-1} near the branch point:

$$W_{-1}(y) = \sum_{t=0}^{\infty} (c_t)^t = -1 + \sigma - \frac{1}{3}\sigma^2 + \frac{11}{72}\sigma^3 + \ldots \tag{2.39}$$

In this expression, $\sigma = -\sqrt{2ey+2}$, and thus $|\sigma| \leq 1$. Again the c_t are constants that are irrelevant for our purpose, since we truncate the series after $t = 1$ to obtain an asymptotic approximation. Inserting again our $\hat{y}$ from the proof of Proposition 2.4 yields

$$W_{-1}\left(-\frac{(1-\alpha)^{\frac{T}{1-\rho}}}{e}\right) = -1 - \sqrt{2 - 2(1-\alpha)^{\frac{T}{1-\rho}}} + O\left(1 - (1-\alpha)^{\frac{T}{1-\rho}}\right)\,, \tag{2.40}$$

from which we get the following expression for R_{opt}:

$$R_{\text{opt}} = \frac{-\sqrt{2}}{\ln(1-\alpha)} \sqrt{1 - (1-\alpha)^{\frac{T}{1-\rho}} + O\left(\left(1 - (1-\alpha)^{\frac{T}{1-\rho}}\right)^2\right)}\,. \tag{2.41}$$

By the definition of locally dominated parameters, we have $T\,\alpha/(1-\rho) \ll 1$. Thus we can substitute $(1-\alpha)^{T/(1-\rho)}$ by $1 - T\,\alpha/(1-\rho)$. This yields the claimed expression. □

2.4 Searching in Parallel

So far we have only investigated the behavior of single searchers. However, the optimality results of course carry over to populations of independent (non-communicating) searchers, because optimizing each searcher in such a population will optimize the entire population. In this section, we will briefly investigate how much the search is accelerated by m independent parallel searchers.

Let H_m denote the hitting time of such a parallel search. For values of m that are significantly smaller than the expected hitting time $\mathrm{E}[H]$ of a single searcher, the expectation of the m-parallel search can be approximated by

$$\mathrm{E}[H_m] \approx \frac{\mathrm{E}[H]}{m}\,, \tag{2.42}$$

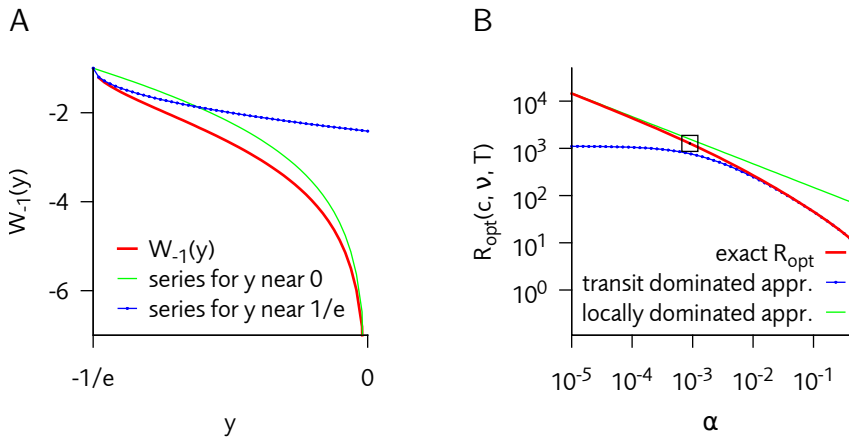


Figure 2.4: **(A)** Illustration of the two different approximations (Equations 2.30 and 2.39) used for the Lambert W function in the proofs of Propositions 2.6 and 2.7. **(B)** Transition of the optimal residence time between the two regions described by Proposition 2.7 (densely dashed) and Proposition 2.6 (dashed) for $T = 1000$, $\rho = 0.1$, and varying α. The square marks the point where $\frac{T}{1-\rho} = \frac{1}{\alpha}$.

since in this case the hitting probability of a single step grows approximately by a factor m. This approximation becomes invalid for large m, because the bags become saturated with searchers and thus additional searchers no longer yield substantial speedup. However, in the saturation limit it is possible to use the following approximation instead:

$$\mathrm{E}[H_m] = \frac{1}{1-(1-R\rho\,\alpha/(R+T))^m}\ . \tag{2.43}$$

This approximation is obtained by noting that for a randomly chosen time step, every searcher has an overall chance of $R\rho/(R+T)$ to be in a good bag, and within a good bag the chance of finding a target is α. Assuming that the fraction of searchers in good bags at every timestep is indeed equal to $\gamma = R\rho/(R+T)$ (rather than a random variable with expectation γ), we can approximate the parallel search by random sampling with replacement with a success probability of $1-(1-\gamma\alpha)^m$. Note that for $m = 1$, the above equation is (up to the constant T) equal to the equation in Proposition 2.3 that describes the asymptotics of $\mathrm{E}[H]$ for small R.

Some simulation results on the speedup efficiency are displayed in Figure 2.5. Notably, while for optimally tuned searchers the speedup is indeed well described by the above equations that predict a linear speedup with saturation, for *non-optimal* residence times it is possible to obtain a *superlinear* speedup by increasing the number of searchers (e.g. in Figure 2.5, around $m = 100$ for $R = 1$ and around $m = 20$ for $R = 100$).

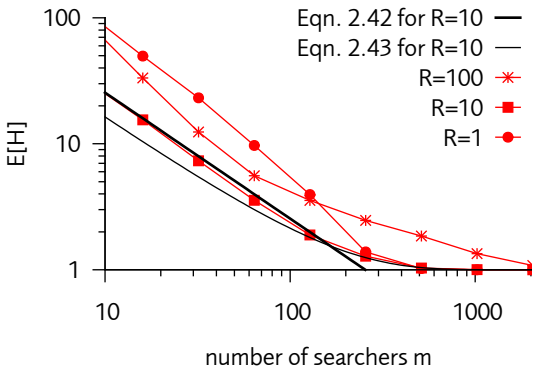


Figure 2.5: Speedup of the expected hitting time by m independent parallel searchers. $\mathrm{E}[H]$ is plotted as a function of m for $T = 10, \rho = 0.1$, and $\alpha = 0.12591$, which gives $R_{\mathrm{opt}} = 10$. For the optimally tuned population, the approximate expected hitting times predicted by Equations 2.42 and 2.43 are displayed, where Equation 2.42 describes a power law with slope -1. Per data point we performed 1000 simulations, so that all standard errors of the mean were less than 2%.

2.5 Discussion

The asymptotic results from the previous Section 2.3 yield some important arguments that our model is indeed suitable for use within our intended biological context. First, Propositions 2.4, 2.7, and 2.6 jointly show that the optimal residence time is hardly dependent on ρ as long as ρ is not close to 1. This is critical because in reality, we cannot assume that ρ is constant – infection is accompanied by inflammation, which accelerates the recruitment of T cells to the infected area.

Second, in the next chapter we will investigate the hypothesis that evolution has indeed optimized the lymph node residence time of real T cells. For this to be a reasonable question to ask, it is crucial for the optimum not to change too drastically when the search space parameters are slightly perturbed; during the lifetime of an animal, it will encounter different infections that are not all equally severe, leading to variations in α, and the parameters T and ρ will change as the animal grows, shrinks, perhaps loses a lymph node or two due to severe illnesses, et cetera. Thus, assume that we set our R to within a factor κ of R_{opt}, i.e., $R_{\mathrm{opt}}/\kappa < R < \kappa \cdot R_{\mathrm{opt}}$. Then it follows from the results in Sections 2.3 and 2.3.1 that $\mathrm{E}[H]$ is also within a factor κ of its optimal value. Combining this with the results from the previous section, we may illustrate the effect of the perturbance by considering the two characteristic parameter regimes: In the *locally dominated* regime (Proposition 2.7), we have *square root* asymptotics for $1/\alpha, 1-\rho$, and R, implying that $\mathrm{E}[H]$ would be within factor $\sqrt{\kappa}$ of its optimal value if one of these parameters is perturbed by factor κ. Hence, the perturbation has sublinear impact. In the *transit dominated* regime (Proposition 2.6), the effect of perturbing T and $1-\rho$ would even be merely *logarithmic*; however, the effect of perturbing c in this regime would be *linear*. In

either case, perturbing ρ instead of $1-\rho$ by a small factor κ has virtually no effect if ρ is already small.

In summary, the optimum appears to be robust against mild parameter perturbations. Taken together with our finding that even large proportional variations of ρ do not matter as long as ρ remains well below 1, our analysis provided no argument that disqualifies our model from being applied to T cell immune surveillance. One assumption that we will need to take care of is that T is a constant; in reality, the time spent in circulation between different lymph node is of course a random variable with considerable variance. Even though intuition seems to tell us that this should not invalidate our results as long as the probability distribution is not somehow degenerate or very skewed, future work is needed to put this intuition on a firmer basis.

2.6 Epilogue: Connection to Stochastic Local Search

Optimization questions of the type studied in this chapter arise in many fields, including the following:

- *Security:* Some distributed intrusion detection systems (e.g. [68, 66]) operate in a similar fashion like T cells: They consist of large numbers of agents, each specialized to detecting a certain intrusion type, which continuously migrate between different hosts in the network. This leads to exactly the question considered here: Which strategy should the agents use to decide when to migrate from the current host to another one?

- *Economy:* Some control procedures, e.g. quality control in companies or control of goods transported across borders, rely on taking random samples because continuous supervision of the entire system is impossible. This again leads to a trade-off between the need to investigate a single unit (company branch, truck at the border) as thoroughly as possible and the need of controlling a reasonably large fraction of all units.

- *Population Ecology:* A foraging animal that searches for food in a patchy environment (think picking apples from trees, or hunting in forests) needs to ensure that it takes up sufficient energy over time. Because the animal depletes its prey in the current patch (tree, forest), it will need to look for a fresh food patch from time to time, and start doing so early enough that sufficient energy is still left to find the next patch. When is the optimal time to give up the current patch, and migrate to a new one?

- *Algorithms:* A *Las Vegas algorithm* is an algorithm with deterministic output whose runtime is a random variable. Such algorithms can be accelerated by restarting when an unfavorable region of the runtime distribution is reached; a well-known example is Schöning's probabilistic SAT solver [75]. When

restarts are themselves expensive, one needs to balance between the time needed to generate a new candidate solution and the expected time gain.

We wish to discuss the last two examples in some more detail. The subfield of population ecology that is concerned with optimal behavior of foraging animals is called *optimal foraging theory* (OFT) [135]. A key difference between OFT and the model considered in this chapter, however, is the definition of the environment. In OFT, different patches are defined by means of different energy uptake functions, and the goal is to maximizing the cumulative energy uptake under the assumption that the animal knows the uptake function for each patch it visits. In our case, the "patches" are defined by two different probability distributions, between which the migrating cell can *not* distinguish, and the goal is to minimize the expectation of H. This means that the core theorem of OFT, the *marginal value theorem* [29], cannot be applied to our problem.

In the field of randomized search heuristics, a theorem related to the work presented in this chapter is that of Luby et al. [100] on optimal restart of Las Vegas algorithms. Luby et al. show that an optimal strategy for speeding up probabilistic algorithms with finite expected runtime and a known runtime distribution is to restart the algorithm in fixed time intervals. This nice theorem is however not applicable to our case: Within bad bags the expected "runtime" (time until a successful coin flip) is infinite, so our problem cannot be stated as a Las Vegas algorithm. Moreover, Luby et al. assume that there is no "cost" associated with restarting the algorithm, even though in practice this may be substantially larger than the cost of a single search step [75]. For example, this is the case for suitably encoded versions of the traveling salesman problem (TSP): generating a starting point (a tour with a reasonably low weight) takes substantially longer than a local search step in the problem space (e.g. by locally modifying some edges). It remains to be seen whether taking the restart time into account explicitly could lead to substantially faster stochastic algorithms for such problems. In the terminology of the present chapter, the "bags" for such a problem would be different starting locations in the problem search space[3], and we would need to generalize the local search process from drawing with replacement to an arbitrary runtime distribution $P_i(t)$, which depends on the starting location i in the problem space.

Thus, an interesting question for future research would be to address how existing results on optimal stochastic search processes in computer science and nature could be integrated into a common framework, and whether some results could be transferred across these fields to provide new tools and insights.

[3]Note that our definition does not require that the bags be disjoint regions in the problem space.

Chapter 3

A Two-Scale Model of T Cell Circulation, Recruitment, and Activation

Summary. *Lymphocytes survey the body for antigen by perpetually migrating within and between different secondary lymphoid organs (SLOs). Upon infection, the rare antigen specific T cells need to be recruited from the recirculating pool into the immune response. The major routes of lymphocyte circulation* between *SLOs have been mapped out since the 1960s, and are now well established. In the past decade, technological advances like two-photon microscopy have provided important new insight on lymphocyte migration* within *SLOs. Because both scales of lymphocyte migration play important roles for timely detection and control of infections, we are now facing the challenge of combining the established macroscopic information and the new microscopic data into a coherent two-scale model. Here, we propose a simple model comprising T cell circulation, recruitment, and activation. Our model provides quantitative and functional links between experimental data obtained by diverse techniques like lymph cannulation, histology, two-photon microscopy, and cellular barcoding. We show that these data are in good mutual agreement, and they indicate that random recirculation of T cells combined with draining lymph node (LN) enlargement is sufficient to explain the remarkable efficiency of T cell recruitment. The model predicts that the LN residence time is subject to a trade-off between local and global reliability of antigen detection, and mathematical analysis of this trade-off reveals that physiologic LN residence times coincide with the optimal range for timely detection of localized infections.*

3.1 Introduction

Initiation of an immune response requires that rare antigen-specific lymphocytes come together in SLOs. When an organism is challenged by a previously unseen

pathogen, the number of immune cells in the host that are capable of detecting that pathogen is extremely low – e.g., for CD8 T cells in mice, roughly 100-200 in a pool of 2-4 $\cdot$ 10^7 [15]. Recruiting these rare specific cells, which are scattered all across LNs, the spleen, and other places [159, 32], is often compared to finding a needle in a haystack [129]. And yet, it was recently shown [153, 144] that T cell recruitment is remarkably efficient: for systemic infections, most specific cells get recruited in the first 72 hours post infection. Although it is not yet fully understood how this comes about, it is clear that two different time scales play a major role.

The macroscopic scale of lymphocyte circulation and migration *between* organs and tissues was intensively studied from the late 1960s to the early 1990s. Using techniques like lymph drainage, cell counting in organ suspensions or histologic analysis of microtome slices, the routes of lymphocyte circulation were mapped out in detail [158, 162]. For many questions, e.g. the distribution of lymphocytes in the body [58], the studies of that era still represent the most detailed and thorough sources. However, one limiting factor was that it was seldom possible to observe lymphocyte migration *within* SLOs (with a few notable exceptions [155]). This changed in the early 2000s with the advent of two-photon microscopy, which allowed imaging migrating lymphocytes directly in intact, living organs [23]. Two-photon imaging has since generated valuable new information about the microscopic kinetics of lymphocyte migration, antigen presentation, and lymphocyte activation in tissue. These new data support a stochastic model of antigen detection by migration along random paths [156], which seems to contrast the prevailing view that chemokines play an important role in guiding lymphocytes within LNs [102].

Lymphocyte migration research on both of these scales has been accompanied by efforts to model and understand the data mathematically. The main goals of macroscopic lymphocyte migration models [134, 132] were to explain the characteristic lymphocyte distribution and recirculation patterns, and especially to reconcile partially contradictory findings from cannulation experiments [133]. The interest in mathematical and computational models of microscopic lymphocyte migration [52] was even larger, and several experimental groups have directly collaborated with modelers [11, 48, 123]. One of the many questions studied was how the motility mode of lymphocyte migration could be inferred from two-photon data [10]. While some models have attained a considerable degree of complexity [41, 123, 98], we are not aware of any model so far that spans both scales of lymphocyte migration.

In this work, we propose a simple model that comprises the following three stages of T cell migration: *Circulation* between blood, LNs, and the spleen; *recruitment* into a draining SLO post infection; and *activation* within the draining organ. Our model brings together both old and new lymphocyte migration data on different spatial and temporal scales. The main questions we investigate are: (i) How does the immune system accomplish T cell recruitment with such a remarkable efficacy? (ii) Which quantitative and functional connections can we establish between the macroscopic and microscopic kinetics of T cell migration?

3.2 Results

3.2.1 Simulating T Cell Circulation

Our model distinguishes between three compartments: the blood, the LNs, and the spleen. Moreover, we will later distinguish between draining and non-draining LNs. The T cell circulation pathway is illustrated in Figure 3.1: Cells start in the blood, from where they are recruited into the spleen with rate ρ_S and to LNs with rate ρ_{LN}. After traveling through the respective organ, cells return to the blood, and a new round of circulation begins. The compartments are modeled as one-dimensional rods. A cell entering a compartment from the blood is released in the middle of this rod, and starts to migrate randomly until it reaches either end. Mathematically, this migration process can be formalized using a 1D diffusion equation, and by solving this differential equation for compartment transit analytically (Section 3.4.1), one can calculate diffusion coefficients that reproduce desired average residence times. Thus, we set the lengths of the rods to 1, and introduced two parameters R_S and R_{LN} giving the average residence times in the spleen and in LNs, respectively. The random walk diffusion coefficient within the compartments was then adjusted to produce the desired average transit time R_S or R_{LN} (Section 3.4.1). The differential equations that define the model can be converted to a simple rule-based stochastic description of the cell trafficking process (Section 3.4.2). Simulating the model stochastically instead of solving the differential equations allows investigating the fates of individual cells, e.g., determining the range of times that cells need to travel from one LN to another one.

This first component of our model is an adaptation of the model that Stekel et al. [134] used to investigate lymphocyte circulation. The most important differences are that their model was deterministic rather than stochastic, and that they modeled the compartment transit as directed rather than random migration. Presumably, they did so because at the time it was assumed that lymphocytes migrate through SLOs in a directed fashion even in the absence of antigen. However, no evidence has been found for such directed migration. Instead, it appears that lymphocytes migrate by default along random paths [106, 18, 156]. Despite this differences, it turned out that the parameters used by Stekel et al., namely $R_s = 6$h, $R_\ell = 13.5$h, $\rho_S = 1.0$/h, and $\rho_{LN} = 1.5$/h [134], give fairly accurate predictions of the cell distribution at steady state, blood transit time, and blood-to-lymph transit time (Table 3.1). Therefore, we use these same values as defaults in the rest of this chapter.

3.2.2 Kinetics of T Cell Recruitment After Infection

Using the "cellular barcoding" technique [125], van Heijst et al. [153] determined the percentage of the specific T cells that get recruited into the immune response for various systemic and local infections. For an intravenously administered *listeria monocytogenes* infection, they found that recruitment was >95% at day 7 post

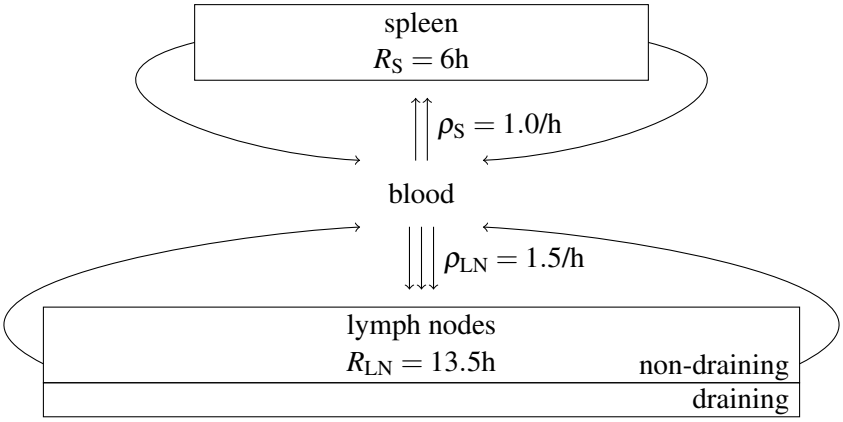


Figure 3.1: **Basic model of T cell circulation.** Our simple model of T cell transit and recirculation considers three compartments: blood, lymph nodes (LNs), and the spleen; for simulating local infections, we also distinguish between draining and non-draining LNs. The diagram illustrates the flux of T cells between the compartments: T cells start in the blood, from where they are recruited into LNs with rate ρ_{LN} and into the spleen with rate ρ_s; these rate parameters also determine the blood residence time, which is equal to $1/(\rho_{LN} + \rho_S)$. With our default parameters, they remain in the respective organ on average for $R_{LN} = 13.5h$ (LNs), or $R_S = 6h$ (spleen), before recirculating to the blood. Compartment transit is modeled as a random walk starting in the middle of the compartment, and the cell is moved back into the blood when it reaches either end. The random walk diffusion coefficient is set according to the desired mean residence time (Section 3.4.1).

	parameter	*value*	*references*
R_{LN}	LN residence time	13.5h	[51]
R_{S}	spleen residence time	6h	[50, 49, 51, 34, 115]
ρ_{LN}	recruitment to LN	1.5/h	
ρ_{S}	recruitment to spleen	1.0/h	
ρ_{dLN}	recruitment to dLN	0.05–0.8/h	[130, 92]
α	activation in dLN	1/8h	[105]

	prediction	*value*	*references*
	blood residence time	25 min	[161, 14]
	T cell ratio blood:spleen:LNs	3:23:75	[150, 120, 51, 159, 58]
	blood-to-lymph transit time	21 h	[161, 158]

Table 3.1: **Model parameters and basic quantitative predictions.** The T cell circulation component of our model (Figure 3.1) has the four parameters $R_{\mathrm{LN}}, R_{\mathrm{S}}, \rho_{\mathrm{LN}}$, and ρ_{S}, for which we used by default the estimates obtained by Stekel et al. [134] for a more complex circulation model with seven parameters. References that support these values are given where available. Although no direct estimates are available for recruitment to LN and spleen, the default values can accurately predict the T cell distribution at steady state, the blood residence time, and the blood-to-lymph transit time. For modeling localized infections with a non-draining and a draining LN compartment, we used recruitment rates ρ_{ndLN} and ρ_{dLN}, where $\rho_{\mathrm{dLN}} + \rho_{\mathrm{ndLN}} = \rho_{\mathrm{LN}}$ in the absence of infection. The increase of ρ_{dLN} post infection was estimated based data from Iwasaki's group [130, 92]. Finally, the parameter α denotes the T cell activation rate in draining LNs, and the quoted value is based on an estimate by Mempel et al. [105] of T cell activation times.

infection (p.i.). Even when the infection was given together with an antibiotic that limits T cell priming to the first three days p.i., the percentage of recruited T cells was still >75%. To determine whether pure random T cell circulation is consistent with these results, we assumed that the immune response to this blood-borne bacterial infection was largely formed in the spleen. In the steady state distribution of our model, the spleen harbors on average 23% of the T cells. With default parameters, the model predicts that 90% of the circulating T cells have visited the spleen by day 3 p.i. (Figure 3.2A). Hence, randomly circulating T cells arrive at the spleen quickly enough to support activating most of the clone by day 3, provided that activation itself is efficient and timely.

For a *local* infection, the picture looks considerably different: Since the draining LNs (dLNs) form only a small fraction of all LNs, pure random circulation would lead to very long cellular arrival times. Rodents typically have around 35 LNs [84]. For example, consider a local infection with 3 dLNs. One would expect a fraction of only $1.5/2.5 \times 3/35 \approx 4\%$ of the cells leaving the blood to enter the dLNs. Consequently, merely 1/3 of all T cells would have arrived by day 7 p.i. (Figure 3.2a). However, van Heijst et al. [153] demonstrated that even for an intranasally administered influenza infection, 2/3 of the specific T cells get recruited into the immune response. To explain these data we must take LN enlargement into account. Iwasaki's group [130, 92] has obtained excellent data on LN enlargement: In an HSV-2 infection model, or even by merely injecting CpG, they found that dLNs increased up to 10-fold by day 4-5 p.i., with the total number of T cells in the dLNs by that time surpassing that of the spleen [130]. We incorporated this information into our model by splitting the lymphoid compartment into a draining and a non-draining part with corresponding recruitment rates ρ_{dLN} and ρ_{ndLN}. In the absence of infection, we set $\rho_{\mathrm{dLN}} + \rho_{\mathrm{ndLN}} = 1.5/\mathrm{h}$ like in our previous model, and LN enlargement was modeled by increasing ρ_{dLN} as a function of time. Soderberg et al. [130] showed that LN enlargement is roughly linear between day 1 and day 4-5 p.i. Thus, we held ρ_{dLN} constant before day 1 and after day 4.5, and increased it linearly in between. Increasing ρ_{dLN} from 0.05/h to 0.8/h in this fashion (with ρ_{ndLN} kept at $1.5/\mathrm{h} - 0.05/\mathrm{h} = 1.45/\mathrm{h}$), we obtained a good fit to the HSV-2 LN enlargement kinetics by Soderberg et al. [130] (Figure 3.2B). With these physiologic LN enlargement kinetics, our model predicts that 66% of all T cells have visited the dLNs by day 4-5 p.i. (Figure 3.2A), which is consistent with the findings of van Heijst et al. [153] for the influenza infection. Because the magnitude of LN enlargement is presumably highly variable, we performed additional simulations with varying increases of ρ_{dLN}. These simulations predicted that half of the circulating cells are recruited before day 5 as long as ρ_{dLN} increases to 0.2, i.e., 20% of the recruitment to the spleen, by day 4.5 (Figure 3.2C).

More recently, Stock et al. [144] determined the kinetics of T cell recruitment in a localized HSV-1 infection indirectly by measuring the depletion of antigen specific T cells from non-draining LNs. To determine how well these data are in agreement with our model, we performed simulations where we assumed that specific T cells are retained in dLNs, and thus effectively removed from the circu-

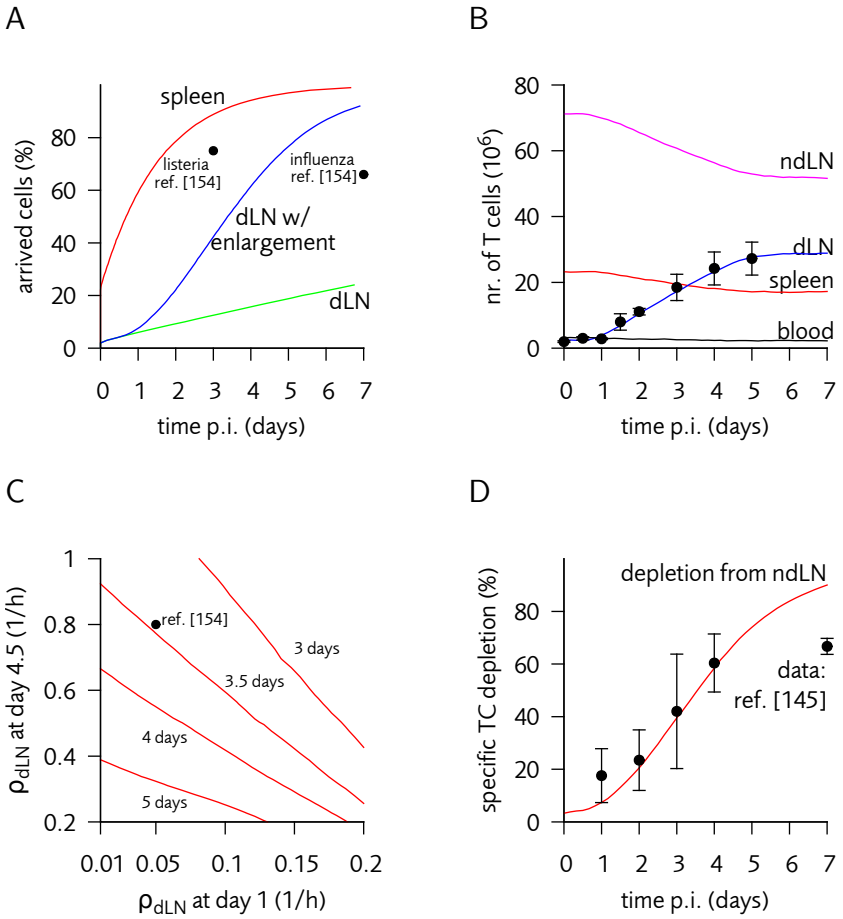


Figure 3.2: **T cell arrival kinetics.** All simulations use standard parameters (Table 3.1, text) unless otherwise stated. **(A)** T cell arrival at the spleen (red) and draining LNs (5% of the LNs); green: without LN enlargement; blue: with enlargement as described in the text and shown in (B). Dots indicate the percentage of specific T cells that according to van Heijst et al. [153] get recruited into the immune response. **(B)** Detailed kinetics of the cell distribution in our simulations following a localized infection. LN enlargement was set to fit the data of Soderberg et al. [130], who counted the number of T cells in dLNs following intravaginal HSV-2 infection (dots with error bars denoting standard error of the mean). Our simulation results were converted into cell numbers assuming that a mouse harbors roughly 10^8 T cells, a common estimate [15]. **(C)** Isoclines of the mean T cell arrival time at dLNs as a function of the rate parameters for LN enlargement. The dot indicates the default parameters used in (A), (B) and (D). **(D)** Depletion of specific T cells from non-draining LNs according to Stock et al. [144] (dots and error bars) compared to the prediction of our model (line). The depletion at day 7 p.i. is lower than predicted because our model assumes that T cell priming goes on indefinitely after infection.

lation. These simulations predicted depletion kinetics that are in good agreement with the data of Stock et al. until day 5 p.i. (Figure 3.2D), with the depletion at day 7 p.i. being significantly lower in the data than in the model. This discrepancy is probably because T cell priming (and thus trapping in the dLNs) stops at some point after the infection. Judging from Figure 3.2D, this should occur after 4-5 days p.i., which is exactly what Stock et al. [144] estimated. Interestingly, the percentage of T cells our model predicts to arrive in the dLNs by day 4 is close to 66%, which is also the percentage of specific T cells that van Heijst et al. [153] found to be recruited into the localized influenza infection. Taken together, these results show a good mutual agreement between our model, the barcoding data, the LN enlargement kinetics, and the T cell depletion kinetics.

3.2.3 Modeling T Cell Activation in Lymph Nodes

To link the macroscopic kinetics investigated so far to microscopic T cell migration, we next augmented our model with a simple representation of T cell activation in LNs. Two-photon microscopy has provided invaluable information on this subject [17], which lead to the current view that T cells scan dendritic cells (DCs) in the LN paracortex in a mostly random fashion [156]. It was estimated that T cells screen up to 150 DCs per hour in this manner [11], with 100 of these contacts being unique. Indeed, mathematical results from random walk theory (Section 2.2.1) can be used to show that the 3D random walk is a simple, but quite efficient search process: Imagine a cell crawling through the LN in such a highly systematic fashion that no DC is ever visited twice. This sophisticated cell would on average visit only 50% more DCs per hour than a randomly moving one. Accordingly, a simple but quite accurate way to model DC screening is to add a single additional parameter α, denoting the activation rate per hour (Table 3.1). One can envisage the augmented model as follows: Within a dLN, the T cells flips a biased coin every hour, with α being the probability of head coming up; if this happens, the T cell gets activated. If the T cell fails to get activated by the time it reaches the exit of the LN, it egresses and returns to the circulation just like in the basic model.

Recent two-photon results provide a basis for estimating α. In two elegant studies, Mempel et al. [105] and Henrickson at al. [65] investigated the activation kinetics of adoptively transferred T cells whose LN entry had been "synchronized" by injecting an integrin blocking molecule shortly after transfer. Their findings suggest a three-phase model of T cell activation: In phase one, the T cell makes transient contacts to DCs which last a few minutes, but continues migrating. In phase two, the cell establishes a long-lasting contact with a single DC that lasts several hours. In phase three, the cell detaches from the DC and migrates away rapidly, presumably in search for a LN exit site. We mapped this three-phase process to our model by setting α to the inverse of the duration of phase one, which based on the data by Mempel et al. [105], gives $\alpha = 1/8h$. This defines T cell activation as the initiation of a long-lasting contact, which prevents the T cell from exiting the LN prematurely.

3.2.4 A Trade-Off Constrains Lymph Node Residence Times

By taking the stochastic nature of T cell activation into account, it becomes clear that the LN residence time is subject to a trade-off with respect to the timely detection of localized infections. If a T cell remains too long in a given LN, the risk arises that an infection may occur somewhere else, where the cell would be needed for defense. Conversely, if the cell egresses too quickly from its present LN, it might fail to detect cognate antigen there. Thus the following question arises: Which would be a good range of residence times that balance between these risks?

To investigate this question, we simplified our model[1] by holding the compartment residence times constant (Section 3.4.3). This allowed us to clearly determine the efficiency of *specific* residence times, which establishes a basis for reasoning about the efficiency of physiologic *ranges* of residence times. Simulations of the simplified model indeed predict that overly long or short residence times lead to delays in T cell activation upon infection. For example, a residence time $R_{\mathrm{LN}} = 13.5h$ leads to activation of 60% of the specific T cells around day 4 p.i., whereas for $R_{\mathrm{LN}} = 72h$, merely 40% of the cells would be activated by day 7 p.i. (Figure 3.3A). This relation can be characterized by considering the expected activation time $E[A]$ as a function of R_{LN} (Figure 3.3B).

To gain a deeper understanding of the dependency between R_{LN} and $E[A]$, we derived the theoretically *optimal* LN residence time R_{opt} (Section 3.4.3). Surprisingly, this analysis revealed that R_{opt} is hardly dependent on the dLN recruitment rate ρ_{dLN}, and our simulations confirm this theory (Figure 3.3C). This leads to an interesting conclusion: Evolution can *independently* optimize LN enlargement and LN residence time; i.e., evolving a mechanism that benefits LN enlargement should hardly affect the efficiency of LN residence times. The hypothesis that evolution has indeed optimized LN enlargement is supported by the fact that several independent pathways for LN enlargement exist [92]; whether LN residence time has also been subjected by evolution, however, is less clear.

To shed some light on this question, we calculated R_{opt} a function of the T cell activation parameter α. We found that LN residence times of 7-20h, as used by Stekel et al. [134] and estimated in different thoracic cannulation experiments [51, 161], are consistent with T cell activation times of about 5-30h (Figure 3.3D). Strikingly, the T cell activation times (i.e., the duration of phase one in the three-phase activation model) estimated by Mempel et al. [105] and Castellino et al. [25] indeed fall within this range. However, Henrickson et al. [65] observed long-lasting contacts already 1.5h after injection of DCs pulsed with an engineered high-affinity peptide. The corresponding optimal residence time would lie outside the usually estimated range, which suggests that this experimental setup may not reflect typical antigen presentation in localized infections.

Finally, we investigated the effect of *transient lymph node shutdown* [24, 103, 20] on the residence time trade-off. Transient LN shutdown inhibits cell egress

[1] This simplification leads to the model studied in the previous chapter.

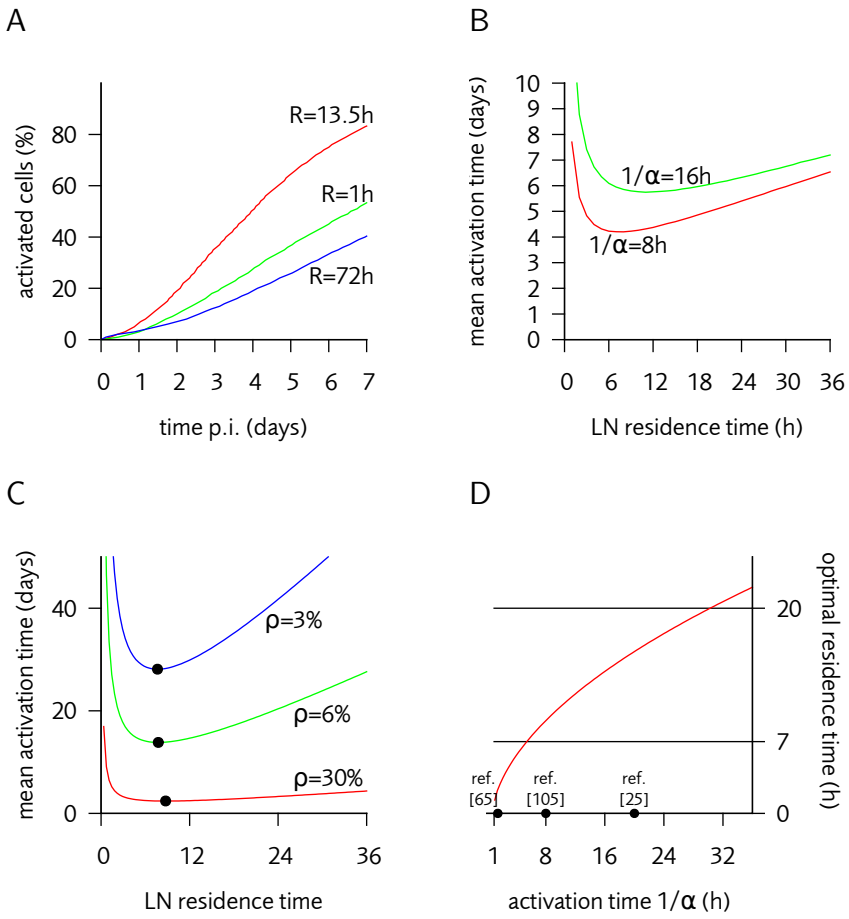


Figure 3.3: **A trade-off constrains the lymph node residence time.** **(A)** T cell activation kinetics for a localized infection with default LN enlargement and default activation rate $\alpha = 1/8h$ in the dLN. The default residence time predicts a reasonably fast activation (>50% by day 4 p.i.), while a pathologically low (1h) residence time would lead to slower activation because T cells exit prematurely from dLNs. On the other hand, a pathologically large residence time (72h) would slow down T cell transfer from the ndLNs to the dLNs. **(B)** Trade-off curves for LN residence time versus mean T cell activation time. **(C)** Via mathematical analysis (Section 3.4.3), we determined the theoretically *optimal* LN residence time, which surprisingly revealed that the location of the optimum is hardly dependent on dLN recruitment rate (Section 2.5). However, note that overly large residence times are penalized more when the dLN recruitment rate is lower. **(D)** Physiologic LN residence times of 7-20h, which is the range used by Stekel et al. [134] and estimated in several cannulation experiments [51, 161], are consistent with near-optimal detection of localized infections when T cell activation time lies between roughly 6-30 hours. The dots indicate experimental estimates by Henrickson et al. [65], Mempel et al. [105], and Castellino et al. [25] of the duration of T cell activation. The estimate by Henrickson et al. [65], which lies outside the consistent range, was obtained by injecting DCs pulsed with an engineered high-affinity peptide.

from dLNs during the first 12-18h p.i. We suspected that shutdown might counter the effect of overly short residence times, because the few specific T cells that are present in the LN at the time of infection are retained, which could make T cell residence times in the absence of infection less relevant. However, T cell residence times go back to normal soon after the shutdown period to prevent that the dLNs become clogged with non-specific cells. According to our predictions above, the large majority of T cells should arrive after the shutdown period. Indeed it turned out that the trade-off curve is hardly affected when transient LN shutdown is incorporated into the simulation, and the expected hitting time decreased by less than 1%. Hence, adjusting recruitment efficiency does *not* appear to be the main purpose of transient LN shutdown. Recent results of Kumamoto et al. [92] demonstrate a pathway of LN enlargement that depends on specific T cells; this could give the few antigen specific T cells that are present in the dLN of the time of infection a very important role, and the role of shutdown might thus be to accelerate LN enlargement.

In summary, we can state that physiologic LN residence times are near-optimal for timely control of localized infections with respect to physiologic T cell activation times. Moreover, note that the optimum is not a very "sharp" one (Figure 3.3C), such that a large range of LN residence times is admissible. This is consistent with the hypothesis that T cells transit lymphoid organs by random migration, which leads to a rather wide distribution of residence times.

3.3 Discussion

In this chapter, we have proposed a model of T cell circulation, recruitment and activation reflecting new information that has become available on these subjects during the past decade. In particular, we have verified that random compartment transit, which appears to be the default mode of T cell migration in SLOs in the absence of antigen [106, 156, 9, 147], is consistent with realistic steady-state T cell distributions across organs. Moreover, we investigated recruitment kinetics for localized infections and the role of LN enlargement in this process, and revealed a quantitative connection between T cell activation kinetics and LN residence time. Our findings emphasize that a two-scale approach is necessary to fully understand how the immune system detects and clears antigen in a timely and efficient manner.

The following anatomical factors were found to have key roles in efficient antigen detection. (1) The spleen provides a central hub for systemic infections, and its huge cell turnover yields T cell arrival kinetics that by far outperform those of LNs. Patients without a spleen are facing an increased risk of bacterial infections, one important reason being that the spleen harbors a unique B cell population that produces pentameric IgM antibodies, which are important to clear encapsulated bacteria like *Streptococcus pneumoniae* [39]. Our results indicate that the slower T cell recruitment after spleen removal might be another, currently unappreciated factor contributing to the partially impaired immunity in splenectomized patients.

(2) LN enlargement is crucial for clearing localized infections. Without LN enlargement, comprehensive T cell recruitment to the draining area would be a matter of weeks and not days. Accordingly, recent data show that LN enlargement is mediated by several independent, both innate and antigen-specific pathways [130, 92]. (3) The three-dimensional LN microenvironment supports a simple and yet highly efficient mechanism for T cell antigen screening. Mathematical considerations (Section 2.2.1) show that a 2D environment would be far less effective,

Our model may help resolve some contradictions between experimental data. First, the time required for T cells to establish long-lasting contacts to DCs varies greatly between different experimental systems [17]. Our results show that some of these results, which predict activation times of $<$2h, are not in agreement with the assumption that T cell residence time in LNs is tuned for timely detection of localized infections. Thus, these data may be less likely to reflect the average physiologic situation. Second, it has been difficult to establish the kinetics of T cell blood-to-lymph transit. While earlier experiments indicated a transit time of roughly 24 hours and an LN residence time of 12-18 hours [51, 161], which is consistent with T cell turnover rates measured in tissue [62], the data with the highest recovery rate so far of $>$90% [160] suggest a much larger mean blood-to-lymph transit time of about 30 hours. Our predictions indicate that such a large residence time would yield suboptimal T cell activation. One explanation for this discrepancy is the hypothesis of Stekel et al. [134] that T cell recirculation kinetics are density-dependent; this would cause the blood-to-lymph transit to slow down over the course of a drainage experiment. Stekel's hypothesis appears reasonable because drainage of T cells and lymph fluid presumably deflates the LN architecture and slows down the flow of the remaining lymph. This effect should be strongest in the most prolonged drainage experiments with the highest recovery rates.

Our mathematical analysis of the optimal residence time bears some similarity to the subject of "Optimal Foraging Theory" [135], in which optimal migration patterns of foraging animals in patchy environments (think picking apples from trees) are investigated. This theory has received ample and partly harsh [118] criticism for some of its assumptions, most notably (1) that the foraging animal is able to determine the concentration of food in its current environment, and make informed decisions accordingly; (2) that the spatial distribution of food is stable on a long enough timescale for evolution to optimize the animal's decisions accordingly. We would like to point out that in contrast to this well-established class of models, the assumptions of ours are much weaker: (1) we do not assume that the T cell "knows" whether its cognate antigen is present in a dLN upon entry (otherwise there would be no need for DC screening in the first place); and (2) we do not assume that single T cells can make informed decisions on when to leave their current LN. Instead, we assume that the residence time is determined globally, e.g. by choice of LN size. Since the parameters of our model are mostly determined by anatomical factors, it appears reasonable to hypothesize that LN residence times have been subjected by evolution.

We based most of our model parameters on rat data, because rats were the

most widely used animal model in classic lymphocyte migration studies. Most of the modern two-photon results, however, are obtained in the mouse model, so one limitation of our current approach is the assumption that lymphocyte migration in tissue is similar in mice and in rats. However, many aspects of lymphocyte migration and distribution are known to be similar in most animal models [158], which suggests that our model should be easily adaptable to other species. Furthermore, most of the recirculation and migration patterns discussed here apply also for B cells, even though the kinetics are different. In this sense, a similar model as the present one could be derived for B cells once B cell recirculation kinetics become better understood than they are at present.

3.4 Methods

3.4.1 Modeling Compartment Transit by Random Walk

We represent the three compartments of our model – spleen, LNs, and dLNs – as one-dimensional intervals $[0,L] = \{x \in \mathbb{R} : 0 \leq x \leq L\}$. We assume that T cells entering these compartments are released at position $x = L/2$, and migrate via Brownian motion with unitary diffusion coefficient $D = 1$. Although T cell motility is better described as a persistent random walk on a timescale of minutes [106], the Brownian motion approximation should be sufficiently accurate for residence times on the order of hours [147]. Once the cell reaches either end of the compartment, it moves back into the blood. Thus, the probability distribution p for the cell position is described by the differential equation

$$\frac{\partial}{\partial t} p = \frac{\partial^2}{\partial x^2} p \tag{3.1}$$

subject to the initial condition

$$p(x,0) = \delta(x - L/2) \ , \ x \in [0,L] \ , \tag{3.2}$$

where $\delta(x)$ is the Dirac delta distribution, and the absorbing boundary conditions

$$p(0,t) = p(L,t) = 0 \ , \ t \in [0,\infty) \ . \tag{3.3}$$

This differential equation can be solved by standard methods [46]. The solution can be expressed as a Fourier sine series

$$p(x,t) = \sum_{k=0}^{\infty} a_k \cdot \exp\left(-\left(\frac{k\pi}{L}\right)^2 t\right) \cdot \sin\left(\frac{k\pi}{L} x\right)$$

with coefficients

$$a_k = \frac{2\sin(k\pi/2)}{L} \in \left\{-\frac{2}{L}, \frac{2}{L}\right\} \ .$$

By integrating the solution over $x \in [0,L]$, we obtain the function $\phi(t)$ giving the probability that the cell is still in the compartment at time t:

$$\phi(t) = \int_{x=0}^{L} p(x,t)\,dx = \sum_{k=0}^{\infty} a_k \cdot \frac{2k\pi}{L} \cdot \exp\left(-\left(\frac{k\pi}{L}\right)^2 t\right) .$$

Now let R denote the residence time in the compartment, then $1-\phi$ is the probability distribution of R. With some algebra[2], one can show that

$$\mathrm{E}[R] = \int_{t=0}^{\infty} t \cdot (1-\phi(t))'\,dt = \frac{L^2}{8} .$$

Hence, by setting the compartment length L to $\sqrt{8R}$ one obtains a mean compartment residence time R.

3.4.2 Stochastic Simulation of T Cell Circulation

Our stochastic simulations alternate between two different modes: (1) the recruitment of cells from the blood to a lymphoid organ; (2) the transit through the lymphoid organ. In mode (1), a cell starts in the blood (B) at time $t = 0$. We define the recruitment to lymphoid organs via the following master equation for the probability $B(t)$ that the cell is still in the blood at time t:

$$\frac{dB}{dt} = -\rho_{\mathrm{LN}} B - \rho_{\mathrm{dLN}} B - \rho_{\mathrm{S}} B .$$

This equation is identical to the corresponding one in the Stekel model [134], except for the additional term $-\rho_{\mathrm{dLN}} B$. The duration of phase (1) and the target compartment are determined by a simple stochastic simulation of this equation using the kinetic Monte Carlo method [47].

In mode (2), the cell transit through a compartment is modeled as described above using the initial-boundary-value problem defined by Equations 3.1-3.3. We simulate these equations stochastically in a standard way [110] via a discrete-time random walk with step increments drawn from an appropriately scaled normal distribution.

3.4.3 Simulating T Cell Activation

Based on the circulation model outlined above, we formulated a simplified discrete-time stochastic model of localized infections that is defined by the following simple rules; see the previous Chapter 2 for a detailed derivation. (1) A cell transits a LN in exactly R hours. (2) Upon egress from a LN, it takes exactly R hours to travel to the next one. T represents the time spent on average in the spleen and in the blood between two LN visits. (3) In each round, the cell enters a dLN with

[2] The key ingredient to this calculation is the identity $\sum_{n=0}^{\infty}((-1)^n/(2n+1))^3 = \pi^3/32$.

constant probability ρ and a non-draining LN with probability $1-\rho$. In terms of the parameters of our circulation model,

$$\rho = \frac{\rho_{\mathrm{dLN}}}{\rho_{\mathrm{dLN}} + \rho_{\mathrm{ndLN}}} \, .$$

(4) Per hour spent in a dLN, the cell is activated with probability α. For some mathematical justification of step (4), see Section 2.2.1 in the previous chapter.

Thus, the two major simplifications of this model are that (1) T is a constant rather than a random variable, and that (2) ρ is a constant rather than a function of time. The fact that R is constant is not a simplification but a desired property, as our goal is to determine the efficiency of specific residence times. We set $T = 5.5$h according to the mean time between two consecutive LN visits in our stochastic simulations with default parameters without LN enlargement. Taking LN enlargement into account would result in a small decrease of T. The optimal residence time was determined according to Equation 2.21 in Chapter 2.

Part II

Thymic Negative Selection

Chapter 4

An Algorithmic Model of Thymic Negative Selection

Summary. *Thymic negative selection generates a population of T cells that tolerate their host organism (self) and attack only foreign, potentially dangerous substances (nonself). We re-consider an established algorithmic model of negative selection, which we formalize using the notational framework of algorithmic learning theory. The model can be understood as a classifier trained on negative examples, which is based on a pattern matching rule that models antigen recognition by T cell receptors. We discuss how inexact matching enables negative selection algorithms to generalize beyond input data, and point out a previously unappreciated sampling-based generalization mechanism.*

Although negative selection algorithms have been successfully applied to biological research questions, their exponential runtime has limited their practical usefulness. We thus investigate methods to compute the classification outcome efficiently. We precisely characterize the computational complexity of this task based on properties of the matching rule, and show that for some matching rules there exists no polynomial time solution unless P=NP. However, by taking advantage of techniques from string processing and data compression, we obtain polynomial time algorithms for some important matching rules, including the "r-contiguous" rule for which negative selection had been conjectured to be NP-hard.

Our results demonstrate the usefulness of methods from learning theory in studying the immune system. We hope that this will form a starting point for a comprehensive learning theoretical approach to the immune system, which despite being one of the two important learning systems of the human body, has so far received little attention from learning theorists.

4.1 Introduction

In his PhD thesis on negative selection algorithms, Thomas Stibor came to the conclusion that "future work in this direction is not meaningful" [136, p. 118]. Then

why do we devote the largest chapter of this thesis to the study of precisely such algorithms? Historically, negative selection algorithms were designed for providing better protection to computer systems and networks [55]; Stibor argued that existing machine learning techniques would probably be a better choice for this application. Here however, our perspective is different: we use the negative selection algorithm as an algorithmic model of negative selection in the real immune system, whose purpose is to derive testable predictions from formalized biological assumptions. One example use case, which boils down to a one-class classification problem, will be demonstrated in the next chapter: We are given an input dataset S consisting of strings of length 6 over the amino acid alphabet – these represent the protein chunks (peptides) that are normally present in the human body. Moreover, we are given another input dataset X containing peptides from the HI virus. Now, the task is to predict which of the strings in X will be recognized by the immune system's T cells. In the next chapter, we will use the results of the this chapter to solve this problem, and demonstrate that the prediction is indeed significantly correlated with the true recognition status (which is known for the HI virus).

Now, it is crucial to note the following. Of course we could also try to solve our classification problem with a standard machine learning technique – say, a one-class support vector machine (SVM) with a string kernel [126]. Presumably, with enough tuning, the SVM prediction could yield at least as significant results as our negative selection algorithm. Suppose, for instance, that an SVM would be capable of predicting recognition status accurately in 99% of the cases. Now the immunologist would ask: what does this result tell me? We would answer: it tells you that, if the strings in S and X are mapped to a multidimensional space via a custom-designed kernel function, then the recognized viral peptides can be roughly linearly separated from the non-recognized ones and those in S. Now suppose we could achieve a 70% accuracy with a negative selection algorithm. This would tell the biologist that some of the viral peptides cannot be recognized *because the T cells that could potentially recognize them are all deleted during negative selection.* We hope that this example illustrates why, for the purpose of theoretical immunology, we are *not* able to substitute the negative selection algorithm by an arbitrary machine learning technique – the latter would give us purely data-driven (phenomenological) predictions rather than hypothesis-driven (mechanistic) ones, which would provide only very limited biological insight.

In summary, for the time being our goal is *not* to devise a general-purpose classification algorithm that competes with one-class SVMs or kernel density estimators. Nevertheless, as a side-product, our analysis provides the first negative selection algorithms that run in worst-case polynomial time, and thus we do solve one of the problems that so far prevented the application of negative selection algorithms large-scale machine learning problems [136]. For our purposes, these techniques make it possible to perform the quantitative predictions in the next chapter with reasonable computational resources (i.e., in a matter of minutes rather than days).

4.1.1 Organization of this Chapter

In the upcoming Section 4.2, we will briefly explain the immunology behind negative selection. In Section 4.3, we will introduce an algorithmic model of the negative selection process that was proposed by Forrest et al. [55], and describe applications of this model in the field of artificial immune systems and in computational immunology. Section 4.4 will give formal definitions of general and specific negative selection algorithms using concepts and notation from algorithmic learning theory. Based on these definitions we will then investigate the ways in which negative selection algorithms generalize beyond input data (Section 4.5).

The remainder of the chapter will be concerned with the computational complexity of negative selection algorithms – or more precisely, the complexity of the *functions computed by* such algorithms. We start in Section 4.6 with some general techniques and theorems that provide upper and lower complexity bounds. After that, in Section 4.7, we present a technique that can often be used to implement efficient negative selection classifiers efficiently, namely *pattern compression*. Section 4.8 then wraps up the chapter with a short discussion. In the epilogue (Section 4.9), we discuss the implications of our results for future research on negative selection algorithms as general-purpose machine learning techniques.

4.2 A Brief Primer on Thymic Negative Selection

The purpose of thymic negative selection is to generate a population of immune cells that are capable of performing cognate *self-nonself-discrimination*. The paradigm of self-nonself-discrimination is today the most widely accepted theory of immune system function. A computer scientist once described it as a "nice hack of nature" [86]: Since there is no simple way to discriminate benign or harmless intruders (e.g., a donated organ or pollen) from malign ones, nature resorts to self-nonself-discrimination as an approximation. Thymic negative selection produces immune cells that will tolerate anything self, and attack anything nonself. In the following, we give a brief overview of the immunological principles related to negative selection.

4.2.1 T cells and Their Functions

Even though the immune system is a cognitive system, it functions independently of the brain and the central nervous system. The two main cell lines that perform cognitive tasks in the immune system are the *B cells*, named after the *bursa of fabricius* where they develop in birds[1], and the *T cells*, which develop in the *thymus*. B and T cells are present in the body in almost equal numbers, and perform different functions in the immune response. The entire set of B and T cells that an individual possesses is called the *repertoire*. In the scope of this thesis, we will

[1] Coincidentally, B cells in humans develop in the bone marrow.

restrict our attention mostly to T cells, and thus we shall refrain from a detailed discussion of B cell functions; the interested reader is invited to consult an immunology textbook like the one by Janeway et al. [79] for further reference, or perhaps Hofmeyr's "interpretative introduction to the immune system" [70], which is written for engineering and computer science audiences.

T cells interact with antigen via *T cell receptors* (TCRs), which are anchored into the cell membrane. All TCRs carried by a given T cell are identical. The molecular structures recognized by TCRs are called *epitopes*. The probability that a given epitope will bind to a given TCR, called *precursor frequency*, is very low. For example, the T cell precursor frequency in mice has been estimated to $10^{-5} - 10^{-6}$ [15, 108]. Therefore, to provide reliable protection against the huge potential number of pathogens, the T cell repertoire must be very large and diverse. Evolution has found a simple and powerful way to achieve this: T cell receptors are generated *randomly* by stochastic rearrangement of DNA fragments. It was estimated [79, p. 151] that 10^{18} different TCRs can be produced in this fashion, which is several orders of magnitude larger than the number of basepairs in the human genome ($\approx 3 \cdot 10^9$ [94]). Altogether, a repertoire of about 10^9 T cells is generated in a mouse, such that an antigen should be detected by 100 T cells on average – a number which of course varies considerably for different antigens [108, 113].

T cells do not interact directly with foreign entities like by bacteria or viruses. They need to have the antigen *presented* to them in form of small chunks (peptides) bound to *MHC molecules* on other cells. This may be compared to our eating habits: we need to process our food in some way, e.g. by chopping and cooking, before we can eat it – the MHC molecule can be seen as an analogue to a plate on which food is served. There are two different pathways for antigen presentation that are each tailored to one T cell subset. T cell subsets are phenotypically defined by their expression of different surface molecules, called *CDs* (clusters of differentiation). Of particular importance are two CD molecules associated with different antigen presentation mechanisms – CD8, which is associated with the *MHC class I pathway*, and CD4, which is associated with the *MHC class II pathway*.

4.2.2 The MHC Class I Pathway

The MHC class I molecule is expressed by every nucleated cell in the body, i.e., by practically all cells except most importantly red blood cells. Peptides presented on MHC class I are of *intracellular* origin. These peptides are derived from proteins in the cytosol via the following pathway (Figure 4.1): Proteins in the cytosol are continuously degraded (chopped into small pieces) in the *proteasome*, a molecule that functions as the "waste recycling facility" of the cell. The protein *TAP* transports peptides from the cytosol into the lumen of the endoplasmatic reticulum (ER), where they are loaded onto MHC class I molecules. In conjunction with peptides, MHC class I molecules form stable peptide-MHC conjugates (pMHCs), which are anchored into the cell surface. The peptides presented on MHC class I are about

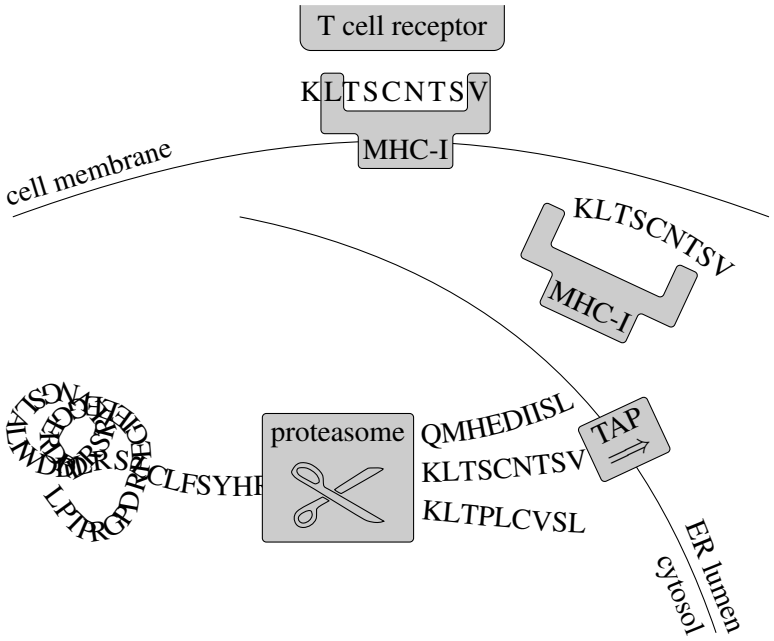


Figure 4.1: The MHC class I pathway for antigen presentation to CD8 T cells.

8-10 amino acid residues long, which apparently provides enough information[2] for discriminating self from nonself [21].

In this fashion, the MHC class I molecule provides a "window" through which a CD8 T cell can see the interior state of a cell: In a normal physiological situation, the peptides presented on MHC class I are just products of the cell metabolism. If abnormal peptides are presented, this can indicate for example that the cell has been infected by a virus and is now synthesizing viral proteins. The T cells that screen MHC class I react to such a nonself signal by killing the screened cell, or more precisely, by instructing it to undergo programmed cell death. CD8 T cells are therefore called *cytotoxic T lymphocytes* (CTLs), or *T killer cells*.

4.2.3 The MHC Class II Pathway

The MHC class II molecule is only expressed by a few cell types including macrophages, dendritic cells, and also B cells. MHC class II is dedicated to the presentation of peptides of *extracellular* origin, e.g., peptides originating from bacteria or parasites. The cells that carry MHC class II are so-called *antigen presenting cells* (APC), which can internalize proteins or even entire microorganisms and digest them using lysosomes in the cytosol. The MHC class II molecule binds to diges-

[2]Since there are 22 amino acids, this is comparable to a string of the same length in the Latin alphabet. Coincidentally, a sample string of this size is often enough to discriminate between human languages [40].

tive products of the original proteins, which may be up to 25 amino acid residues long, and forms pMHC complexes with these peptides. The pMHCs then migrate to the cell membrane where they are presented to T cells.

In contrast to a cytotoxic CD8 T cells, the CD4 T cells do not take action against the cell that is presenting the antigen (which would make little sense indeed), but rather activate other lymphocytes such as B cells and instruct them to act against the infection. For this reason, CD4 T cells are usually referred to as a *T helper cells*. However, there also exist other types of T cells that express CD4, such as regulatory T cells.

4.2.4 Positive and Negative Selection in the Thymus

Both MHC pathways for antigen presentation ultimately rely on the T cell to decide whether the presented molecule is self or nonself, i.e., whether the immune system should take an action or not. A newborn T cell with a randomly generated receptor is unlikely able to make this decision. How do T cells then "learn" to detect only foreign peptides, while steering clear of self?

The answer does in fact not involve much learning. Rather, T cells that detect nonself are generated by a "trial and error" in the thymus. There, each newborn T cell must prove its usefulness by passing two screening processes. First, during *positive selection*, it is ensured that the cell is capable to bind to MHC molecules and to scan the presented peptide. Second, during *negative selection*, the T cell is confronted with normal peptides from self. Cells that bind too strongly to these self peptides are killed in the thymus, because they would be harmful if allowed to become part of the immune system. Those cells that survive both positive and negative selection are thus capable of screening MHC and will bind only to peptides that are not normally encountered in the host. Remarkably, 97% of the initial T cell population die during this process [128]. Thus, forming a mature and immunocompetent T cell repertoire requires a large amount of resources and energy, and takes several years in a human.

It has to be pointed out that negative selection is very likely not the only mechanism that prevents T cell autoreactivity. First, it is considered unlikely that all proteins that could arise in the human body during the entire lifetime are present in the thymus. Second, it is known that circulating T cells can under certain conditions be *tolerized*, and switch to an *anergic* state in which they no longer respond to their cognitive signals. Third, there is a subclass of T cells called *regulatory T cells* whose role is thought to be the prevention of autoreactive helper or cytotoxic T cell responses. In fact, mutations of the gene encoding for *FoxP3*, a molecule used by regulatory T cells, leads to the rare disease *IPEX* that is associated with severe systemic autoimmune reactions, which are mostly lethal during the first year of life [164]. On the other hand, mutations in the gene AIRE, which partially inhibit negative selection, cause the autoimmune disease APS1 [2], which is severe, but not lethal. These facts indicate that the prevention of autoimmune reactions cannot be solely due to negative selection, a fact that is often overlooked in the literature.

4.3 The Negative Selection Algorithm

Despite the large required effort and energy, random generation of the T cell repertoire combined with negative selection achieves some remarkable properties in a simple fashion:

- *Diversity.* Immune cell repertoires are extremely diverse both at the cell level – a given immune cell is different from most other immune cells in the same organism – and at the population level – each individual carries its own unique set of immune cells. Both levels of diversity jointly make it very difficult for pathogens to circumvent the immune system of a vertebrate species, since in each new host they face a different environment.

- *Robustness.* Because there is no central point of control that coordinates the actions of T cells, there is also no single point of failure: Many lymphoid organs, such as single lymph nodes, the spleen, the mandibular glands, and the appendix can be removed without losing the system function (even though removal of *all* lymph nodes is fatal). This is an evolutionary advantage, because a central point of control would provide an obvious target for pathogens.

- *Protection against unseen pathogens.* A large enough number of randomly generated receptors provides protection against pathogens that have not been encountered previously. The alternative strategy is to produce receptors that are capable of detecting certain specific pathogen-associated molecular patterns (PAMPs), e.g. polysaccharides that occur in the membranes of gram-negative bacteria but do not normally occur in the vertebrate metabolism. Although many cells of the unspecific (or innate) immune system do carry such receptors as a means to quickly react to frequently occurring infections, a host defense depending solely on such mechanisms would be too easily circumvented by evolution of mechanisms to hide the PAMPs.

In the 1990s, when the threat of computer viruses began to be widely recognized as a major economical and infrastructural threat to the dawning network age, Forrest and coworkers [55, 54, 53] looked at these features of the immune system from a computer scientist's perspective and found that in comparison, the software employed to protect computers from security threats looked very immature. Computer security systems like firewalls and signature based virus scanners had single points of failure, lacked the capacity to react to threats they had not specifically been programmed for, and were extremely homogeneous on a global scale – much of this is still true at the time of writing. Could the immune system be exploited as a source of inspiration for better protection of computers and networks?

In a first attempt to import an immunological paradigm into the computer security domain, Forrest identified thymic negative selection as a key process that facilitates many of the important desirable features of the real immune system.

Moreover, compared to other immunological phenomena, negative selection is relatively well understood on an abstract conceptual level. These and other reasons lead Forrest to propose an algorithmic framework called the *negative selection algorithm*, which we adopt here as a model of negative selection in the real immune system.

The basis of a negative selection algorithm is a *shape space*, i.e., an abstract representation of T cell receptors and peptides. For example, one could represent both receptors and peptides as points in a 2-dimensional space (Figure 4.2). Then one defines a *matching rule* that determines whether a given receptor will bind to a given epitope. For our two-dimensional space, the matching rule could simply state that a receptor and an epitope match if their distance is lower than some threshold (see middle panel in Figure 4.2).

The input of the algorithm is a set of self-peptides (which correspond to those that T cells see in the thymus), and another set of peptides (often called *monitor set*) that are to be classified as either self or nonself (corresponding to the peptides that T cells see later in the tissue). The algorithm then proceeds in two phases:

1. *Training phase:* A set of receptors is generated randomly, and each receptor is matched against all self-peptides. All receptors that match one or more self-peptides are deleted.

2. *Classification phase:* The peptides in the monitor set are matched against the remaining receptors. A peptide that is matched by at least one receptor is classified as *nonself* (which would correspond to an anomaly), and otherwise as *self*.

The algorithm works for all shape spaces and matching rules in the same fashion. More complex matching functions have been designed to mimic the way that T cells recognize their epitopes, which still is not completely understood. Notably, the *r-contiguous bits rule* designed by Percus, Percus, and Perelson [116] was adopted in the first negative selection algorithms and has played an important role ever since [139], even though the fact that it was (back in 1993) deemed biologically plausible does not necessarily make it appropriate for use in a computer security setting. The *r*-contiguous bits rule and several other matching rules will be defined in Section 4.4.3.

4.3.1 Use in Theoretical Immunology

Boldly speaking, all attempts to design a productive computer security system based on the negative selection paradigm failed, and some reasons why this happened will be explained in more detail later on. However, within the scope of this thesis, we are more interested in the model from a theoretical immunology perspective: how accurately does the model reflect negative selection in the real immune system, and can we derive predictions from it that can be tested in real

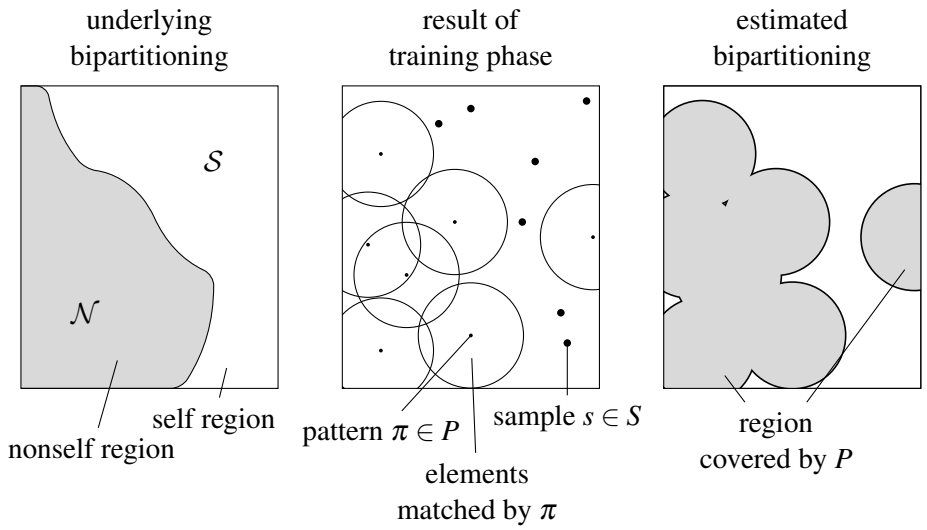


Figure 4.2: Example negative selection algorithm where both T cell receptors and peptides are represented as points in a two-dimensional space, and a receptor is defined to bind all peptides within a fixed distance. The space is assumed to be pre-partitioned in two regions $\mathcal{S}$ (self) and $\mathcal{N}$ (nonself; left panel). The input to the algorithm is a set of samples from the self region (thick dots in middle panel), which represent the self peptides presented to T cells in the thymus. In the training phase (which corresponds to negative selection in the thymus), receptors are generated randomly and those that match any of the given samples are deleted. This results in a set of receptors (thin dots in middle panel) that do not match any of the input points (circles in middle panel). This negatively selected receptor set is then used to classify further elements (right panel). If the training phase is successful, the region covered by the generated receptors (gray region in right panel) should be similar to the underlying nonself region (gray region in left panel). Figure adopted from Elberfeld and Textor [43].

data? And more generally, how does the negative selection algorithm relate to theoretical models of other learning processes?

In the literature, one finds a few notable examples of similarly motivated applications of the negative selection algorithm. Detours and Perelson [37] showed that T cell alloreactivity[3] can be explained as a consequence of negative selection. Košmrlj et al. [90] used the algorithm with a refined matching function based on biophysical amino acid properties. They predicted that certain MHC alleles can lead to a rather narrow window for T cells to survive negative selection, and interestingly, they found that this effect may be a possible explanation for elite control of HIV infection [91]. This was perhaps the most successful application of a negative selection algorithm so far. Furthermore, negative selection algorithms are routinely used as components of complex agent-based simulations of entire immune responses, e.g. by Chao et al. [27, 28] and Rapin et al. [121], to name only a few. Such agent-based models are often of a more qualitative nature in the sense that they reproduce emergent properties of the real immune system (e.g., a larger secondary than primary response), but because of their huge numbers of parameters they are less suited for making quantitative predictions.

4.3.2 Use in Artificial Immune Systems

As explained above, the negative selection algorithm was originally conceived for use in intrusion detection systems. For this application it is key to design an appropriate shape space and matching function, which capture the essential features of the data. An interesting idea pursued by Forrest's group was to use sequences of system calls from normally running UNIX processes to define a sense of self for computers [54, 72]. The receptors (which they called *detectors*) are then essentially patterns that match to small sets of system call sequences. With these detectors one can monitor a running computer system for suspicious activity – for example, unusual sequences of system calls, which may point to an intrusion or a computer virus infection.

This simple algorithmic scheme does indeed achieve some degree of "biodiversity" in computer systems: For example, what is a "normal" sequence of system calls differs widely between hosts and networks, as do the randomly generated detector sets. Forrest identified several further promising properties that would be achieved by computer immune systems built on the negative selection paradigm [53], such as straightforward parallelization – detectors can be distributed among threads, processors, network switches and hosts. Forrest's group built several proof-of-concept software packages such as "process homeostasis" [131], which is based on system call sequences, and "LISYS" [69, 71, 8], which is based on TCP packages. However, despite these initially promising and influential[4] results,

[3] Alloreactivity is the phenomenon that T cells to bind to foreign MHC molecules, e.g. from a donated organ, much more strongly than could be expected.

[4] At the time of writing, some of Forrest's early papers have been cited more than 1000 times.

negative selection based intrusion detection systems never made it beyond the prototype stage.

Two main issues that impeded the progress of negative selection algorithms were identified early on. One of these issues was the large computational cost: The number of detectors needed to achieve acceptable coverage of the potentially anomalous region is often prohibitively large [88]; in other cases, the dataset can be conditioned in a way that makes it difficult to find any detectors at all that do not match any element of the given self-set. While several algorithms were proposed to generate detectors more efficiently than by pure random sampling, all of them still suffered from exponential worst-case complexity. Stibor [143, 139] approached these difficulties by considering the following decision problem, which we will later call the *consistency problem*: Given a self-set S, can any detector be generated that does not match any element in S? While this is indeed exactly the right problem to consider as we will see later, he did not prove the problem complete for any "hard" complexity class. Instead, he showed how to convert instances of the problem to instances of the boolean satisfiability problem in conjunctive normal form (k-CNF-SAT), and then solved the resulting formulas using the DLL algorithm and evaluated its runtime to reason about the complexity of generating detectors. At first, Stibor explicitly acknowledged the limitations of this approach by noting that [143]

> *"this* not *implies [sic] that finding a satisfying set for [the generated formula] is an NP-complete problem."*

Nevertheless, he wrote two years later [137]:

> *"Recently, it has been shown that finding r-contiguous detectors is equivalent [sic] to the problem of finding assignment sets for a Boolean formula in k-CNF. This result explained the lack of efficient algorithms for finding detectors."*

Similarly, in a joint work of his with Timmis, Hone and Andrews published in the journal "Theoretical Computer Science" [149], it is concluded that

> *"the problem of generating detectors in a negative selection algorithm turns out to be equivalent to an NP-complete problem."*

These and other similar statements [136, 138, 139] created an impression in the community that the negative selection *approach* (rather than its existing implementations) suffered from prohibitive computational complexity. For example, Aickelin [1] wrote about negative selection algorithms:

> *"More recently, [...] theoretical proofs have been established regarding their performance: [...] Negative Selection systems can never scale to cover real-world multi-dimensional data [...]. "*

Clearly this was a premature conclusion, and later in this chapter it will be shown that the computational complexity issue can in fact be solved by approaching the problem in a novel way.

The second issue brought up by Stibor was that negative selection algorithms do not generalize well on some datasets that standard techniques, such as kernel density estimation using Bernoulli kernels, handle relatively well [139]. This was due to the pattern matching rule employed in most implementations of the algorithm, the so-called *r*-contiguous bits rule – a rule that was originally designed as a model for the way that real T cells scan peptides presented on MHC [116]. It remains unclear why this rule was deemed appropriate for analyzing network packets or other arbitrary data, but its inability to correlate data from *non-contiguous* positions soon proved to be a major obstacle. Techniques like permutation masks were applied to try and overcome these obstacles, but again with limited success; another modification introduced were the so-called *r-chunk detectors*. Finally, Stibor described the problem as a failure of negative selection algorithms to *generalize* properly beyond the training set – an essential feature of any machine learning algorithm. Taken together with the computational complexity issues, he concluded that Negative Selection is not appropriate at least for network intrusion detection [141, 136]. In his dissertation, Zhou Ji [81, ch. 5] responded to some of his criticism, arguing that many of these results can be tracked down to the incompatibility of the matching rules (which assume correlation between adjacent string positions) with the data representation (e.g., floating point numbers represented as binary strings), where such a correlation cannot be expected to exist.

4.4 Formalizing Negative Selection Algorithms

The field of *algorithmic learning theory* investigates algorithmic models of learning processes in nature and in machines. For example, the seminal paper by Gold [61] that laid the foundation for the *inductive inference* class of models aimed at a formal understanding of how children can learn languages from only positive examples. In this section and the upcoming ones, we settle the negative selection algorithm into the notational framework of algorithmic learning theory. This means that we will drop the immunological terminology with which negative selection algorithms are usually described in favor of using more technical terms. For example, what is called a *detector* in the artificial immune systems literature will here be called a *pattern*. This will make it easier to show what negative selection algorithms really are, and how they relate to established models from algorithmic learning theory.

4.4.1 Samples, Classification, and Anomaly Detection

Definition 4.1 (Samples)**.** *Let $\mathcal{U}$ be any set* (universe)*. A* labeled sample*, or simply* sample*, over $\mathcal{U}$ is a set $S = \{(x_1, \ell_1), (x_2, \ell_2), \ldots\} \subseteq \mathcal{U} \times \{+1, -1\}$ of positively or*

negatively labeled elements from $\mathcal{U}$, such that for no $x \in \mathcal{U}$, both $(x,+1)$ and $(x,-1)$ are contained in S. A sample containing at most k positively labeled elements is called a k-positive sample. *A* 0*-positive sample is also called a* negative sample.

Because we will be working with negative samples quite a lot, we will shorten notation in some cases by omitting the negative labels. For instance, we will sometimes write "let $S \subseteq \mathcal{U}$ be a negative sample" instead of the more formal "let $S \subseteq \mathcal{U} \times \{-1\}$ be a negative sample".

As illustrated in Figure 4.2, a negative selection algorithm is essentially a classification algorithm that partitions a universe $\mathcal{U}$ into two classes based on the information in an input sample S. However, negative selection algorithms use only samples from one of the underlying classes. Classifiers with this property are called *one-class classifiers* [127], and are also sometimes referred to as *anomaly detectors* or *novelty detectors* [136, 26]. Well-known examples include the one-class support vector machine [126] and kernel density estimation [12]. Negative selection differs from these methods mainly in that it is not a statistical, but a combinatorial technique.

Definition 4.2 (Classifiers, Anomaly Detectors)**.** *A* classifier *is an (in general probabilistic) algorithm $\mathfrak{C}(S,X)$ that, given a labeled sample S over $\mathcal{U}$ and a set $X \subseteq \mathcal{U}$, outputs for each $x \in X$ a labeling $(x,\ell), \ell \in \{+1,-1\}$. If $\mathfrak{C}$ outputs $(x,+1)$ for some $x \in X$, we also write "$\mathfrak{C}$ positively labels x"; if it outputs the label $(x,-1)$ for x, we write that "$\mathcal{C}$ negatively labels x". An* anomaly detector *is a classifier that accepts only negative samples as input.*

Anomaly detection problems appear very natural to humans. For example, those of us with a reasonable command of the English language would not find it very difficult to answer the following question: Which of the five text fragments

"j saed dau", "el answere", "nceits tha", "matuod gin", "re lies th"

were *not* extracted from a book written in English? Remarkably, we need *not* know which language the non-English strings come from to answer this question. This is analogous to what negative selection achieves for the immune system. We will use this "language anomaly detection problem" as a running example to illustrate some of the concepts introduced later.

4.4.2 Pattern Classes and Consistency

Definition 4.3 (Pattern Classes)**.** *A* pattern class $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ *consists of a set $\mathcal{U}$* (universe)*, a set $\mathcal{P}$* (patterns)*, and a polynomial time computable function*[5] $\mathcal{M}$:

[5] We are being somewhat imprecise here, because later on we will mostly be concerned with the universe Σ^*. In that case, by "polynomial time computable" we formally mean that a Turing machine can compute the desired output in time $O(n^c)$, where n is the amount of space required to store the input and c is a constant. However, e.g. for real-valued inputs, which can not always be stored in finite space, one would need to use a different model of computation.

$\mathcal{P} \times \mathcal{U} \to \{+1, -1, \perp\}$ (matching rule). *If $\mathcal{M}(\pi, x) = +1$, we also write "π matches x"; if $\mathcal{M}(\pi, x) = -1$, we write "π does not match x"; and if $\mathcal{M}(\pi, x) = \perp$, we write "$\pi$ and x are incompatible".*

The incompatibility symbol $\perp$ may appear a bit unusual at this point, but it will turn out very useful later on when we start reasoning about the computational complexity of pattern matching.

As mentioned before, patterns are referred to as *detectors* in the negative selection algorithms literature, reflecting the analogy to T cells in the immune system. In a learning theory context, we call a pattern instead a *concept*[6], a pattern class is called a *concept class*, and a matching rule is called a *membership function*. Perhaps the most natural example for a pattern class is the following, which was intensively investigated by Zhou Ji in his PhD thesis [81] where he called these patterns "V-detectors".

Example 4.4 (Hyperballs [81])**.** *Let $\mathbb{R}^* = \cup_{d \in \mathbb{N}} \mathbb{R}^d$ be the set of all real-valued vectors with finite dimension, and let $dim(x) = d$ denote the dimension of the vector $x \in \mathbb{R}^d$. Let $\|x - y\|$ denote the Euclidean distance between $x, y \in \mathbb{R}^d$. Let $\mathcal{P}_{\text{BALL}} = \{(c, r) : c \in \mathbb{R}^*, r \in \mathbb{R}\}$. We define the pattern class* BALL $= (\mathbb{R}^*, \mathcal{P}_{\text{BALL}}, \mathcal{M}_{\text{BALL}})$ *where*

$$\mathcal{M}_{\text{BALL}}((c, r), x) = \begin{cases} \perp & dim(c) \neq dim(x) \\ +1 & dim(c) = dim(x) \text{ and } \|c - x\| \leq r \\ -1 & otherwise\ . \end{cases}$$

In this example, the pattern (c, r) stands for the hyperball with center c and radius r. More generally, any family of hypervolumes defined by a finite set of real-valued parameters could be used as a pattern class in this fashion.

As has been illustrated in Figure 4.2, the essential step of a negative selection algorithm is to generate a set of patterns that all match to a certain input. This requirement can be connected to algorithmic learning theory via the following notion, which will play a central role in our analysis.

Definition 4.5 (Consistency)**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a pattern class and let $S \subseteq \mathcal{U} \times \{+1, -1\}$ be a labeled sample. A pattern $\pi \in \mathcal{P}$ is said to be* S-consistent *if for every $(x, \ell) \in S$, $\mathcal{M}(\pi, x) = \ell$. A pattern set $P \subseteq \mathcal{P}$ is called S-consistent if it only contains S-consistent patterns. We write $\mathcal{C}[S]$ for the set of* all S-consistent *patterns in $\mathcal{P}$.*

In particular, the empty pattern set $\emptyset$ is consistent with every negative sample, but it is not consistent with any sample that contains a positive example. Returning again to our hyperball example, given a set S of labeled points from $\mathbb{R}^d$ an S-consistent set of hyperballs would be one that contains all positively labeled points but none of the negatively labeled ones (see Figure 4.2).

[6] More precisely, a pattern is analogous to the *representation* of a concept.

A technical complication arises if the input set S contains two elements that do not admit any consistent patterns for purely syntactic reasons. For example, suppose that we would be given the input $S = \{((0,0),-1),((0,0,0),-1)\}$ for our hyperball patterns defined above. For the specific pattern classes that we analyze in this thesis, such syntactic incompatibilities are trivial to detect, and there are only finitely many syntactically compatible patterns for each universe element and vice versa. This property can be formalized using the following technical notion.

Definition 4.6 (Layered Pattern Classes)**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a pattern class and let $\lambda : \mathcal{U} \cup \mathcal{P} \to \mathbb{N}$ be polynomial time computable. Let $\mathcal{U}^{(L)} = \{x \in \mathcal{U} : \lambda(x) = L\}$ and $\mathcal{P}^{(L)} = \{\pi \in \mathcal{P} : \lambda(\pi) = L\}$. Suppose the following holds: (1) For all $L \in \mathbb{N}$, $|\mathcal{U}^{(L)}|$ and $|\mathcal{P}^{(L)}|$ are finite; (2) for all $\pi \in \mathcal{P}$, $x \in \mathcal{U}$, $\mathcal{M}(\pi, x) = \bot$ if and only if $\lambda(\pi) \neq \lambda(x)$. Then $\mathcal{C}$ is called* layered *and λ is called the* layering *of $\mathcal{C}$.*

4.4.3 String Patterns

The pattern class of a negative selection algorithm models the interaction between a T cell receptor and an epitope. This is most commonly done by representing both receptor and epitope as strings of a fixed length. Accordingly, we focus our analysis here on classes of patterns where the universe is the set of strings over some alphabet. Because unary alphabets are not interesting for our purposes, we assume that all alphabets are at least binary.

Definition 4.7 (Alphabets, Strings, Languages)**.** *An* alphabet Σ *is a finite set of* symbols *with $|\Sigma| \geq 2$. A* string $s \in \Sigma^*$ *is a sequence of symbols from Σ with finite length $|s|$. A* language $\mathcal{L}$ *is a set of strings. The string with $|s| = 0$ is called the* empty string*. For $i \in \{1, \ldots, |s|\}$, $s[i]$ is the i-th symbol in s. For two indices i and j with $j \geq i$, $s[i \ldots j]$ is the* substring *of s with length $j - i + 1$ starting at position i, and we write that $s[i \ldots j]$* occurs in *s at position i. For $j < i$, $s[i \ldots j]$ is empty. For $i \in \{1, \ldots, |s|\}$, $s[1 \ldots i]$ is a* prefix *of s and $s[i \ldots |s|]$ is a* suffix *of s. A prefix or suffix s' of s is* proper *if $0 < |s'| < |s|$.*

The following definition contains two concepts that are stated separately because they are less common than those defined above.

Definition 4.8 (Sub-Languages, Avoided Strings)**.** *For two indices i and j, we define $S[i \ldots j] = \{s[i \ldots j] : s \in S\}$. We say that S* avoids a string d at position i *if $d \notin S[i \ldots i + |d| - 1]$.*

In this thesis we will investigate four specific string pattern classes. Figure 4.3 shows for each of these classes an example pattern along with a matching and a non-matching string. In general, a string pattern class can be defined as follows.

Definition 4.9 (String Patterns)**.** *Fix a finite alphabet Σ. A* string pattern class *is a pattern class $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ with $\mathcal{U} = \Sigma^*$ with the layering $\lambda(x) = |x|$. Hence, $\mathcal{U}^{(L)} = \Sigma^L$.*

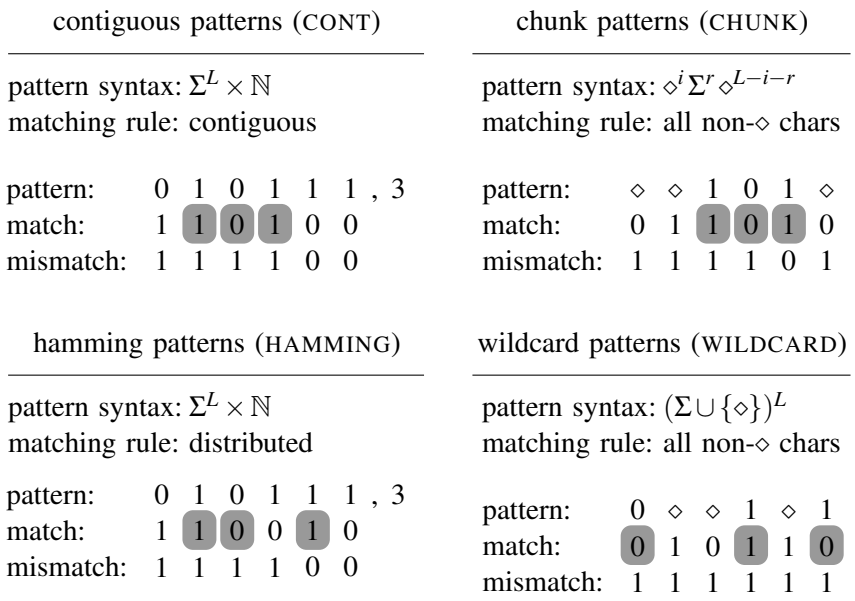


Figure 4.3: Illustration of the four string-based pattern classes that we consider in this chapter. See the text for precise definitions of the matching rules used. For each matching string, the letters that are relevant for the match are highlighted.

Our first pattern class is perhaps the most widely known one in the literature on negative selection algorithms [139]. It was designed by Percus, Percus, and Perelson [116] and is one of the oldest string-based models of T cell receptor–epitope interaction.

Definition 4.10 (Contiguous Patterns [116])**.** *The class of* contiguous patterns *is defined as* $\text{CONT} = (\Sigma^*, \Sigma^* \times \mathbb{N}, \mathcal{M}_{\text{CONT}})$ *where*

$$\mathcal{M}_{\text{CONT}}((\pi, r), x) = \begin{cases} \bot & |\pi| \neq |x| \\ +1 & |\pi| = |x| \text{ and there exists an } i \in \{1, \ldots, |x| - r + 1\} \\ & \text{where } \pi[i \ldots i + r - 1] = x[i \ldots i + r - 1] \\ -1 & \text{otherwise} \end{cases}$$

The r-contiguous pattern class thus defines a notion of imprecise matching that does not rely on wildcard symbols, such that all positions in the two strings are relevant for determining a match. Balthrop et al. [7] introduced a variant of the above pattern class that allows for partial matching.

Definition 4.11 (Chunk Patterns [7])**.** *Let* $\diamond$ *be a symbol that does not occur in* Σ*. The class of* chunk patterns *is defined as* $\text{CHUNK} = (\Sigma^*, \mathcal{P}_{\text{CHUNK}}, \mathcal{M}_{\text{WILDCARD}})$ *where*

$$\mathcal{P}_{\text{CHUNK}} = \{\diamond^i w \diamond^j : w \in \Sigma^*, i, j \in \mathbb{N}\}$$

and

$$\mathcal{M}_{\text{WILDCARD}}(\pi,x) = \begin{cases} \bot & \text{if } |\pi| \neq |x| \text{ ,} \\ +1 & \text{if } |\pi| = |x| \text{ and for all } j, \pi[j] = x[j] \text{ or } \pi[i] = \diamond \text{ ,} \\ -1 & \text{otherwise .} \end{cases}$$

Hence, the wildcard $\diamond$ in a chunk pattern stands for a single arbitrary symbol, and $\mathcal{P}_{\text{CHUNK}}$ is the set of those patterns in which all non-$\diamond$s form a contiguous substring. In the literature, the chunk pattern $\diamond^i w \diamond^j$ is usually written as the tuple $(w, i+1)$ [7, 42]. Moreover, an asterisk $*$ is normally used instead of $\diamond$, e.g. by Esponda [44]. We prefer the symbol $\diamond$ because $*$ usually denotes an arbitrary string of arbitrary length.

More recent models of theoretical immunology have often preferred the "simple" hamming distance to model TCR–peptide interaction [27, 28, 90, 91] over the seemingly more complex *r*-contiguous rule. However, it will turn out later that the *r*-contiguous rule is at least computationally more friendly than the Hamming distance.

Definition 4.12 (Hamming Patterns [63])**.** *The class of* Hamming patterns *is defined as* $\text{HAMMING} = (\Sigma^*, \Sigma^* \times \mathbb{N}, \mathcal{M}_{\text{HAMMING}})$ *using the matching function*

$$\mathcal{M}_{\text{HAMMING}}((\pi,r),x) = \begin{cases} \bot & |\pi| \neq |x| \\ +1 & |\pi| = |x| \text{ and } \delta_{\mathcal{H}}(\pi,x) \leq r \\ -1 & \text{otherwise} \end{cases}$$

where δ_H *denotes the Hamming distance* $\delta_H(\pi,x) = |\{i : \pi[i] \neq x[i]\}|$.

Note that the meaning of the parameter *r*, which is akin to a radius, is exactly the opposite between Hamming patterns and contiguous patterns, which is a somewhat unfortunate historical artifact.

Finally, we suggest to consider the following pattern class. It has to our knowledge not been used so far in negative selection algorithms, but formed the basis for so-called *learning classifier systems*, which are a hybrids of pattern-based classification schemes and genetic algorithms [74].

Definition 4.13 (Wildcard Patterns [74])**.** *The class of* wildcard patterns *is defined as* $\text{WILDCARD} = (\Sigma^*, \Sigma^* \cup \{\diamond\}, \mathcal{M}_{\text{WILDCARD}})$.

This pattern class is simply a general version of $\mathcal{P}_{\text{CHUNK}}$, in which wildcards may occur in arbitrary positions. For the binary alphabet $\Sigma = \{0,1\}$, the pattern class WILDCARD is equivalent to the class of Boolean *monomials*, i.e., formulas consisting only of conjunctively joined literals. For example, consider the pattern $\diamond 01 \diamond$: this would match all binary bitstrings described by the monomial $\overline{x_2} x_3$.

4.4.4 Restricted Patterns

We have so far treated the matching parameter (or "matching radius") of our patterns as a part of the pattern itself. Traditionally, the matching radius is considered as a constant that is extrinsically defined. Or in other words, what is usually considered are *restricted* versions of our pattern classes.

Definition 4.14 (Restricted Patterns)**.** *A* restricted pattern class $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M}, \rho)$ *consists of a pattern class* $(\mathcal{U}, \mathcal{P}, \mathcal{M})$ *and a polynomial time computable function* $\rho : \mathcal{P} \to \mathbb{N}$ (restriction)*. Let* $S \subseteq \mathcal{U} \times \{+1, -1\}$ *be a labeled sample, then we define the set of* S-consistent r-restricted *patterns for* $\mathcal{C}$ *as* r-$\mathcal{C}[S] = \{\pi \in \mathcal{C}[S] : \rho(\pi) = r\}$.

For a first intuitive example of restricted patterns, we turn again to the hyperballs in Euclidean space, where a natural restriction leads to *fixed-radius* hyperballs.

Example 4.15 (Hyperspheres with Radius r)**.** *For the pattern set* $\mathcal{P}_{\text{BALL}}$ *and the matching function* $\mathcal{M}_{\text{BALL}}$ *from Definition 4.4, we define the class of restricted hyperspheres as* r-BALL $= (\mathbb{R}^*, \mathcal{P}_{\text{BALL}}, \mathcal{M}_{\text{BALL}}, \rho_r)$ *where* $\rho_r(c, r) = r$.

Similarly natural restricted equivalents exist for the string patterns defined in the previous section. In all cases, we will use the notation r-$\mathcal{C}$ to refer to the restricted version of the pattern class $\mathcal{C}$.

Definition 4.16 (r-Contiguous Patterns [55])**.** *The class of* r-contiguous patterns *is defined as* r-CONT $= (\Sigma^*, \Sigma^* \times \mathbb{N}, \mathcal{M}_{\text{CONT}}, \rho_r)$ *where* $\rho_r(c, r) = r$.

For example, the contiguous pattern shown in Figure 4.3 is a 3-contiguous pattern from the class from the class 3-CONT.

Definition 4.17 (r-Chunk Patterns [7])**.** *The class of* r-chunk patterns *is defined as* r-CHUNK $= (\Sigma^*, \mathcal{P}_{\text{CHUNK}}, \mathcal{M}_{\text{WILDCARD}}, \rho_\diamond)$ *where* $\rho_\diamond(\pi) = |\{i : \pi[i] = \diamond\}|$.

For instance, $\diamond 01\diamond$ and $00\diamond\diamond$ are 2-chunk patterns over the alphabet $\Sigma = \{0, 1\}$ but $\diamond 0\diamond 1$ is neither a 2-chunk pattern nor a chunk pattern. The 2-chunk pattern $\diamond 01\diamond$ matches 0011 but it does not match 1100. The following two definitions use the same restrictions ρ_r and $\rho_\diamond$ as above.

Definition 4.18 (r-Hamming Patterns [63])**.** *The class of* r-Hamming patterns *is defined as* r-HAMMING $= (\Sigma^*, \Sigma^* \times \mathbb{N}, \mathcal{M}_{\text{HAMMING}}, \rho_r)$ *where* $\rho_r(\pi, r) = r$.

For example, the hamming pattern shown in Figure 4.3 is a 3-hamming pattern.

Definition 4.19 (r-Wildcard Patterns)**.** *The class of* r-wildcard patterns *is defined as* r-WILDCARD $= (\Sigma^*, \Sigma^* \cup \{\diamond\}, \mathcal{M}_{\text{WILDCARD}}, \rho_\diamond)$.

The wildcard pattern shown in Figure 4.3 contains three non-$\diamond$ letters, and thus belongs to the class 3-WILDCARD. If we use the binary alphabet $\Sigma = \{0, 1\}$, then the class r-WILDCARD is equivalent to the class of Boolean monomials with exactly r literals.

Algorithm $\mathfrak{N}(S,M)$.
Input: Negative sample $S \subseteq \mathcal{U} \times \{-1\}$, set $X \subseteq \mathcal{U}$.
Output: For each $x \in X$, either $(x,+1)$ or $(x,-1)$.

```
    training phase
1   P ← some subset of C[S]

    classification phase
2   for each x ∈ X do
3       if there exists a π ∈ P that matches x then
4           output (x, +1)
5       else
6           output (x, −1)
```

Figure 4.4: General outline of a negative selection algorithm as it is usually defined [55, 149]. Adapted from Liśkiewicz and Textor [99].

4.4.5 Negative Selection Algorithms

We are now prepared to give a formal definition of the negative selection algorithm that corresponds to the usual informal description [55, 149].

Definition 4.20 (Negative Selection Algorithm [55])**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a pattern class over some universe $\mathcal{U}$. A* negative selection algorithm *(NSA) using pattern class $\mathcal{C}$ is a classifier $\mathfrak{N}(S,X)$ that, on input of a negative sample $S \subseteq \mathcal{U} \times \{-1\}$ and a set $X \subseteq \mathcal{U}$, does the following:*

1. *$\mathfrak{N}$ generates some set $P \subseteq \mathcal{C}[S]$ of S-consistent patterns* (training phase).
2. *For each $x \in X$, $\mathfrak{N}$ outputs $(x,+1)$ if there exists a $\pi \in P$ that matches x, and $(x,-1)$ otherwise* (classification phase).

Figure 4.4 shows a pseudocode equivalent of this definition. Crucially, note that this general definition does *not* specify *how* the pattern set P is generated. Most existing implementations perform either some kind of brute force search or generate detectors randomly. We formalize these two basic approaches for implementing the training phase by defining two corresponding special cases of the generic negative selection algorithm. For simplicity, we define these special cases only for layered pattern types. We also assume that the input sample S contains only elements from a single layer $\mathcal{U}^{(L)}$ of the universe; otherwise, no consistent pattern would exist and the output would always be $(x,-1)$ for all $x \in X$. Moreover, one can assume that the set X contains only strings from the same layer as all other strings would always be labeled negatively. These "syntax checks" can be easily performed in polynomial time.

First we consider the most extreme case of a brute force training phase, in which the algorithm generates *all* consistent patterns.

Definition 4.21 (Exhaustive Negative Selection Algorithm [99]). *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a layered pattern class. An* exhaustive negative selection algorithm *using pattern class $\mathcal{C}$ is a negative selection algorithm $\mathfrak{N}(S, X)$ that, on input $S \subseteq \mathcal{U}^{(L)} \times \{-1\}, X \subseteq \mathcal{U}^{(L)}$, in its training phase generates the pattern set $P = \mathcal{C}[S]$.*

In general, exhaustive negative selection is impractical because $\mathcal{C}[S]$ is typically too large to be explicitly generated. Nevertheless, it is a useful scenario to consider, because we will later see that exhaustive negative selection can be *simulated* e.g. by generating a compact representation of $\mathcal{C}[S]$ and using that for classification instead of the pattern set itself. Moreover, it may be possible to achieve the classification outcome of an exhaustive negative selection algorithm with a smaller pattern set. Such a pattern set has been called *perfect pattern set* in the literature [82].

The second special case we consider is a Monte Carlo version of the negative selection algorithm, which generates patterns by rejection sampling. For sake of simplicity, we assume that only a single pattern is generated, which of course is not the case in practice; however, an algorithm that generates several patterns can be simulated by running our version several times. It is noted that we previously used a similar definition where the number of patterns to be generated was given as an additional parameter [99], which leads to some technical complications without providing substantially more insight than the simpler definition given below.

Definition 4.22 (Sampling Negative Selection Algorithm). *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a layered pattern class. A* sampling negative selection algorithm *using $\mathcal{C}$ is a negative selection algorithm $\mathfrak{N}(S, X)$ that, given a negative sample $S \subseteq \mathcal{U}^{(L)} \times \{-1\}$ and a set $X \subseteq \mathcal{U}^{(L)}$, performs the following:*

1. *$\mathfrak{N}$ iteratively samples a pattern π uniformly at random from $\mathcal{P}^{(L)}$ until π is S-consistent* (training phase).

2. *For each $x \in X$, $\mathfrak{N}$ outputs $(x, +1)$ if π matches x, and $(x, -1)$ otherwise* (classification phase).

Note in particular that the sampling negative selection algorithm does not terminate in the case that no S-consistent patterns exist.

While uniform sampling is possible for the pattern classes that we defined above, it can become highly nontrivial for *ambiguous* pattern languages, i.e., pattern languages where two different patterns can match exactly the same strings. For example, consider boolean formulas: Two different formulas can have exactly the same satisfying assignments. Uniform sampling from combinatorial structures is a currently very active and growing field, and many *approximate* uniform sampling algorithms have been obtained with the Markov Chain Monte Carlo method (MCMC). *Exact* uniform samplers are known however only for a few structures such as DNF formulas [80] and graph colorings [19].

4.5 Learning in Negative Selection Algorithms

The negative selection algorithm can be understood as a computational model of a learning process. *Algorithmic Learning Theory* is concerned with the development and analysis of such models, and has established a rich taxonomy to characterize algorithmic learners [3]. In this section we first describe some basic properties of negative selection algorithms using this taxonomy, and thereafter we address in some detail the aspect of *generalization*, i.e., the capability to infer an underlying partitioning of the universe from a sample set.

4.5.1 Negative Selection in the Context of Learning Theory

The first essential property we wish to state follows directly from Definition 4.20: Negative selection algorithms are *consistent learners* – an element whose label is known from the input data is never assigned a different label. The consistency requirement has been investigated in many contexts like inductive inference and efficient PAC learning [3]. It is known that the consistency requirement sometimes leads to an infeasible complexity of algorithmic learners; this issue will be investigated in more detail in Section 4.6.

One key difference between the negative selection algorithm and many models considered in algorithmic learning theory is the learning goal. For example, the learning goal of an inductive inference machine [61] or a PAC learner [152] is to infer a certain *concept* from the shown examples (up to a modest error in case of PAC learning). In our case, this would correspond to assuming that the "anomalous" region of the universe can be described by a single pattern from our pattern language, and that all elements of the sample S are strings that are *not* matched by this pattern. Kearns et al. [85] have shown that for such a case, exponentially many examples would be needed for any pattern class subsumed by the Boolean monomials. However, in our case the learning goal is not explicitly specified, as we make no assumption on how the underlying bipartitioning of the universe, according to which our examples are labeled, is generated. Indeed it is a defining assumption of anomaly detection problems that we have no a priori knowledge on the structure of the anomalies [26], and that the anomaly detection system should be free of such assumptions; otherwise, we would potentially provide attackers with an easy way to circumvent the system. In this regard, negative selection differs from models like PAC learning or inductive inference.

Version space learning [107] is another family of learning algorithms that are more similar to negative selection. Here the learner maintains a description of *all* hypotheses that are consistent with the labeled examples, which is called the version space. This can be mapped as follows to our terminology: A single S-consistent pattern would be one element of the version space (hypothesis), and the entire version space would correspond to the set of all S-consistent patterns. Hence, an exhaustive negative selection algorithm is indeed identical to a version space learner that is given only negative examples. This connection will be em-

phasized again later on when we characterize the computational complexity of exhaustive negative selection via the consistency problem – the same characterization has been performed for general version space learning algorithms by Hill et al. [67], even though we cannot directly use their results due to our restriction to negative examples. Finally, however, note that in version space learning the goal is usually defined as pruning the version space down to a single hypothesis, while in negative selection we use the version space simply as a description of a subset of the universe.

4.5.2 Coverage and Generalization

The classification outcome of a negative selection algorithm can be described via the following term.

Definition 4.23 (Coverage and Maximal Coverage)**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a pattern class. The* coverage *of a pattern set $P \subseteq \mathcal{P}$ is the set*

$$\mathcal{L}(P) = \{x \in \mathcal{U} : \text{ there exists a pattern } \pi \in P \text{ with } \mathcal{M}(\pi, x) = +1\}$$

of elements that are matched by the patterns in P. A pattern set P has maximal coverage *with respect to a sample S if it is S-consistent and there exists no other S-consistent pattern set P' such that $\mathcal{L}(P) \subsetneq \mathcal{L}(P')$. In particular, the pattern set $\mathcal{C}[S]$ has maximal coverage.*

The symbol $\mathcal{L}$ was chosen to denote coverage because for string-based pattern classes it is natural to think of the coverage of a pattern set P as the language of all strings described by the patterns in P. We refer the reader to Figure 4.5 for an illustration of a negative sample S along with all its S-consistent 3-chunk patterns and the induced coverage. The precise relationships between maximal coverage and the classification outcome of exhaustive and sampling negative selection algorithms are given by the following two observations.

Fact 4.24 (Maximal Coverage and Exhaustive Negative Selection)**.** *Let $\mathfrak{N}(S, X)$ be an exhaustive negative selection algorithm using pattern class $\mathcal{C}$. Then for every $x \in X$, $\mathfrak{N}(S, X)$ outputs $(x, +1)$ if and only if $x \in \mathcal{L}(\mathcal{C}[S])$.*

Fact 4.25 (Maximal Coverage and Sampling Negative Selection)**.** *Let $\mathfrak{N}(S, X)$ be a sampling negative selection algorithm using pattern class $\mathcal{C}$. Then for every $x \in X$, $Pr[\mathfrak{N}(S, X)$ outputs $(x, +1)] > 0$ if and only if $x \in \mathcal{L}(\mathcal{C}[S])$.*

In Figure 4.5, the maximal coverage $\mathcal{L}(\mathcal{C}[S])$ is highlighted in gray. The *generalization* performed by the classifier corresponds to the strings that are labeled negatively even though they do not occur in the sample S (non-bold strings with a white background). Somewhat misleadingly, such strings have been called *holes* in the negative selection literature. In immunology, *holes* are regions of the peptide shape space that cannot be detected by immune cells. Hence, pathogens can

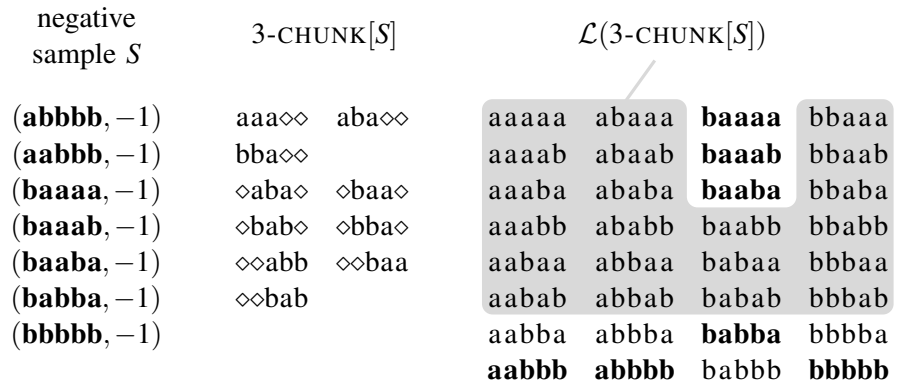


Figure 4.5: A negative sample $S \subseteq \{a,b\}^* \times \{-1\}$ (left) along with the set of its S-consistent 3-chunk patterns 3-CHUNK$[S]$ (middle) and the induced maximal coverage $\mathcal{L}$(3-CHUNK$[S]$) (right) consisting of the strings that are positively labeled by the patterns in 3-CHUNK$[S]$ (strings with shaded background). The maximal coverage induces a bipartitioning of $\{a,b\}^5$, in which the negatively labeled partition consists of the given samples (bold strings) and the *generalization region* (non-bold, non-shaded strings). Figure adapted from Elberfeld & Textor [43].

use these holes to evade recognition by the immune system (e.g. by evolving their proteins such that they lie within the holes). In a somewhat ad-hoc fashion, the existence of "holes" in a negative selection algorithm was – and is still today – considered as a "problem" [83]. In fact, as Stibor correctly pointed out [142], holes are a *necessary* property of the algorithm: Without holes, the algorithm would do nothing but memorize the training data and label everything else positively, which would hardly be interesting. In machine learning terms, the set of "holes" corresponds to the *generalization region* of the classifier, i.e., the subset of the universe that is inferred to be the source of the examples. The capability to perform such generalization is essential for a learning algorithm. In the following sections, we discuss how generalization is performed in negative selection algorithms.

4.5.3 Generalization in the Exhaustive, Unrestricted Case

It has been conjectured that maximal coverage is a desirable feature of a negative selection algorithm: "A perfect detector set, which recognizes all nonself strings that can be covered [...] is what we hope for" [82]. This would lead to think that exhaustive negative selection should be superior to sampling negative selection. However, it turns out that when using unrestricted pattern classes, maximal coverage can lead to a meaningless classification. This case can be characterized via the following property.

Definition 4.26 (1-Slicing)**.** *A pattern class $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ is called* 1-slicing *if for every $x \in \mathcal{U}$, $\mathcal{P}$ contains a pattern that matches only x.*

Proposition 4.27. *Let $\mathfrak{N}(S, X)$ be an exhaustive negative selection algorithm using a 1-slicing pattern class $\mathcal{C}$. Then for all $x \in X$, $\mathfrak{N}$ outputs $(x, +1)$ if and only if $(x, -1) \notin S$.*

Proof. The exhaustive negative selection algorithm uses the pattern set $\mathcal{C}[S]$. For each $x \in \mathcal{U} \setminus \{s : (s, -1) \in S\}$, there exists a pattern $\pi_x \in \mathcal{C}[S]$ that matches only x. Thus $x \in \mathcal{L}(\mathcal{C}[S])$ and the proposition follows from Fact 4.24. □

Thus, exhaustive negative selection algorithms (that use "perfect" detector sets) based on 1-slicing pattern classes do not generalize at all. In particular, we now show that this is the case for most pattern classes that involve Boolean formulas.

Definition 4.28. *A pattern class $\mathcal{D}_1 = (\mathcal{U}, \mathcal{P}_1, \mathcal{M}_1)$ is said to* subsume *another pattern class $\mathcal{D}_2 = (\mathcal{U}, \mathcal{P}_2, \mathcal{M}_2)$ over the same universe $\mathcal{U}$ if for every $\pi_1 \in \mathcal{P}_1$, there exists a $\pi_2 \in \mathcal{P}_2$ such that for all $x \in \mathcal{U}$, $\mathcal{M}_1(\pi_1, x) = \mathcal{M}_2(\pi_2, x)$.*

Corollary 4.29 (of Proposition 4.27)**.** *Any string-based pattern class subsumed by* WILDCARD, CHUNK, HAMMING *or* CONT *is* 1*-slicing.*

Proof. Every string $s \in \Sigma^*$ is a pattern from the class $\mathcal{P}_{\text{WILDCARD}}$ that matches only itself. Similar arguments hold for each of the other pattern classes. □

As noted in Section 4.4.3, if $\Sigma = \{0, 1\}$, then WILDCARD is equivalent to the class of Boolean *monomials*, i.e., formulas consisting only of conjunctively joined literals. Hence the corollary implies that exhaustive negative selection with a pattern class based on Boolean formulas will give useless results. However, we will see in the following that using Boolean patterns may be reasonable (but not necessarily computationally tractable) for *sampling* negative selection algorithms.

4.5.4 Generalization by Restriction

Because the combinatorial power of many unrestricted pattern classes leads to meaningless classification, a straightforward approach to obtain generalization is to restrict the set of valid patterns. This type of generalization in negative selection algorithms has been discussed in the literature [7, 142, 136]. The degree of generalization is controlled by the restriction parameter r (Figure 4.6): For example, consider the r-chunk patterns from the layer L. If we set $r = 0$, then the only pattern $\pi \in \mathcal{P}^{(L)}_{\text{chunk}}$ contains nothing but wildcards and matches *all* strings in Σ^L. Thus, the algorithm cannot generate an S-consistent pattern, and labels strings negatively. Obviously this is a too permissive generalization (top left panel in Figure 4.6). On the other hand, if we set $r = L$, the pattern (π, r) matches only the string π, and no generalization is performed as all strings except those from S are positively labeled (*overfitting* [127]; bottom right panel in Figure 4.6). The in-between values of r ranging from 1 to $n - 1$ provide varying degrees of generalization.

$|\mathcal{L}(0\text{-CHUNK}[S])| = 0$

aaaaa	abaaa	**baaaa**	bbaaa
aaaab	abaab	**baaab**	bbaab
aaaba	ababa	**baaba**	bbaba
aaabb	ababb	baabb	bbabb
aabaa	abbaa	babaa	bbbaa
aabab	abbab	babab	bbbab
aabba	abbba	**babba**	bbbba
aabbb	**abbbb**	babbb	**bbbbb**

$|\mathcal{L}(1\text{-CHUNK}[S])| = 0$

aaaaa	abaaa	**baaaa**	bbaaa
aaaab	abaab	**baaab**	bbaab
aaaba	ababa	**baaba**	bbaba
aaabb	ababb	baabb	bbabb
aabaa	abbaa	babaa	bbbaa
aabab	abbab	babab	bbbab
aabba	abbba	**babba**	bbbba
aabbb	**abbbb**	babbb	**bbbbb**

$|\mathcal{L}(2\text{-CHUNK}[S])| = 16$

aaaaa	abaaa	**baaaa**	bbaaa
aaaab	abaab	**baaab**	bbaab
aaaba	ababa	**baaba**	bbaba
aaabb	ababb	baabb	bbabb
aabaa	abbaa	babaa	bbbaa
aabab	abbab	babab	bbbab
aabba	abbba	**babba**	bbbba
aabbb	**abbbb**	babbb	**bbbbb**

$|\mathcal{L}(3\text{-CHUNK}[S])| = 21$

aaaaa	abaaa	**baaaa**	bbaaa
aaaab	abaab	**baaab**	bbaab
aaaba	ababa	**baaba**	bbaba
aaabb	ababb	baabb	bbabb
aabaa	abbaa	babaa	bbbaa
aabab	abbab	babab	bbbab
aabba	abbba	**babba**	bbbba
aabbb	**abbbb**	babbb	**bbbbb**

$|\mathcal{L}(4\text{-CHUNK}[S])| = 23$

aaaaa	abaaa	**baaaa**	bbaaa
aaaab	abaab	**baaab**	bbaab
aaaba	ababa	**baaba**	bbaba
aaabb	ababb	baabb	bbabb
aabaa	abbaa	babaa	bbbaa
aabab	abbab	babab	bbbab
aabba	abbba	**babba**	bbbba
aabbb	**abbbb**	babbb	**bbbbb**

$|\mathcal{L}(5\text{-CHUNK}[S])| = 25$

aaaaa	abaaa	**baaaa**	bbaaa
aaaab	abaab	**baaab**	bbaab
aaaba	ababa	**baaba**	bbaba
aaabb	ababb	baabb	bbabb
aabaa	abbaa	babaa	bbbaa
aabab	abbab	babab	bbbab
aabba	abbba	**babba**	bbbba
aabbb	**abbbb**	babbb	**bbbbb**

Figure 4.6: Illustration of generalization by restriction in a negative selection algorithm with r-chunk patterns. For the input sample $S \subseteq \{a,b\}^5$ from Figure 4.5 (bold strings), the classification outcome of an exhaustive negative selection algorithm with r-chunk patterns is shown with r ranging from 0 (top left) to 5 (bottom right).

To illustrate restriction-based generalization, let us come back to the analogy of detecting foreign languages mentioned in Section 4.4.1, and consider the following toy problem (which is not unlike the application we will pursue in Chapter 5). We trained an exhaustive negative selection algorithm with r-contiguous patterns on an a dataset $S \subseteq \{\mathrm{a},\ldots,\mathrm{z},\_\}^{10}$ obtained by extracting non-overlapping substrings of length 10 from the first two pages of Hermann Melville's classic novel "Moby Dick" [104]. This yielded 403 strings for the input sample S. To generate the input data X, we applied the same procedure to the first chapter of the Book of John in the English language (495 strings) and in the Hiligaynon[7] language (708 strings). Hiligaynon uses only the letters from the modern Latin alphabet (including "y") without accents, ligatures, or other special symbols – if this were not the case, then the anomaly detection problem would be trivial. Now, ideally our negative selection algorithm should negatively label all English ("normal") strings in X, whereas the Hiligaynon ("anomalous") strings should be positively labeled.

To evaluate the performance of the exhaustive negative selection algorithm on this problem, we generated ROC curves (receiver operating characteristic). A ROC curve visualizes the trade-off between sensitivity and specificity: For every value of r, one determines the false positive rate

$$\mathrm{FP}_r = \frac{\#\text{ normal instances labeled with } +1}{\#\text{ normal instances}}$$

and the true positive rate

$$\mathrm{TP}_r = \frac{\#\text{ anomalous instances labeled with } +1}{\#\text{ anomalous instances}}\,.$$

The ROC curve is then given by the points $(\mathrm{FP}_r, \mathrm{FP}_r)$ for all possible values of r. The larger the area under the ROC curve (AUC), the better the performance of the classification algorithm. The ROC curve of a "classifier" that tosses a coin to determine the label of each $x \in X$ is a diagonal line from the origin to the point $(1,1)$ with an AUC of 0.5. The ROC curve of a meaningful classifier should thus lie well above the diagonal and have an AUC higher than 0.5. A near-perfect classifier, which assigns almost all labels correctly for almost all parameters, has an AUC close to 1. The results of the ROC analysis applied to our language classification example are shown in Figure 4.7. The performance is hardly better than that of random coin tossing (AUC$=0.59$). For comparison: A one-class support vector machine [127] using a kernel based on the edit distance achieves an AUC of 0.73 on this dataset.

Our ROC analysis and Figure 4.6 jointly point to an obvious problem with restriction based generalization: The range of values for r that give acceptable classification results is typically very small, and possibly empty. This can be explained by noting that for the pattern classes we defined, the probability that an r-restricted pattern π matches a randomly generated string of the same length falls exponentially to 0 for increasing r, and rises exponentially to 1 for decreasing r. These limit

[7] Hiligaynon is an austronesian language spoken in the Philippines.

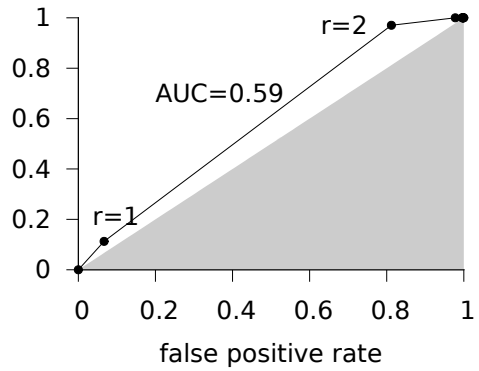


Figure 4.7: ROC curve for restriction-based generalization in an exhaustive negative selection algorithm with r-contiguous patterns. The algorithm was trained on a sample S of 403 lowercase strings (with all punctuation replaced by a single blank space) of length 10 extracted from Melville's "Moby Dick", and then given an input dataset X consisting of 495 strings extracted in the same manner from the English Bible plus 708 strings from the Bible in the Hiligaynon language. False and true positive rates were then determined for $r \in \{0, \ldots, 10\}$.

cases both give rise to meaningless classification, because either "almost none" or "almost all" strings are assigned the label $+1$. The effect of this can be witnessed in our language classification example: The classification is only (very slightly) nontrivial for $r = 1, 2$; however, $r = 1$ severely *overfits* (low detection rate) while $r = 2$ severely *underfits* (high detection rate, but unacceptable false positive rate).

4.5.5 Generalization by Pattern Sampling

Negative selection in the real immune system is most likely *not* exhaustive. The number of different T cell receptors that a mouse can generate has been estimated to 10^{15} [36], and the number of T cells that survive negative selection in a mouse is roughly 10^8 [15]. Even if one considers that roughly 95% of the T cells are deleted during negative selection, this still shows that only a very small fraction of the potential T cells are actually generated. We now show that this in fact need not be a disadvantage, as the stochastic nature of incomplete repertoire sampling leads to a previously unappreciated type of generalization in the negative selection algorithm. This type of generalization was first pointed out by the author in 2010 [99], and formalized via the so-called *pattern sampling distance*.

The idea behind the pattern sampling distance is to consider an algorithm that generates S-consistent patterns by rejection sampling (Definition 4.22), as it is done in the real immune system and also in most existing applications of the negative selection algorithm [149, 139]. The probability that an element x is classified as nonself with respect to the sample S is related to the difficulty of sampling an S-consistent pattern that matches x. If x is "far away" from S in the universe $\mathcal{U}$

(where the precise meaning of "far away" is induced by the matching rule), then the number of consistent patterns that can match x is larger than if x is "near" to S. These differences lead to a notion of *similarity* between S and x, which can take values between 0 and 1. Formally, we define the pattern sampling distance as the probability with which a sampling negative selection algorithm positively labels an input element.

Definition 4.30 (Pattern Sampling Distance)**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a layered pattern class. Let $x \in \mathcal{U}^{(L)}$, and let π be a pattern sampled from $\mathcal{P}^{(L)}$ uniformly at random. The* pattern sampling distance $\Delta_{\mathcal{C}}(S,x)$ *is defined by*

$$\Delta_{\mathcal{C}}(S,x) \quad = \quad \begin{cases} Pr[\,\mathcal{M}(\pi,x) = +1 \mid \pi \in \mathcal{C}[S]\,] & |\mathcal{C}[S]| > 0 \\ \perp & \textit{otherwise} \end{cases} .$$

Fact 4.31. *If $\Delta_{\mathcal{C}}(S,x) \neq \perp$, then*

$$\Delta_{\mathcal{C}}(S,x) = \frac{|\mathcal{C}[S \cup \{(x,+1)\}]|}{|\mathcal{C}[S]|} . \tag{4.1}$$

Fact 4.32. *If $\Delta_{\mathcal{C}}(S,x) \neq \perp$, then the sampling negative selection algorithm $\mathfrak{N}(S,\{x\})$ (Definition 4.22) using pattern class $\mathcal{C}$ terminates and $\Delta_{\mathcal{C}}(S,x)$ is the probability that it outputs $(x,+1)$. Otherwise, $\mathfrak{N}(S,\{x\})$ does not terminate.*

The sampling distance is thus intimately related to the detection outcome of stochastic negative selection algorithms: Consider a generalized version of our sampling negative selection algorithm that generates N patterns rather than just one. Then the algorithm will label an input element $x \in X$ positively if N is at least on the order of $1/\Delta_{\mathcal{C}}(S,x)$. Thus, by adjusting N, one can control the degree of generalization. This is orthogonal to the restriction-based generalization explained in the previous section.

We can not only predict the outcome of a sampling negative selection algorithm $\mathfrak{N}(S,X)$ by computing $\Delta_{\mathcal{C}}(S,x)$ for each $x \in X$, but also use these values directly for threshold-based classification. In fact, for classification purposes it is sufficient to compute only the cardinality of $\mathcal{C}[S \cup \{(x,+1)\}]$, as the denominator in equation 4.1 does not depend on x. For ROC analysis, we can define a threshold $\theta \in \mathbb{N}$ and assign the label $+1$ to those $x \in X$ where $|\mathcal{C}[S \cup (\{(x,+1)\}]| \geq \theta$; the ROC curve is then obtained by determining the resulting true positive rates TP_θ and false positive rates FP_θ for all possible values of θ. Note that TP_θ and FP_θ are monotone increasing functions of θ. Figure 4.8 illustrates the results of such a ROC analysis for our running example. Threshold-based classification via $\Delta_{\mathcal{C}}(S,x)$ or $|\mathcal{C}[S \cup (\{(x,+1)\}]|$ can be understood as a generalization of exhaustive negative selection, which would correspond to the special case $\theta = 0$. Therefore, one could say that sampling based generalization simply introduces a further classification parameter. This explains why sampling based generalization should normally yield a better classification result than merely restriction based generalization, as witnessed by comparing Figure 4.7 and Figure 4.8.

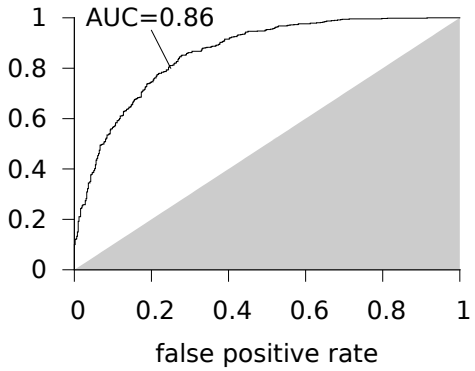


Figure 4.8: ROC curve for generalization via pattern sampling. For the sample S and input data X from Figure 4.7, the pattern sampling distance $\Delta_{r\text{-CONT}}(S,x)$ was computed for every $x \in X$, and the ROC curve was determined by ranking the $x \in X$ accordingly.

For sampling based generalization to be meaningful, different patterns from the same layer should match at least approximately the same number of strings; in other words, $\Delta_{\mathcal{C}}(\emptyset,x)$ should be roughly the same for all $x \in \mathcal{U}^{(L)}$. Otherwise, the algorithm will suffer from a built-in classification bias. Formally, one could require the pattern class to fulfill the following property at least approximately.

Definition 4.33 (Uniformly Layered Pattern Classes)**.** *Let $\mathcal{C} = (\mathcal{U},\mathcal{P},\mathcal{M})$ be a layered pattern class. Then $\mathcal{C}$ is called* uniformly layered *if for every layer $L \in \mathbb{N}$, the following holds: For all $\pi_1,\pi_2 \in \mathcal{P}^{(L)}$, $|\mathcal{L}(\pi_1)| = |\mathcal{L}(\pi_2)|$.*

In summary, we can state that generalization via the pattern sampling distance is a more fine-grained version of restriction-based generalization, which could alleviate the built-in tendency [137, 138, 139] of restriction-based generalization to over- or underfit the input sample.

4.5.6 Beyond Consistency: The Matching Profile

The fact that negative selection algorithms are consistent learners should seem bad news from a machine learning perspective, because consistent learners have inherent difficulties in dealing with noisy data – unfortunately, most real-world data is noisy. Before turning our attention to computational complexity questions, we wish to briefly outline a technique that can be used to build non-consistent negative selection algorithms. For pattern classes with a built-in matching radius, like r-contiguous or r-hamming patterns, this technique can be obtained as follows.

Definition 4.34 (Matching Profile)**.** *Let $\mathcal{C} = (\mathcal{U},\mathcal{P},\mathcal{M},\rho)$ be a layered, restricted pattern class where $\mathcal{P} = \Sigma^* \times \mathbb{N}$. Let $S \subseteq \Sigma^L \times \{-\}, x \in \Sigma^L$. Then the* matching profile $\Pi_{r\text{-}\mathcal{C}}(S,x)$ *is defined by*

$$\Pi_{r\text{-}\mathcal{C}}(S,x) = \left(\Pi^{(1)}_{r\text{-}\mathcal{C}}(S,x),\ldots,\Pi^{(L)}_{r\text{-}\mathcal{C}}(S,x)\right)$$

where

$$\Pi^{(r')}_{r\text{-}\mathcal{C}}(S,x)) = |\{(\pi,r) \in r\text{-}\mathcal{C}[S] : \mathcal{M}((\pi,r'),x) = +1\}| \ .$$

We can envisage the components of the matching profile as asymmetric versions of the pattern sampling distance. In particular, $\Pi^{(r)}_{r\text{-}\mathcal{C}}(S,x)$ is the denominator of $\Delta_{r\text{-}\mathcal{C}}(S,x)$. More generally, $\Pi^{(r')}_{r\text{-}\mathcal{C}}(S,x)$ is the number of patterns π with the property that (π,r) does not match any $s \in S$, but (π,r') matches x. Hence, the matching radius used for training and classification are no longer the same.

The matching profile has numerous applications. For instance, one may wish to rank the elements in X by their pattern sampling distance to S as we have done above (this will also be our goal in Chapter 5). Then it often is desirable to avoid ties in the ranking, e.g., if one wants to perform non-parametric statistical tests. Given the matching profiles of two elements $x,y \in X$ with $\Delta_{r\text{-}\mathcal{C}}(S,x) = \Delta_{r\text{-}\mathcal{C}}(S,y)$, which implies that $\Pi^{(r)}_{r\text{-}\mathcal{C}}(S,x) = \Pi^{(r)}_{r\text{-}\mathcal{C}}(S,y)$, the tie can be broken by comparing $\Pi^{(r-1)}_{r\text{-}\mathcal{C}}(S,x)$ to $\Pi^{(r-1)}_{r\text{-}\mathcal{C}}(S,y)$, then $\Pi^{(r-2)}_{r\text{-}\mathcal{C}}(S,x)$ to $\Pi^{(r-2)}_{r\text{-}\mathcal{C}}(S,y)$, and so forth until a difference is found.

Moreover, one can directly base the classification on some element $\Pi^{(r')}_{r\text{-}\mathcal{C}}(S,x)$ of the matching profile rather than the pattern sampling distance. For $r' \leq r$, this yields a non-consistent classifier, which may be more robust towards noisy data. Let us illustrate this fact using again our language classification example, where our training set S consists of 406 strings of length 10 from "Moby Dick", and the normal instances in our test set X are 495 strings from the English Bible. Now, suppose an adversary would sneak some anomalous strings into our training set S. No matter how small the number of such strings would be compared to the size of S, a classifier based on the pattern sampling distance would always consider them as normal.

This is not the case if we use the matching profile for classification. As an example, suppose we now use only the first 7 strings of the Hiligaynon Bible as anomalous instances, and perform a classification based on, say, $\Pi^{(2)}_{7\text{-CONT}}$ instead of the pattern sampling distance. This gives an AUC value of 0.8 ($p = 0.006$). Now, we inject the 7 Hiligaynon strings into S. While our pattern sampling distance based algorithm would now classify all of them as normal, the matching profile is not so easily fooled, and seems to detect that these strings are really outliers (AUC$= 0.74, p = 0.03$). This effect appears to warrant further investigation, which, however, will not be accomplished within the scope of this thesis. Thus, let us leave it at this basic example and turn our attention to the computational complexity of negative selection.

4.6 The Computational Complexity of Negative Selection

One can view a negative selection algorithm as a specific way of computing a certain function. Even though this specific implementation might be very inefficient, other algorithms might exist that compute the desired function more quickly. In

the standard *black box* approach to computational complexity, one thus seeks to prove upper and lower bounds for the complexity of *any* algorithm that computes the desired function. In this section, we investigate upper and lower complexity bounds for the functions computed by negative selection algorithms.

It was long thought that the negative selection algorithms suffer inherently from prohibitive computational complexity [136, 139]; however, this turns out to be true only for naive implementations of the scheme that Forrest et al. [55] originally proposed. In fact, the classification outcome of negative selection algorithms is often computable in polynomial time; whether this is possible or not depends on the employed pattern matching rule. We will give examples for some matching rules that allow efficient implementations, but also provide hardness results for some cases where this is unlikely to be possible.

Our analysis will again be concerned with the two special cases of negative selection algorithms that we defined in Section 4.4.5 – exhaustive negative selection and sampling negative selection. We will show that the computational complexity of the function computed by an exhaustive negative selection algorithms is tightly bounded by the complexity of an associated formal decision problem, the *consistency problem*. For sampling negative selection, we can provide upper bounds for the computational complexity of the probabilistic function computed by a sampling negative selection algorithm through the complexity of computing the pattern sampling distance, which leads to (slightly generalized) *counting problems*.

4.6.1 Computational Complexity Tools

To provide evidence for the computational hardness of decision problems, we use the standard notion of *NP-completeness*. Informally, a decision problem is NP-complete if it is roughly as hard as finding a satisfying assignment for a formula in propositional logic – it is conjectured, although not proved, that no efficient algorithm (i.e., with polynomial runtime) exists for this problem. For formal definitions of Turing machines, polynomial time computability, NP-completeness and other standard notions of computational complexity, see the textbook by Garey and Johnson [59].

Furthermore, we will discuss the computational complexity of rational-valued functions $\varphi : \Sigma^* \to \mathbb{Q}$. For these functions we will either show that they can be computed in polynomial time, or are hard for the complexity class #P introduced by Valiant [151] in the sense that we define as follows.

Definition 4.35 (The Complexity Class #P [151])**.** *Let M be a nondeterministic Turing machine with input alphabet Σ. Let $\#acc_M$ denote the function that maps every string $x \in \Sigma^*$ to the number of accepting computation paths of M on input x. Then*

$$\#P = \{\#acc_M \mid M \text{ is a polynomial-time nondeterministic Turing machine}\} .$$

To show that rational-valued functions are hard for #P, we use the following reduction. We assume that rational numbers are encoded as pairs of integers.

Definition 4.36 (#P-Hardness of Rational-Valued Functions)**.** *For two functions $f, g : \Sigma^* \to \mathbb{Q}$, we write that f is* polynomial-time one-Turing reducible *to g if there is a pair of polynomial-time computable functions $R_1 : \Sigma^* \to \Sigma^*$ and $R_2 : \Sigma^* \times \mathbb{Q} \to \mathbb{Q}$, such that for all x we have $f(x) = R_2(x, g(R_1(x)))$. We say that a function $\varphi : \Sigma^* \to \mathbb{Q}$ is* #P-hard *if every function in #P is polynomial-time one-Turing reducible to φ.*

#P-hardness of a function φ provides strong evidence about its intractability: For every problem Π from the *polynomial hierarchy* [114], there exists a deterministic Turing machine M solving Π in polynomial time using a single query to a φ-oracle. In this sense, #P is a "harder" complexity class than NP, which is the first level of the polynomial hierarchy.

4.6.2 Complexity of Exhaustive Negative Selection

The computational complexity of the function computed by an exhaustive negative selection algorithm (Definition 4.21) can be precisely characterized by a related formal decision problem: the well-known *consistency problem* from learning theory.

Definition 4.37 (Consistency Problem)**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a pattern class. Then the* consistency problem for $\mathcal{C}$ *is defined as follows: Given a sample $S \subseteq \mathcal{U} \times \{+1, -1\}$ as input, reject if $\mathcal{C}[S]$ is empty, and accept otherwise. Accordingly, the* restricted consistency problem *for a restricted pattern class $\mathcal{C}$ is defined as follows: Given a sample $S \subseteq \mathcal{U} \times \{+1, -1\}$ and a number $r \in \mathbb{N}$ as input, accept if and only if r-$\mathcal{C}[S]$ is not empty. We say that a consistency problem is k-*positive *if it is only defined for k-positive samples.*

Proposition 4.38 (Exhaustive Negative Selection and the Consistency Problem)**.** *Let $\mathfrak{N}(S, X)$ be an exhaustive negative selection algorithm using pattern class $\mathcal{C} = \{\mathcal{U}, \mathcal{P}, \mathcal{M})$. The function $\varphi_{\mathfrak{N}} : (S, X) \mapsto \mathfrak{N}(S, X)$ where $S \subseteq \mathcal{U} \times \{+1, -1\}$ and $X \subseteq \mathcal{U}$ is polynomial time computable if and only if the 1-positive consistency problem for $\mathcal{C}$ is polynomial time decidable.*

Proof. According to Fact 4.24, $\mathfrak{N}(S, X)$ positively labels exactly the elements of $\mathcal{L}(\mathcal{C}[S])$. Hence, every $x \in \mathcal{X}$ is negatively labeled if and only if there exists no S-consistent $\pi \in \mathcal{P}$ that matches x. This means that to determine the label assigned to x, we only need to check whether $\mathcal{C}[S \cup \{(x, +1)\}] > 0$. Thus we can simulate $\mathfrak{N}(S, X)$ by solving $|X|$ instances of the 1-positive consistency problem for $\mathcal{C}$.

For the other direction, suppose we can compute the output of $\mathfrak{N}(S, X)$ in polynomial time. Then we can answer whether $\mathcal{C}[S \cup \{(x, +1)\}] > 0$ in polynomial time by inspecting the output of $\mathfrak{N}(S, \{x\})$. □

The consistency problems plays a key role in consistent learning algorithms in general, and the first instance of an NP-complete consistency problem was given in

1988 by Pitt et al. [119]: Given a sample of labeled bitvectors, is it possible to construct a 3-term DNF formula that is consistent with the sample? Hirsh et al. [67] showed that solving the consistency problem is essential for many operations in version space learning – in this sense, Proposition 4.38 can be understood as a specialization of their result to negative selection algorithms. However, the hardness results obtained by Hirsh et al. for various consistency problems do not carry over to negative selection – a consistency problem may become much easier if we allow only one positive sample. This is the case for the 1-slicing pattern classes defined in the previous section (Definition 4.26), for which we saw that they are inappropriate for exhaustive negative selection algorithms because the resulting algorithm does not generalize. Thus it is not surprising to obtain the following proposition, which we state here only to emphasize that consistency problems often become substantially easier in the 1-positive case.

Proposition 4.39. *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a 1-slicing pattern class layered by λ. Then the 1-positive consistency problem for $\mathcal{C}$ is polynomial time decidable.*

Proof. If there exists a pair of samples $(x_1, \ell_1), (x_2, \ell_2) \in S$ with $\lambda(x_1) \neq \lambda(x_2)$, then there can be no pattern consistent with both and we can reject the input upfront. This can be tested in polynomial time. Otherwise, there exists a pattern that matches the unique positive example $(x, +1)$ in S and does not match any of the negative examples in S, and we can accept the input. □

Remember that most pattern classes involving Boolean formulas are 1-slicing, including the 3-term DNF formulas for which the general consistency problem is NP-complete [119]. Moreover, all pattern classes we defined in Section 4.4.3 are 1-slicing. Therefore, it is only of interest to study the consistency problems for the *restricted* versions of these pattern classes: For some of these, the consistency problem is efficiently solvable, while for others we can prove it NP-hard. We start out with the tractable cases. To avoid unnecessary technicalities, we assume from now on that for layered pattern classes, input samples are "syntactically valid", i.e. they contain only strings from a single layer $\mathcal{U}^{(L)}$; recall that this was also assumed in the definition of the exhaustive negative selection algorithm (Definition 4.21).

Proposition 4.40 (Tractability of r-Chunk Patterns)**.** *The 1-positive restricted consistency problem for r-*CHUNK *is polynomial time decidable.*

Proof. Let $(s, +1)$ be the single positively labeled example in the input $S \subseteq \Sigma^L \times \{+1, -1\}$. We simply need to verify that for all $i \in \{1, \ldots, L - r + 1\}$, it holds that $s \notin S[i \ldots i + r - 1]$. If that is the case, we accept the input, and reject it otherwise. □

The proof of the following theorem is somewhat intricate and is thus postponed to Section 4.7, where we will have more sophisticated string processing tools available. The theorem is a direct consequence of Lemma 4.62 and Lemma 4.65. We include the 0-positive restricted consistency problem in this case because it is equiv-

alent to the "detector generability" problem considered by Stibor [143], which was conjectured to be NP-complete [149].

Theorem 4.41 (Tractability of r-Contiguous Patterns)**.** *The* 0- *and* 1*-positive restricted consistency problems for* r-CONT *are polynomial time decidable.*

For the remaining two of our string-based pattern classes, we will now show that the 1-positive restricted consistency problems are NP-complete. Therefore, the outcome of an exhaustive negative selection algorithm is unlikely to be polynomial time computable[8].

Theorem 4.42 (Intractability of r-Wildcard Patterns)**.** *The* 1*-positive restricted consistency problem for* r-WILDCARD *is NP-complete.*

Proof. The proof is a straightforward reduction from the set cover problem. The set cover problem is defined as follows: Given L sets $X_1, X_2, \ldots, X_L \subseteq \{1, \ldots, n\}$ and a number k, can we choose k of the given sets such that their union contains all numbers from 1 to n? We can reduce an instance of set cover to an instance of the 1-positive restricted consistency problem for r-WILDCARD as follows. Set $\Sigma = \{a, b\}$. Create negative examples $(s_i, -1), s_i \in \Sigma^L, 1 \leq i \leq n$, such that $s_i[j] = b$ if $i \in X_j$ and $s_i[j] = a$, otherwise. Then create a positive example $s_0 = (a^L, +1)$. Now let π be any S-consistent pattern with k non-$\diamond$ symbols. All of these non-$\diamond$ symbols must be the letter a. Let $\{i_1, \ldots, i_k\}$ be the indices of these symbols, then $X_{i_1}, \ldots, X_{i_k}$ is a solution to the set cover instance. Conversely, if $X_{i_1}, \ldots, X_{i_k}$ is a solution to the set cover instance, then the pattern with the letter a at positions $\{i_1, \ldots, i_k\}$ and the symbol $\diamond$ everywhere else is S-consistent. □

Theorem 4.43 (Intractability of r-Hamming Patterns)**.** *The* 1*-positive restricted consistency problem for* r-HAMMING *is NP-complete.*

Proof. The 0-positive consistency problem for r-distributed patterns can be equivalently stated as follows: Given a set S of strings of length L, find a string π such that $\delta_H(\pi, s) > r$ for all $s \in S$, where δ_H denotes the Hamming distance. This is the well-known *farthest string problem* [93], which is NP-complete. For the 1-positive consistency problem, we have the additional constraint $\delta_H(d, x) \leq r$, where x is the unique positive example. It is easy to show that despite the additional constraint the problem remains NP-hard. □

4.6.3 Complexity of Sampling Negative Selection

The counterpart of the consistency problem for a *sampling* negative selection algorithm is the pattern sampling distance (Definition 4.30) Δ_C. Via computing Δ_C, a sampling negative selection algorithm can be simulated in polynomial time.

[8] As a side note, we conjecture that the 0-positive restricted consistency problem for this class admits a pseudo polynomial time algorithm. In this case it is unlikely to be NP-hard.

Proposition 4.44 (Efficient Simulation of Sampling NSA)**.** *Let $\mathcal{C} = (\mathcal{U}, \mathcal{P}, \mathcal{M})$ be a layered pattern class and let $\mathfrak{N}$ be a sampling negative selection algorithm using $\mathcal{C}$. If $\Delta_{\mathcal{C}}$ is polynomial time computable, then there exists a probabilistic algorithm $\mathfrak{A}(S,X)$ with expected polynomial runtime such that for each $x \in X$ we have*

$$Pr[\mathfrak{N}(S,X) \text{ outputs } (x,+1)] = Pr[\mathfrak{A}(S,X) \text{ outputs } (x,+1)] \ .$$

Proof. To simulate $\mathfrak{N}(S,X)$, we simply calculate $p = \Delta_{\mathcal{C}}(S,x)$ for each $x \in X$ and output $(x,-1)$ if $p = \perp$; otherwise we output $(x,+1)$ with probability p or $(x,-1)$ with probability $1-p$. To output things with these exact probabilities using a Turing machine that is only allowed to throw fair coins, one can use rejection sampling, which yields the desired output probability distribution in expected polynomial time [80]. □

Computing $\Delta_{\mathcal{C}}(S,x)$ however provides much more information than any single run of the sampling NSA. Consequently, the computational complexity of calculating the pattern sampling distance provides only an upper bound for the complexity of sampling NSA, and the following theorem shows that this upper bound is not tight.

Theorem 4.45 (Efficient Sampling NSA Without Counting Patterns)**.** *There exist pattern classes $\mathcal{C}$ for which (1) computing $\Delta_{\mathcal{C}}$ is #P-hard, and (2) there exists a sampling NSA using $\mathcal{C}$ with expected polynomial runtime.*

We first prove the following technical lemma.

Lemma 4.46 (Embedding an Additional Object into a Uniform Sampler)**.** *Let X be a finite set of unknown cardinality $|X| > 1$, and suppose that there exists an algorithm $\mathcal{A}$ with expected polynomial runtime that generates an element from X uniformly at random. Let $x^* \notin X$. Then there exists an algorithm $\mathcal{A}^*$ that generates an element from $X \cup \{x^*\}$ uniformly at random in expected polynomial time .*

Proof. Let n denote the unknown cardinality of X. Our procedure $\mathcal{A}^*$ works as follows: (1) $\mathcal{A}^*$ samples an element a uniformly at random from X. (2) $\mathcal{A}^*$ repeatedly samples a tuple (x,y) uniformly at random from X^2 until $(x,y) \neq (a,a)$. (3) If $x = a$, then $\mathcal{A}^*$ outputs x^*. Otherwise, $\mathcal{A}^*$ outputs a. Now the probability that $\mathcal{A}^*$ outputs x^* is

$$\frac{n-1}{n^2-1} = \frac{1}{n+1} \ ,$$

and thus the output probability distribution is uniform over $X \cup \{x^*\}$. The lemma now follows by noting that because $|X| \geq 2$, step (2) above terminates after a constant expected number of iterations. □

Now we are prepared to prove Theorem 4.45.

Proof of Theorem 4.45. We need to define a pattern class $\mathcal{C}$ such that (1) computing $\Delta_{\mathcal{C}}$ is #P-hard, but at the same time (2) it is feasible to sample exactly uniformly from $\mathcal{C}[S]$ for negative samples S. To this end, we will encode a combinatorial problem into our pattern class that has a #P-hard counting version even though one can sample uniformly from the solution space. The first problem that was shown to have this property is DNF satisfiability [80]. However, it does not appear feasible to express an arbitrary DNF formula in terms of a 1-positive consistency problem, which has the form of a *conjunction* of constraints. Instead, we use here the *graph coloring* problem.

Let us denote the set $\{1,\ldots,n\} \subseteq \mathbb{N}$ by $[1,n$. We recall that a k-coloring of a graph $G = (V,E)$ is a mapping $C : V \to [1,k]$, and it is called *valid* if $C(v) \neq C(w)$ for all $\{v,w\} \in E$. Counting the k-colorings of a graph with maximal degree κ is #P-hard for all constants $k, \kappa \geq 3$ [19]. Still, we can sample exactly from the valid k-colorings as long as $k > \kappa(\kappa+2)$ [77]. Below, we assume that $\kappa \leq 3$ and $k \geq 15$. Note that a graph of maximum degree 3 is always 15-colorable, such that the decision version of the graph coloring problem is trivial for these constants.

To make the proof more palatable, we proceed in two steps. First we show that there exists a pattern class $\mathcal{C}$ for which determining $|\mathcal{C}[S]|$ is #P-hard, even though we can sample exactly uniformly from $\mathcal{C}[S]$, where S is a 0-positive sample. This is not yet exactly what we need because $|\mathcal{C}[S]|$ is only the denominator of the pattern sampling distance $\Delta_{\mathcal{C}}$ (see Equation 4.1); it may in principle be infeasible to calculate the denominator, but feasible to calculate the entire fraction. This can however be dealt with rather easily, as will be shown in our second step.

Step 1. We define a layered pattern class $\textsc{gcol} = (\mathcal{U}_{\textsc{gcol}}, \mathcal{P}_{\textsc{gcol}}, \mathcal{M}_{\textsc{gcol}})$ as follows. Our universe is a set of graphs of the form

$$\mathcal{U}^{(L)}_{\textsc{gcol}} = \{(V,E,\rho) : V = [1,L],\ \rho \in V,\ E \subseteq \{\{\rho,v\} : v \in V \setminus \{\rho\}\},\ |E| \leq 3\}\ ,$$

i.e., rooted star graphs with at most 3 edges. Figure 4.9 shows three examples s_1, s_2, and s_3 (with roots ρ in bold).

Our pattern language and matching function will now be defined in such a way that a negative sample $S \subseteq \mathcal{U}_{\textsc{gcol}} \times \{-1\}$ encodes an *induced graph* $G(S)$; hence, counting $G(S)$-consistent patterns will be equivalent to counting the valid k-colorings of $G(S)$. To this end,

$$\mathcal{P}^{(L)}_{\textsc{gcol}} = \{(V,E,C) : V = [1,L],\ E \subseteq \{e \subseteq V : |e| = 2\},\ C : V \to [1,k]\}$$

is simply the set of all arbitrarily k-colored graphs (the coloring need not be valid). The matching function is defined as follows:

$$\mathcal{M}_{\textsc{gcol}}\big((V,E,C),(V',E',\rho)\big) = \begin{cases} \bot & V' \neq V \\ +1 & (1)\ E' \not\subseteq E \\ & \text{or (2) } E \setminus E' \text{ contains a } \rho\text{-adjacent edge} \\ & \text{or (3) } C \text{ is no valid coloring for } (V',E') \\ -1 & \text{otherwise} \end{cases}$$

Consider now a negative sample S, and suppose there does not exist an S-consistent pattern $\pi = (V,E,C)$. This may occur only if one of the following holds: (1) S contains two elements from different layers; (2) there exist two samples $(V',E'_1,\rho_1),(V',E'_2,\rho_2) \in S$ and an edge $\{\rho_1,v\} \in E'_1$ such that $\{\rho_1,v\} \notin E'_2$ and $\rho_2 \in \{\rho_1,v\}$. In other words, once an edge $\{u,v\}$ appears in one sample graph (in which either u or v is the root), it has to appear in *every* sample graph in which either u or v is the root for there to be an S-consistent pattern. Both cases, and thus the 0-positive consistency problem for GCOL, can be verified in polynomial time.

Consider now a negative sample $S = \{((V',E'_1,\rho_1),-1),\ldots,((V',E'_n,\rho_n),-1))\}$ for which at least one consistent graph pattern $\pi = (V,E,C)$ exists. We define the *induced graph* $G(S)$ by $G(S) := (V, \cup_i E_i)$. Then $G(S)$ has maximum degree $\kappa \leq 3$. The definition of $\mathcal{M}_{\text{GCOL}}$ ensures that (1) $G(S)$ has maximum degree $\kappa \leq 3$; (2) every S-consistent graph $\pi \in \mathcal{P}_{\text{GCOL}}$ contains $G(S)$ as a subgraph, and (3) C is a valid coloring of $G(S)$. Edges whose endpoints do not appear as roots in S may or may not be present in π (dashed edges in the consistent graph π in Figure 4.9), and nodes that are not adjacent to any of the sample edges (like the top left and bottom left nodes in the consistent graph in Figure 4.9) may be assigned an arbitrary color.

Let $\#\chi(G(S))$ the number of valid k-colorings of $G(S) = (V,E(S))$. Denote the roots of the star graphs in S by $\rho(S)$. Then, assuming $|\text{GCOL}[S]| > 0$, we have

$$|\text{GCOL}[S]| = \#\chi(G(S)) \cdot 2^{\varepsilon} \ ,$$

where

$$\varepsilon = \frac{\delta^2 - \delta}{2} \ , \ \delta = |V| - |\rho(S)| \ ,$$

is the number of edges missing in $G(S)$ that are not adjacent to some root from S, which may or may not be present in S-consistent graphs.

Now suppose we had an algorithm for computing $|\text{GCOL}(S)|$. Then we could determine the number $\#\chi(G)$ of valid k-colorings for an arbitrary graph G of maximal degree κ as follows: Decompose $G = (V,E)$ into star graphs by creating for each node $v \in V$ a sample graph consisting of the root v and its adjacent edges in G. Label all these samples negatively, which gives a negative sample S. Then $G(S) = G$, and

$$\#\chi(G) = \frac{|\text{GCOL}[S]|}{2^{\varepsilon}}$$

gives the number of valid k-colorings of G (recall that for our values of k,κ, it is guaranteed that $\#\chi(G) > 0$). Because ε can be computed in polynomial time from G, computing $|\text{GCOL}[S]|$ must thus be #P-hard for $k,\kappa \geq 3$.

Conversely, given an arbitrary negative sample S over $\mathcal{U}$, we can sample from $\text{GCOL}[S]$ as follows. First check whether $|\text{GCOL}[S]| = 0$ as discussed above, in which case the output is not defined. Otherwise, compute the induced graph $G(S) = (V,E)$, and sample a valid coloring of $G(S)$ at uniform using Huber's algorithm [77], which runs in expected polynomial time for $k > \kappa(\kappa+2)$. Next, consider every missing edge $\{u,v\} \subseteq V$ where none of the endpoints appear as roots in S, and

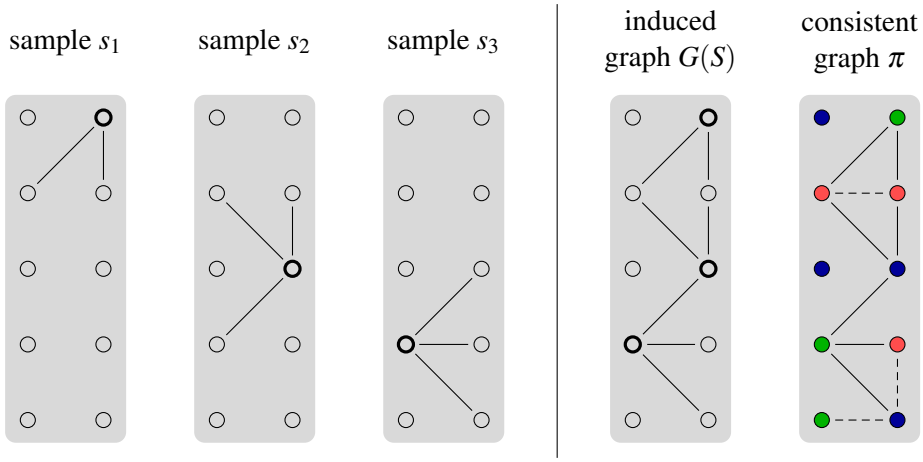


Figure 4.9: Illustration of the universe and pattern class from the proof of Theorem 4.45. The definition of $\mathcal{M}_{\text{GCOL}}$ ensures that an edge $\{u,v\}$ that occurs in some sample s_i must also occur in every other sample where u or v is the root, otherwise there exists no S-consistent pattern. For example, if the common edge of sample s_2 and s_3 were missing in one of these samples, then there would be no pattern consistent with both $(s_2,-1)$ and $(s_3,-1)$. In this fashion, the negatively labeled samples from S describe an induced graph $G(S)$. The pattern language $\mathcal{P}_{\text{GCOL}}$ is the set of all colored graphs, and a graph $\pi \in \mathcal{P}_{\text{GCOL}}$ is S-consistent if and only if it contains all edges of $G(S)$ and its nodes are validly colored with respect to $G(S)$. Edges missing in $G(S)$ may or may not be present in π unless they do not touch a node that appeared as a root in a sample from S, like the dashed edges in the rightmost graph.

insert $\{u,v\}$ into E with probability 1/2. The resulting graph is sampled at uniform from $\text{GCOL}[S]$. An example result of this process is depicted as the rightmost graph in Figure 4.9.

Step 2. So far we have only provided a pattern class GCOL for which computing $|\text{GCOL}[S]|$ is #P-hard, even though we can sample from $\text{GCOL}[S]$ uniformly at random in expected polynomial time. Now we augment GCOL as follows: For each layer L, we insert a special pattern $\hat{\pi}^{(L)}$ into $\mathcal{P}^{(L)}_{\text{GCOL}}$ and a special element $\hat{x}^{(L)}$ into $\mathcal{U}^{(L)}_{\text{GCOL}}$ such that $\hat{\pi}^{(L)}$ matches only $\hat{x}^{(L)}$ and vice versa. From now on, let GCOL denote this augmented pattern class. Suppose we had access to an oracle for computing $\Delta_{\text{GCOL}}(S,x)$. Then we could use this oracle to count the k-colorings of a graph $G=(V,E), V=[1,L]$, as follows: We generate as explained above a negative sample S with $G(S)=G$. Then

$$\Delta_{\text{GCOL}}(S,\hat{x}^{(L)}) = \frac{1}{1+\#\chi(G)\,2^{\varepsilon}}\,.$$

Hence, we could compute $\#\chi(G)$ from $\Delta_{\text{GCOL}}(S,\hat{x}^{(L)})$ by rearranging the above,

which implies that computing $\Delta_{\text{GCOL}}(S, \hat{x}^{(L)})$ is #P-hard in the sense defined at the beginning of this section.

However, we can sample from the S-consistent patterns of any negative sample S, and thus implement a sampling negative selection algorithm (Definition 4.22) in expected polynomial time. To show this, we need to distinguish two cases: If S contains $\hat{x}^{(L)}$, then $\hat{\pi}^{(L)}$ is not S-consistent, and we output a pattern sampled at uniform from $\text{GCOL}[S]$. Otherwise, $\hat{\pi}^{(L)}$ is S-consistent, and we apply Lemma 4.46 to sample a pattern at uniform from $\text{GCOL}[S] \cup \{\hat{\pi}^{(L)}\}$. □

Even though we have now seen that the complexity bound provided by Proposition 4.44 is not sharp, the bound is still very useful: For many important pattern classes we can compute the pattern sampling distance and thereby effectively derandomize the corresponding sampling negative selection algorithms. This includes the r-chunk and r-contiguous patterns, for which we have already solved the consistency problems efficiently. Because generalization via the pattern sampling distance is even possible for the 1-slicing, unrestricted versions of these pattern classes, it would also be useful to determine the complexity of the corresponding unrestricted counting problems. However, we limit our proofs here to the restricted versions; our efficient algorithms for computing the restricted pattern sampling distance can be easily adapted to computing the general version because they all involve computing $|\mathcal{C}[S, r]|$ and $|\mathcal{C}[S \cup (m, +), r]|$, which then can be summed over all matching radii r. As usual, we assume all input samples to be syntactically valid by containing only strings of equal length.

Theorem 4.47 (Tractability of r-Chunk Patterns)**.** *The sampling distance $\Delta_{r\text{-CHUNK}}$ is polynomial time computable.*

Proof. For all negative samples $S \subseteq \Sigma^L \times \{-1\}$ and all strings $x \in \Sigma^L$,

$$\Delta_{r\text{-CHUNK}}(S, x) = \frac{|\{i \in \{1, \ldots, L-r+1\} : S \text{ avoids } x \text{ at position } i\}|}{\sum_{i=1}^{L-r+1} |\Sigma|^r - |S[i \ldots i+r-1]|} .$$

The equation is correct because a string x can match at most one r-chunk pattern per index $i \in \{1, \ldots, L-r+1\}$ (numerator), and the number of S-consistent r-chunk patterns per index i is equal to the number of strings of length r that do not occur in any $s \in S$ at position i (denominator). □

As in the exhaustive case, the proof of the following theorem is postponed to Section 4.7.

Theorem 4.48 (Tractability of r-Contiguous Patterns)**.** *The pattern sampling distance $\Delta_{r\text{-CONT}}$ is computable in polynomial time.*

For the other two string pattern classes from Section 4.4.3, the pattern sampling distance is unlikely to be polynomial time computable because an algorithm that computes it could be used to solve the corresponding 1-positive consistency problem.

Proposition 4.49. *Let $\mathcal{C}$ be any pattern class for which the* 1*-positive consistency problem is NP-hard. Then there exists no polynomial time algorithm for computing $\Delta_{\mathcal{C}}$ unless P=NP.*

Proof. We can use a $\Delta_{\mathcal{C}}$-oracle for solving the 1-positive consistency problem as follows. Given the input $S \cup \{(x,+1)\}$, compute $d := \Delta_{r\text{-WILDCARD}}(S,x)$. If $d = \perp$, then reject the input (there is no S-consistent pattern). Otherwise, reject the input if $d = 0$ (there is no S-consistent pattern matching x) and accept otherwise. □

In particular, this yields the following two negative results.

Corollary 4.50 (of Proposition 4.49 and Theorem 4.42)**.** *There exists no polynomial time algorithm for computing $\Delta_{r\text{-WILDCARD}}$ unless P=NP.*

Corollary 4.51 (of Proposition 4.49 and Theorem 4.43)**.** *There exists no polynomial time algorithm for computing $\Delta_{r\text{-HAMMING}}$ unless P=NP.*

4.7 Efficient Negative Selection By Pattern Compression

Our computational complexity analysis in the previous section was not concerned with negative selection algorithms themselves, but rather with the complexity of the *functions computed by* such algorithms. We argued that the algorithms themselves are (for most pattern classes) inherently inefficient, as they will typically need exponentially many patterns to achieve a sufficient coverage of the universe. In this section, we discuss a general technique that can be applied to overcome this limitation while still preserving the general structure of the algorithm, namely *pattern compression*. Using this technique, we can construct compact representations of the pattern sets P used by negative selection algorithms, and use these for classification instead of the patterns themselves. Typically, our goals will be to construct this representation in polynomial time in the training phase, and to achieve at the same time a fast (ideally: linear time) classification of each input element in the classification phase. Our representations will also allow to efficiently perform several operations on the pattern set itself, like counting the number of S-consistent patterns, removing or inserting patterns, computing the union or intersection of two pattern sets and so on. This can be very useful especially in semi-supervised learning algorithms that use feedback mechanisms to tune a pattern set for optimal classification performance, as it is done for example in *learning classifier systems* [73] that are similar to negative selection algorithms.

We illustrate this technique using the restricted pattern types r-CHUNK and r-CONT defined in Section 4.4.4. For r-CONT in particular, the solution is rather intricate, and requires using some advanced string processing techniques. We thus start by defining the components of our string processing toolkit.

4.7.1 String Processing Tools

For our general terms related to strings and substrings, see Definition 4.7.

Definition 4.52 (Prefix DAGs)**.** *A* prefix DAG *D is a directed acyclic graph whose edges are labeled with symbols from Σ such that for all $\sigma \in \Sigma$, every node has at most one outgoing edge labeled with σ. In analogy to trees, a* root *in the prefix DAG is a vertex with indegree 0, and a* leaf *is a vertex with outdegree 0. We write $s \in D$ if there is a root ρ and a leaf λ in D with a path from ρ to λ such that s is the concatenation of the labels on the path. Given root $\rho \in D$, the language $\mathcal{L}(D,\rho)$ contains all strings s where $s[1\dots i] \in D$ for some $i \geq 1$. If D has only one root ρ, we write for short $\mathcal{L}(D)$ instead of $\mathcal{L}(D,\rho)$.*

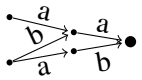


Figure 4.10: A prefix DAG

For the prefix DAG D displayed in Figure 4.10, which has two roots, we have for example $aa, ba, ab \in D$, but $bb \notin D$. If we denote the upper root vertex of D by ρ, then $\mathcal{L}(D,\rho)$ consists of all strings starting with aa. A special case of prefix DAGs are prefix trees. For the prefix tree T shown in Figure 4.11 to the right, $\mathcal{L}(T)$ is the language of all strings that start with either a or ba.

Figure 4.11: A prefix tree

Definition 4.53 (Deterministic Finite Automata (DFA))**.** *A* finite automaton *is a tuple $M = (Q, q_i, Q_a, \Sigma, \Delta)$, where Q is a set of* states*, $q_i \in Q$ is the* initial state*, $Q_a \subseteq Q$ is the set of* accepting states*, Σ is an alphabet, and $\Delta \subseteq Q \times \Sigma \times Q$ the* transition relation*, which is* deterministic*: for every $q \in Q$ and every $\sigma \in \Sigma$ there is at most one $q' \in Q$ with $(q,\sigma,q') \in \Delta$. M can be represented as a graph with node set Q and labeled edges (a σ-labeled edge from q to q' if $(q,\sigma,q') \in \Delta$). M is said to* accept *a string s if there exists a path (which need not be simple) from q_i to some $q \in Q_a$ whose concatenated edge labels equal s. The language $\mathcal{L}(M)$ contains all strings accepted by M.*

Note that every prefix tree or prefix DAG can be turned into a DFA by setting one of its roots as the initial state, setting Q_a to the set of leaves, and adding self-loops to every leaf for every $\sigma \in \Sigma$.

Definition 4.54 (Mealy Automata)**.** *A* Mealy automaton *M is defined as a tuple $M = (Q, q_i, Q_a, \Sigma, \Delta, \Omega, \omega)$ where $(Q, q_i, Q_a, \Sigma, \Delta)$ is a finite automaton, Ω is the* output alphabet*, and $\omega : \Delta \to \Omega$ is the* output function*. Let $m \in \Sigma^*$ and $t_1, \dots, t_{|m|} \in \Delta$ be the sequence of transitions made by M for input m, then the* output of *M* on input *m is the string $\omega(M,m) = \omega(t_1)\dots\omega(t_{|m|}) \in \Omega^*$. If Ω is a set of numbers, we define the accepted language $\mathcal{L}(M,r)$ to be the set of strings $m \in \Sigma^*$ where there exists an $i \leq |m|$ with $\sum_{j=1}^{i} \omega(M,m)[j] \geq r$.*

Note that the accepted language of this Mealy automaton does not depend on the accepting states, but only on the outputs of the transitions. Like a DFA, a Mealy automaton can be represented by a graph where every edge label is a pair of the symbol triggering the corresponding transition and the output of the transition. For the Mealy automaton M displayed in Figure 4.12 to the right, we have $ba \in \mathcal{L}(M,2)$ and $a \in \mathcal{L}(M,1)$, but $a \notin \mathcal{L}(M,2)$. For a more detailed discussion of automata-based string processing, we refer to the textbook of Crochemore, Hancart and Lecroq [33].

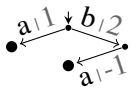


Figure 4.12: A Mealy automaton

4.7.2 Compressing r-Chunk Patterns

Since an r-chunk pattern (π, i) is simply a fixed-length string π that does not occur in the input sample S at position i, a compressed representation of r-chunk patterns is easily constructible. The results in this section are thus straightforward examples for the pattern compression approach and its applications; for the r-contiguous patterns in the next section, we will require substantially more intricate constructions.

As in the previous section, we assume that input samples contain only strings of equal length L; recall that for all string pattern types layered by $\lambda(x) = |x|$ (see Definition 4.6), including r-chunk and r-contiguous patterns, no consistent patterns exist for samples containing two strings of different lengths.

Lemma 4.55 (Compressing r-Chunk Patterns)**.** *For every $r \in \mathbb{N}$ and every negative sample $S \subseteq \Sigma^L \times \{-1\}$, one can construct in time $O(|S|\,L\,r\,|\Sigma|)$ a set of prefix trees $T_i, i \in \{1,\ldots,L-r+1\}$, with the following property: For every $w \in \Sigma^r$, $\diamond^i w \diamond^j \in r\text{-CHUNK}[S,r]$ if and only if $w \in \mathcal{L}(T_i)$.*

Proof. A string $\diamond^i w \diamond^j, w \in \Sigma^r$, is in $r\text{-CHUNK}[S,r]$ if and only if S avoids w at position $i+1$. Hence, for each prefix tree T_i we simply need to ensure that $\mathcal{L}(T_i) \cap \Sigma^r = \Sigma^r \setminus S[i \ldots i+r-1]$. Each prefix tree T_i is constructed as follows: We start with an empty prefix tree and insert into it every $s \in S[i \ldots i+r-1]$. Now we "invert" the resulting tree as follows: for every non-leaf node n and every $\sigma \in \Sigma$ for which no edge with label σ starts at n, we create a new leaf n' and an edge (n,n') labeled with σ. Finally, we delete every node from which none of the newly created leaves is reachable. The claimed runtime follows because each T_i can be constructed in time $O(|S|\,r\,|\Sigma|)$. □

See Figure 4.13 for an example of this construction. Using the prefix trees T_i, we can compute the pattern sampling distance for r-chunk patterns in a straightforward fashion.

Theorem 4.56 (Computing the r-Chunk Pattern Sampling Distance)**.** *There exists an algorithm that, given the input $S \subseteq \Sigma^L \times \{-1\}, X \subseteq \Sigma^L, r \leq L$, for each $x \in X$ outputs the r-chunk pattern sampling distance $\Delta_{r\text{-CHUNK}}(S,x)$ in time $O(L)$ per $x \in X$ after an $O(|S|\,L\,r\,|\Sigma|)$ preprocessing time.*

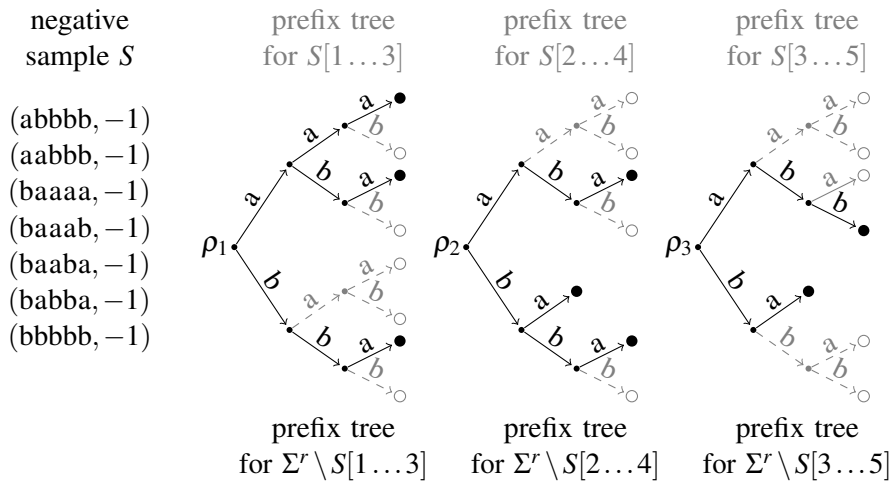


Figure 4.13: Illustration of the prefix tree construction in Lemma 4.55 for $\Sigma = \{a,b\}, L = 5, r = 3$: First all substrings of length 3 are inserted into an empty prefix tree, such that each string corresponds to a path from the root (level 0) to a leaf on level $r-1$ (dashed gray lines and open circles). These trees are then inverted by creating for each missing outgoing edge at a non-leaf node a new edge with an attached leaf (black solid lines and filled circles). The resulting prefix trees encode the S-consistent 3-chunk patterns. Note that the inverted trees can have leaves on levels above $r-1$.

Proof. According to Theorem 4.47, the pattern sampling distance is computed by counting the indices $i \in \{1,\dots,L-r+1\}$ where $x[i\dots i+r-1] \in \mathcal{L}(T_i)$ and dividing the result by $\sum_{i=1}^{L-r+1} |\mathcal{L}(T_i) \cap \Sigma^r|$. Because the latter term does not depend on x, it can be computed after constructing the T_i according to the fact that

$$|\mathcal{L}(T_i)| = \sum_{l \text{ is a leaf of } T_i} |\Sigma|^{(r-1)-\text{level}(l)} \quad ,$$

where $\text{level}(l)$ denotes the distance of the leaf l from the root of T_i ranging from 0 to $r-1$. Hence computing $|\mathcal{L}(T_i)|$ requires just one traversal of T_i. After this preprocessing, $\Delta_{r\text{-CHUNK}}(S,m)$ can be computed for each $x \in X$ in time $O(L)$. □

If we are not interested in the pattern sampling distance but only in simulating a negative selection algorithm $\mathfrak{N}$ with maximal coverage, we can weave the prefix trees together into a finite automaton that simulates the classification outcome of $\mathfrak{N}$ in merely linear time per element to classify.

Theorem 4.57 (Computing the Maximal Coverage for r-Chunk Patterns)**.** *Given any $S \subseteq \Sigma^L \times \{-1\}$ and $r \in \{1,\dots,L\}$, a finite automaton M with $\mathcal{L}(M) \cap \Sigma^L = \mathcal{L}(r\text{-CHUNK}[S])$ can be constructed in time $O(|S|\,L\,r\,|\Sigma|)$.*

Proof. First we again construct the prefix trees T_i of Lemma 4.55. We construct an automaton by inserting *failure links* [33] between the prefix trees of adjacent levels, similar as in the well-known algorithm of Knuth, Morris and Pratt [89]. The idea is as follows: If a mismatch occurs in a prefix tree T_i at a position k, then we need not restart from the root of tree T_{i+1}, but can go directly to the node in T_{i+1} that corresponds to the last $k-1$ symbols read. By inserting the failure links from right to left, turning the prefix trees into a prefix DAG D, we can inductively ensure that either such a node exists or there is no match at all.

We start by letting D be the disjoint union of $T_1,\dots,T_{L-r+1}$. Then we process the levels from $i = L-r$ down to 1 iteratively as follows: Consider every node n from T_i and every symbol $\sigma \in \Sigma$ where T_i has no outgoing edge with label σ. Let s be the string on the path from the root of T_i to n. Let $s' = s\sigma$ and let n' be the end node of the path from the root of T_{i+1} that is labeled by $s'[2\dots|s'|]$. If this n' exists, we insert an edge from n to n' with label σ. By induction one can show that after every iteration i we have $\mathcal{L}(T_i) \cap \Sigma^{L-i+1} = \mathcal{L}(r\text{-CHUNK}[S[i,\dots,L]])$. This construction is illustrated in Figure 4.14.

Finally, we turn D into a finite automaton with the claimed property by making all leaves accepting states with self-loops for all $\sigma \in \Sigma$ and setting the initial state to the root ρ_1.

As mentioned above, each prefix tree T_i can be constructed in time $O(|S|\,r\,|\Sigma|)$. The failure links between each pair of adjacent levels i and $i+1$ can be inserted in time $|S|r|\Sigma|$ by a simultaneous recursive traversal of T_i and T_{i+1}. Since the number of levels is $L-r+1$, we obtain the claimed runtime. □

This construction gives us the following version of Theorem 4.56, which yields a better runtime for simulating exhaustive negative selection.

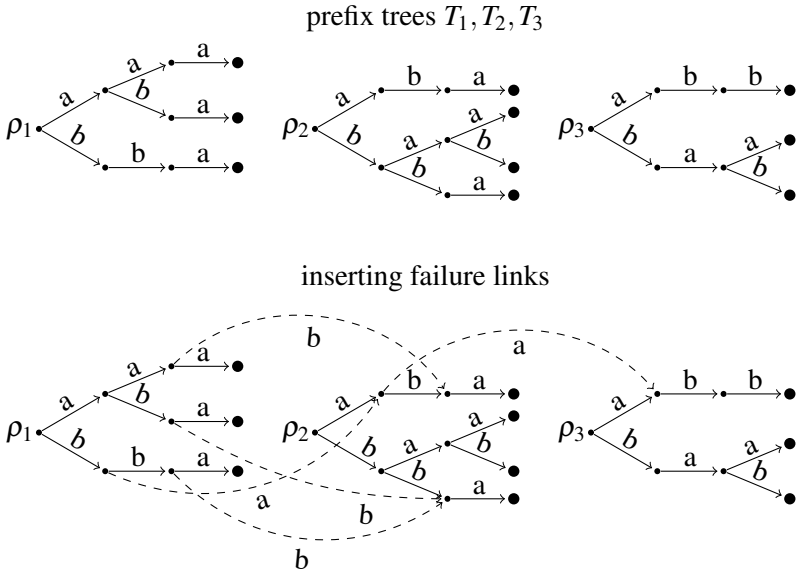
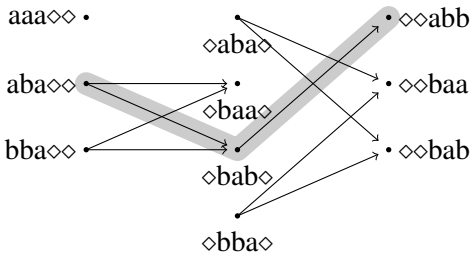


Figure 4.14: Illustration of the insertion of failure links (dashed edges) between adjacent prefix trees in Theorem 4.57 for the set of trees from Figure 4.13. The DAG with failure links can be turned into a finite automaton M by turning all leaves (filled black circles) into accepting states with self-loops for all $\sigma \in \Sigma$. For this automaton, we then have $\mathcal{L}(M) \cap \Sigma^L = \mathcal{L}(3\text{-CHUNK}[S])$. Thus, we can subsequently simulate the output of a negative selection algorithm with 3-chunk pattern in linear time $O(L)$ per input string.



derivation of contiguous pattern from the highlighted path:

3-chunk patterns from the path	aba◇◇
	◇bab◇
	◇◇abb
3-contiguous pattern	ababb

Figure 4.15: For the negative sample from Figure 4.13, this figure shows how one can construct 3-contiguous patterns from overlapping 3-chunk patterns. The 3-chunk patterns are arranged in levels and there is a directed edge from a pattern π in level i to a pattern π' in level $i+1$ if $\pi[i+1,\ldots,i+r-1] = \pi'[i,\ldots,i+r-2]$. Every path from the leftmost to the rightmost level of the graph corresponds to a 3-contiguous pattern. Figure adapted from Elberfeld and Textor [42].

Corollary 4.58 (Simulating Exhaustive Negative Selection with r-Chunk Patterns)**.** *There exists an algorithm that, given the input $S \subseteq \Sigma^L \times \{-1\}, X \subseteq \Sigma^L, r \leq L$, produces the output of an exhaustive negative selection algorithm using r-chunk patterns in time $O(L)$ for each $x \in X$ after an $O(|S|\,L\,r\,|\Sigma|)$ preprocessing time.*

4.7.3 Compressing r-Contiguous Patterns

For r-contiguous patterns, the construction of a compressed representation is substantially more difficult since the r-contiguous matching function combines locally overlapping substrings, in contrast to the r-chunk matching function which is a simple substring test. It had even been thought that constructing a single r-contiguous pattern for a given negative sample S is an NP-complete problem [149]. Fortunately, this turns out not to be the case,

The key idea behind our construction is that any S-consistent r-contiguous pattern is composed of a set of overlapping S-consistent r-chunk patterns. If we arrange all S-consistent r-chunk patterns in a graph with edges between adjacent overlapping patterns (Figure 4.15), an r-contiguous pattern corresponds to a path from the leftmost to the rightmost level in this graph. This graph has elsewhere been called the *crossover closure* [45, 139]. We based preliminary versions of the algorithms presented in this section on the crossover closure [42, 99] because we anticipated that they would then be more easy to understand by the artificial immune systems community. However, these constructions result in rather slow algorithms and are not easily generalizable beyond binary alphabets. For this reason the algorithms presented in this section are again based on prefix trees.

Formally, we can express the correspondence between r-chunk and r-contiguous patterns via the following terms.

Definition 4.59 ((S,r)-Avoiding Completions of Chunk Patterns)**.** *Let $S \subseteq \Sigma^L$, $r \leq L$. Let $\pi = \diamond^i v \diamond^j$ with $|v| \leq r$ and $i + j + |v| = L$ be an S-consistent chunk pattern.*

The chunk pattern $\pi_r = \diamond^i vw, w \in \Sigma^j$, is called an (S,r)-avoiding right-completion *of π if for all $k \in \{i+1,\ldots,L-r+1\}$, there exists some $r' \leq r$ such that S avoids $\pi[k,\ldots,k+r'-1]$ at position k. Similarly, the chunk pattern $\pi_r = uv\diamond^j, u \in \Sigma^i$, is called an* (S,r)-avoiding left-completion *of π if for all $k \in \{1,\ldots,i-1\}$, there exists some $r' \leq r$ such that S avoids $\pi[k,\ldots,k+r'-1]$ at position k.*

For instance, consider the pattern $\pi = \diamond bab\diamond$ in Figure 4.15. Then $\diamond babb$ is an $(S,3)$-avoiding right-completion of π and $abab\diamond$ is an $(S,3)$-avoiding left-completion of π. Directly from this definition, we obtain the following.

Corollary 4.60. *Let $S \subseteq \Sigma^L$, $r \leq L$. A pattern $\pi \in \Sigma^L$ is an S-consistent r-contiguous pattern if and only if there exists a decomposition $\pi = uvw$, $1 \leq |v| \leq r$, such that $\pi' = \diamond^i v \diamond^j, i = |u|, j = |w|$ is an S-consistent r-chunk pattern, $\diamond^i vw$ is an (S,r)-avoiding right-completion of π', and $uv\diamond^j$ is an (S,r)-avoiding left-completion of π'.*

This observation enables us to weave the prefix trees that we constructed to represent S-consistent r-chunk patterns together in such a way that a prefix DAG is constructed that represents the S-consistent r-contiguous patterns.

Definition 4.61 (Leveled Prefix DAG)**.** *A* leveled prefix DAG *is a prefix DAG D with k ordered roots $\rho_1,\ldots,\rho_k$ such that for all $j \leq k$, it holds that $\mathcal{L}(D,\rho_1)[j\ldots k] = \mathcal{L}(D,\rho_j)[1\ldots k-j+1]$. We write for short $\mathcal{L}(D)$ instead of $\mathcal{L}(D,\rho_1)$.*

Lemma 4.62 (Leveled Prefix DAG for r-Contiguous Patterns)**.** *Given any $S \subseteq \Sigma^L \times \{-1\}$ and $r \in \{1,\ldots,L\}$, a leveled prefix DAG D with roots $\rho_1,\ldots,\rho_{L-r+1}$ can be constructed in time $O(|S|Lr|\Sigma|)$ such that the following holds: For every $i \in \{1,\ldots,L-r+1\}$, we have $\mathcal{L}(D,\rho_i) \cap \Sigma^{L-i+1} = \mathcal{L}(r\text{-CONT}[S])[i\ldots L]$.*

Proof. The construction of D is done in four phases, presented and discussed in the next four paragraphs. While the following proof text explains the basic ideas and their correctness, the detailed computational steps are shown by the pseudocode in Figure 4.18.

Construct prefix trees: For every $i \in \{1,\ldots,L-r+1\}$, let T_i be the prefix tree with $\mathcal{L}(T_i) \cap \Sigma^r = \Sigma^r \setminus S[i\ldots i+r-1]$ from Lemma 4.55.

By definition we know that for every r-contiguous pattern π and every $i \in \{1,\ldots,L-r+1\}$, π contains a string at position i that is in T_i. However, conversely there may be strings in T_i that do not occur in any r-contiguous pattern at position i. Those are precisely the strings whose corresponding r-chunk patterns have no (S,r)-avoiding left-completion or no (S,r)-avoiding right-completion. We will remove these strings in the upcoming two steps. For the correctness of this process, the following property of pairs (T_i,T_{i+1}) of adjacent trees is crucial.

Fact 4.63. *Let (T_i,T_{i+1}) be a pair of prefix trees on adjacent levels $i,i+1$ from Lemma 4.55. For each $s \in T_i$ with $|s| \geq 2$, if there is no path from the root of T_{i+1} labeled with $s[2\ldots|s|]$ then $s[2\ldots|s|] \notin \mathcal{L}(T_{i+1})$.*

Proof. Suppose the converse, i.e., $s[2\dots|s|] \in \mathcal{L}(T_{i+1})$ and there exists a proper nonempty prefix s' of $s[2\dots|s|]$ with $s' \in T_{i+1}$. The way that T_i was constructed ensures that all proper prefixes of every $s \in T_i$ occur in S at position i. This implies that s' occurs in S at position $i+1$, which contradicts $s' \in T_{i+1}$. □

Trim the trees in a right-to-left pass: We trim the trees $T_1,\dots,T_{L-r+1}$ to obtain new trees $T_1^{(\mathrm{R})},\dots,T_{L-r+1}^{(\mathrm{R})}$ where every $T_i^{(\mathrm{R})}$ contains exactly the strings from T_i that have (S,r)-avoiding right-completions. This already holds for all strings from the rightmost level (for which there is nothing to complete), so $T_{L-r+1}^{(\mathrm{R})} = T_{L-r+1}$. We trim the other trees in a right-to-left pass from $i = L-r$ down to 1. Each time we initialize $T_i^{(\mathrm{R})}$ to be the empty prefix tree. Then we consider every string $s \in T_i$ and insert it into $T_i^{(\mathrm{R})}$ if $s[2\dots|s|]$ is a prefix of some $s' \in T_{i+1}^{\mathrm{R}}$. There are two potential reasons for a string $s \in T_i$ not to be contained in T_i^{R}: (1) It may be the case that no string from T_{i+1} starts with $s[2\dots|s|]$. Then, due to Fact 4.63 above, $s[2\dots|s|]$ occurs in S at position $i+1$ because $s[2\dots|s|] \notin \mathcal{L}(T_{i+1})$. Hence $(s[2\dots|s|], i+1)$ has no (S,r)-avoiding right-completion and thus, also (s,i) has none. (2) The second possibility is that there is a string starting with $s[2\dots|s|]$ in T_{i+1}, but not in $T_{i+1}^{(\mathrm{R})}$. By induction, one can prove that this is again because $(s[2\dots|s|], i+1)$ has no (S,r)-avoiding right-completion and, therefore, also (s,i) has none. On the other hand, if $s \in T_i$ is also contained in $T_i^{(\mathrm{R})}$, there exists an (S,r)-avoiding right-completion of (s,i) consisting of overlapping strings from the trees $T_i^{\mathrm{R}},\dots,T_{L-r+1}^{\mathrm{R}}$. Thus, precisely those strings from T_i that have (S,r)-avoiding right-completions end up in $T_i^{(\mathrm{R})}$.

Trim the trees in a left-to-right pass: Our next step is to construct a set of prefix trees $T_1^{(\mathrm{L})},\dots,T_{L-r+1}^{(\mathrm{L})}$ containing only those strings that have both left- and right-completions by an analogous left-to-right pass. For these trees it holds that $\mathcal{L}(T_i^{(\mathrm{L})}) \cap \Sigma^r = r\text{-}\textsc{cont}[S][i\dots i+r-1]$.

Weave the trees together into a prefix DAG: Finally, we weave the trees together into a prefix DAG as follows: For the rightmost level $i = L-r+1$, we set $D_{L-r+1} = T_{L-r+1}^{(\mathrm{L})}$. This gives $\mathcal{L}(T_{L-r+1}^{(\mathrm{L})}) \cap \Sigma^r = r\text{-}\textsc{cont}[S][L-r+1\dots L]$.

Now we prove the lemma by decreasing induction on i going from $i = L-r$ down to 1. For the induction step, assume we have a prefix DAG D_{i+1} with $\mathcal{L}(D_{i+1}) \cap \Sigma^{L-i} = r\text{-}\textsc{cont}[S][i+1\dots L]$. For $s \in T_i^{(\mathrm{L})}$, let n' denote the corresponding leaf in $T_i^{(\mathrm{L})}$. Let n'' denote the end node on the path from the root of $T_{i+1}^{(\mathrm{L})}$ with label s, which exists by induction assumption because $s[2\dots|s|]$ is a prefix of some $d \in r\text{-}\textsc{cont}[S][i+1\dots L]$. Create a new edge (n,n'') labeled with σ, where n is the parent of n'. Then delete the leaf n' and the old edge (n,n'). After all leaves have been iterated through, let D_i be the resulting graph. Let $d \in r\text{-}\textsc{cont}[S][i\dots L]$. Then d starts with a prefix from $T_i^{(\mathrm{L})}$ and, thus, $d[2\dots|d|] \in \mathcal{L}(D_{i+1})$. Hence, $d \in L(D_i)$ by construction. Conversely, let $d \in L(D_i)$ with $|d| = L-i+1$. Then d starts with a nonempty prefix that has both an (S,r)-avoiding right-completion and an (S,r)-avoiding left-completion. Furthermore, $d[2\dots|d|] \in L(D_{i+1})$. Hence

$d \in r\text{-CONT}[S][i \dots L]$. Now by setting $D = D_1$ we obtain a leveled DAG with the properties claimed by the Lemma.

The runtime of the construction can be determined from the pseudocode given in Figure 4.18. As stated in Lemma 4.55, constructing the prefix trees in lines 1 and 2 takes time $O(|S|Lr|\Sigma|)$. The inner loops in the right-to-left passes in lines 3–7 and 13–19 as well as in the left-to-right pass in lines 8–12 can be implemented by a simultaneous recursion through the trees on adjacent levels in time $O(|S|r|\Sigma|)$ per iteration. This yields a worst-case runtime of $O(|S|Lr|\Sigma|)$ for each of the passes and, hence, of the overall algorithm. □

The leveled prefix DAG can now be used in several ways. The most obvious one is to turn it into a finite automaton whose accepted language is the set of r-contiguous patterns.

Corollary 4.64 (Compressing r-Contiguous Patterns)**.** *Given any $S \subseteq \Sigma^L \times \{-1\}$ and $r \in \{1, \dots, L\}$, a deterministic finite automaton M with $\mathcal{L}(M) \cap \Sigma^L = r\text{-CONT}[S]$ can be constructed in time $O(|S|Lr|\Sigma|)$.*

Proof. Use the prefix DAG from Lemma 4.62 and turn it into a finite automaton with initial state ρ_1. □

The prefix DAG constructed in Lemma 4.62 also yields a straightforward algorithm for deciding membership of a string $m \in \Sigma^L$ in the set $\mathcal{L}(r\text{-CONT}[S])$ in time $O(Lr)$: For each index $i \in \{1, \dots, L-r+1\}$, check whether $m[i \dots i+r-1]$ lies in $\mathcal{L}(D, \rho_i)$. If one index exists for which this is true, then m lies in $\mathcal{L}(r\text{-CONT}[S])$. Again we can speed up the classification to linear time $O(L)$ using failure links and by extending our prefix DAG with edge outputs, which turns it into a Mealy automaton.

Lemma 4.65 (Mealy Automaton for Maximal r-Contiguous Coverage)**.** *There exists an algorithm that, given a negative sample $S \subseteq \Sigma^L \times \{-1\}$ and $r \in \{1, \dots, L\}$, constructs a Mealy automaton M with output alphabet $\Omega = \{-r, \dots, r\}$ such that $\mathcal{L}(M, r) \cap \Sigma^L = \mathcal{L}(r\text{-CONT}[S])$ in time $O(|S|Lr|\Sigma|)$.*

Proof. Let M be the finite automaton constructed in the proof of Corollary 4.64 and let $\rho_1, \dots, \rho_{L-r+1}$ be the roots of its underlying graph. We turn M into a Mealy automaton with output alphabet $\Omega = \{-r, \dots, r\}$ such that $\mathcal{L}(M, r) \cap \Sigma^L = \mathcal{L}(r\text{-CONT}[S])$ holds. We describe the main ideas of the construction and discuss its correctness. For a presentation of the detailed computation steps, we refer to the pseudocode in Figure 4.19.

We start by assigning to all existing transitions of M the output 1. Our aim is to transform M in a right-to-left pass that inductively ensures the following property: Let $m \in \Sigma^L$ and let $1 \leq i \leq j \leq L$. Let $k \geq 0$ denote the length of the longest suffix of $m[i \dots j]$ that is also a suffix of some $d' \in r\text{-CONT}[S][i \dots j]$. If $k \geq r - L + j$, then there exists a path from ρ_i for $m[i \dots j]$, and the sum of outputs on this path is equal to k. Otherwise there is no such path. Hence, if such a path exists and we

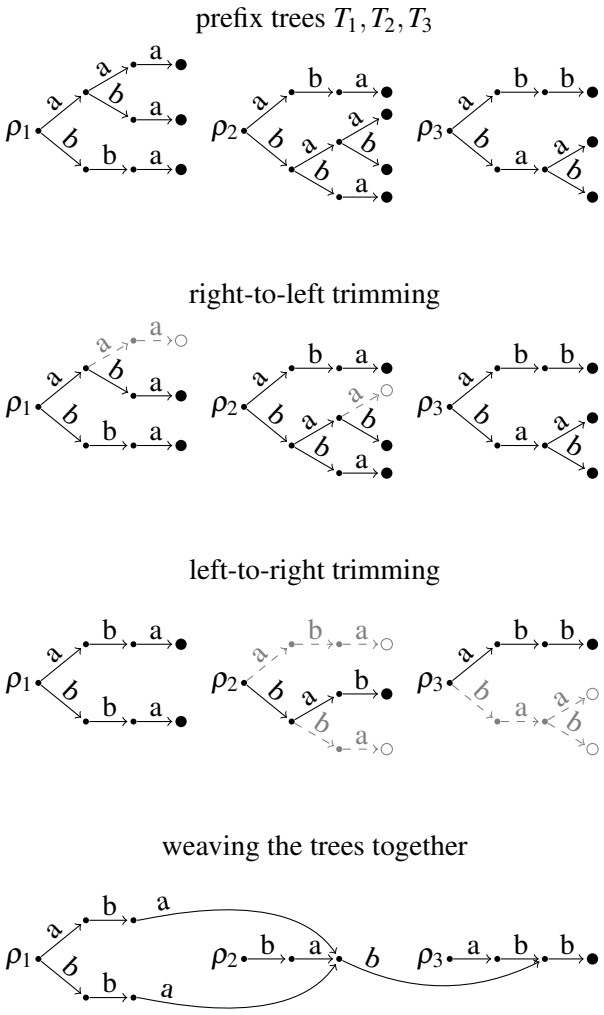


Figure 4.16: Illustration of the four phases of the algorithm from Lemma 4.62 (Figure 4.18) that creates a leveled prefix DAG D with $\mathcal{L}(D) = r\text{-CONT}[S]$ for the negative input sample S from Figure 4.13 with $r = 3$.

have $k \geq r$, then $m \in \mathcal{L}(r\text{-CONT}[S])$; otherwise, k is the length of the longest partial match between $m[i \dots j]$ and some $d \in r\text{-CONT}[S][i \dots j]$ that can still be extended to length $\geq r$.

The property already holds for $i = L - r + 1$. For i decreasing from $L - r$ to 1, we iteratively transform the graph of M as follows: For every node n in M that is reachable from ρ_i, but not from ρ_{i+1}, consider all $\sigma \in \Sigma$ where n has no outgoing edge labeled with σ. Let s be the string on the path p from ρ_i to n, w be the total weight on p, and $s' = s\sigma$. If there exists a path p' labeled with $s'[2 \dots |s'|]$ from ρ_{i+1}, let w' denote the sum of weights on this path. Create an edge from n to the last node of p' and label it with $w' - w$. Now there is a path from ρ_i labeled with s' with weight w', satisfying the required property. The correctness of this procedure can be proved by induction, and we obtain a Mealy automaton whose r-threshold language has the desired property.

Similarly as in Lemma 4.62, the described transformation can be implemented in time $O(|S|\,L\,r\,|\Sigma|)$ by simultaneous recursive descent into the prefix trees on adjacent levels.

□

We can now use the previous two Lemmata to easily prove Theorem 4.41 from the previous section.

Proof of Theorem 4.41. The theorem stated that the 0- and 1-positive restricted consistency problems for r-contiguous patterns are solvable in polynomial time. For the 0-positive problem, we are given the input $S \subseteq \Sigma^L$ and construct the prefix DAG D from Lemma 4.62. We accept the input if this D contains at least one path of length L. For the 1-positive problem, we are additionally given an element $x \in \Sigma^L$. To determine whether x is matched by an S-consistent pattern, construct the Mealy automaton M from the previous Lemma and determine whether $x \in \mathcal{L}(M, r)$. □

Since the Mealy automaton M allows us to solve the 1-positive restricted consistency problem, we can use it to simulate exhaustive negative selection with r-contiguous patterns in polynomial time. This result disproves the claim by Timmis et al. that negative selection with r-contiguous patterns is "equivalent to an NP-complete problem" [149]. We obtain a linear-time classification phase after a polynomial time training phase. Table 4.1 compares the runtimes of the exhaustive negative selection algorithms obtained so far to previous results from the literature.

Theorem 4.66 (Simulating Exhaustive Negative Selection With r-Contiguous Patterns)**.** *There exists an algorithm that, given a negative sample $S \subseteq \Sigma^L \times \{-1\}$, a set $X \subseteq \Sigma^L$, and a number $r \leq L$, simulates an exhaustive negative selection algorithm using r-contiguous patterns in time $O(L)$ per $x \in X$ after an $O(|S|\,L\,r\,|\Sigma|)$ preprocessing phase.*

Finally, we discuss how to use the prefix DAG constructed in Lemma 4.62 to compute the r-contiguous pattern sampling distance in polynomial time. To this

r-chunk pattern-based algorithms	asymptotic runtime	
	training phase	classification phase
Stibor et al. [140]	$(2^r + \|S\|)(L-r+1)$	$\|P\| L$
Elberfeld, Textor [42]	$\|S\|(L-r+1)r^2$	$\|S\| L^2 r$
Present paper	$\|S\| L r$	L

r-contiguous pattern-based algorithms	asymptotic runtime	
	training phase	classification phase
D'haeseleer et al. [38] (linear)	$(2^r + \|S\|)(L-r)$	$\|P\| L$
D'haeseleer et al. [38] (greedy)	$2^r \|S\|(L-r)$	$\|P\| L$
Wierzchón [163]	$2^r(\|P\|(L-r)+\|S\|)$	$\|P\| L$
Elberfeld, Textor [42]	$\|S\|^3 L^3 r^3$	$\|S\|^2 L^3 r^3$
Algorithms in this chapter	$\|S\| L r$	L

Table 4.1: Comparison of the runtimes obtained for exhaustive negative selection using the techniques in this chapter to those of previously published algorithms. All runtimes are given for a binary alphabet ($|\Sigma| = 2$) since not all algorithms are applicable to arbitrary alphabets. The parameter $|P|$ denotes the number of patterns that are generated in the training phase, which is bounded by the runtime of the training phase. For the algorithm of Wierzchoń [163], $|P|$ is an input parameter. Table adapted from Elberfeld & Textor [43].

end, we first prove the following lemma that gives an algorithm for contiguous matching to strings in this DAG.

Lemma 4.67 (Counting Partial Matches in a Prefix DAG)**.** *Let $L \in \mathbb{N}, r \in \{1,\dots,L\}$ and let $x \in \Sigma^L$. There exists an algorithm that, given the prefix DAG D constructed in Lemma 4.62 for a negative sample $S \subseteq \Sigma^L \times \{-1\}$, determines $|\{\pi \in \mathcal{L}(D) : \mathcal{M}_{\text{CONT}}((\pi,r),x) = +1\}|$ in time $O((|V|+|E|)r)$.*

Proof. The algorithm proceeds by computing for each vertex v a set of indices $P_i(v)$ and another index $C(v)$. Intuitively, the $P_i(v)$ *(partial matches)* count the strings on paths to v that do not yet match s sufficiently, while $C(v)$ *(complete matches)* counts the strings that already match s in $\geq r$ contiguous positions, and can thus be arbitrarily extended.

More precisely, let $v \in V$, and let s' denote the prefix of s that has the same length as the strings on the paths from the root ρ_1 to v (which by construction of D are all equally long). In the following we do not distinguish between a path and the string resulting from concatenating the labels on this path. The index $P_i(v)$ denotes the number of paths p from ρ_1 to v for which the following holds: (1) The length of the longest contiguous match between p and s' is less than r; and (2) the length of the longest common suffix of s' and p is i. $C(v)$ denotes the number of paths from the root ρ_1 to v that match s in at least r contiguous positions.

The $P_i(v)$ and $C(v)$ are computed as follows. For the root ρ_1 on level 0, we set

$$P_0(\rho_1) = 1 \quad ; \quad P_{\geq 1}(\rho_1) = 0 \quad ; \quad C(\rho_1) = 0 \quad .$$

Now we iterate through the graph in breadth-first order. Denote by $l \in \{1,\dots,L\}$ the current level in this breadth-first order, i.e., the distance of the current vertex v from ρ_1. The construction of D ensures that every incoming edge e adjacent to v is labeled with the same symbol σ. We need to distinguish two cases: (1) The last symbol does not match the input string, i.e. $\sigma \neq s[l]$, in which case

$$P_0(v) = \sum_{(u,v)\in E} \sum_{i=0}^{r-1} P_i(u) \quad ; \quad P_{j\geq 1}(v) = 0 \quad ; \quad C(v) = \sum_{(u,v)\in E} C(u) \quad ;$$

and (2) the last symbol matches the input string, i.e., $\sigma = s[l]$, such that

$$P_0(v) = 0 \quad ; \quad P_{j\geq 1}(v) = \sum_{(u,v)\in E} P_{j-1}(u) \quad ; \quad C(v) = \sum_{(u,v)\in E} C(u) + P_{r-1}(u) \quad .$$

These definitions are illustrated in Figure 4.17, where case (1) corresponds to the dashed edges, and case (2) to the solid edges.

After the indices $P_i(v)$ and $C(v)$ have been computed for all leaves of D, we can determine the total number of strings in $\mathcal{L}(D)$ that match s in at least r contiguous positions through the following formula:

$$\begin{aligned} &|\{\pi \in \mathcal{L}(D) : \mathcal{M}_{\text{CONT}}((\pi,r),x) = 1\}| \\ &= \sum_{v \text{ is a leaf of } D} \left[|\Sigma|^{L-\text{level}(v)} + \sum_{i=0}^{r-1} P_i(v) \cdot F_r(L-\text{level}(v), i) \right] \end{aligned} \tag{4.2}$$

where $F_r(n,j)$ denotes the number of possibilities to extend the string from ρ_1 to v, whose suffix of length j is equal to the corresponding symbols of s, with a string of length n in such a way that the resulting string matches s in at least r contiguous positions[9]. Computing $F_r(n,j)$ is more complicated than one would perhaps expect; it boils down to a generalization of the so-called *k-step Fibonacci numbers*. A recursive form for $F_r(n,j)$ is easiest found in terms its dual $F_r^*(n,i)$ giving the number of possible extensions that do *not* fulfill the required properties:

$$F_r(n,i) = |\Sigma|^n \cdot F_r^*(n,i) \ .$$

The recursive expression for $F_r^*(n,i)$ is found through combinatorial arguments. We start with the special case $F_r^*(n,0)$. We can assume without loss of generality that the input string s has the form σ^L, with $\sigma \in \Sigma$ arbitrary but fixed (otherwise we relabel each edge from D accordingly). Then computing $F_r^*(n,0)$ leads to the question: How many strings in Σ^n do not contain the substring σ^r? The answer for $n > 1$ can be expressed recursively as

$$F_r^*(n,0) = \sum_{j=1}^{r} (|\Sigma| - 1) F_r^*(n-j,0)$$

with the termination conditions

$$F_r^*(n,0) = 0 \text{ for } n < -1 \quad ; \quad F_r^*(-1,0) = \frac{1}{|\Sigma| - 1} \quad ; \quad F_r^*(0,0) = 1 \ .$$

For $|\Sigma| = r = 2$ the above expression gives the Fibonacci numbers shifted by two indices to the left, i.e., $F_2^*(n)$ is the $(n+2)$th Fibonacci number. More generally, for $|\Sigma| = 2$ the expression gives the r-step Fibonacci numbers shifted by r indices to the left.

Now we generalize to arbitrary i, which leads to the question: How many strings $s \in \Sigma^n$ exist such that the string $\sigma^i s$ does not contain the substring σ^r with arbitrary but fixed $\sigma \in \Sigma$? The answer is

$$F_r^*(n,i) = \sum_{j=1}^{r-i} (|\Sigma| - 1) F_r^*(n-j,0) \ .$$

Thus, we now have fully defined Equation 4.2. To obtain the claimed runtime, the values $F_r(n,i)$ have to be calculated upfront for all $n \leq r$ (note that leaves in D do not appear before level $L-r$). □

The above lemma immediately leads to a polynomial time algorithm for computing $\Delta_{r\text{-CONT}}$, which is however substantially less efficient than the one for deciding membership in $\mathcal{L}(r\text{-CONT}[S])$ (Theorem 4.66). We leave it as an open problem to improve the time and space bounds given by the following corollary.

[9] Note that this problem bears some similarity to a well-known problem in probability: how large is the probability to obtain r consecutive heads when tossing a coin n times?

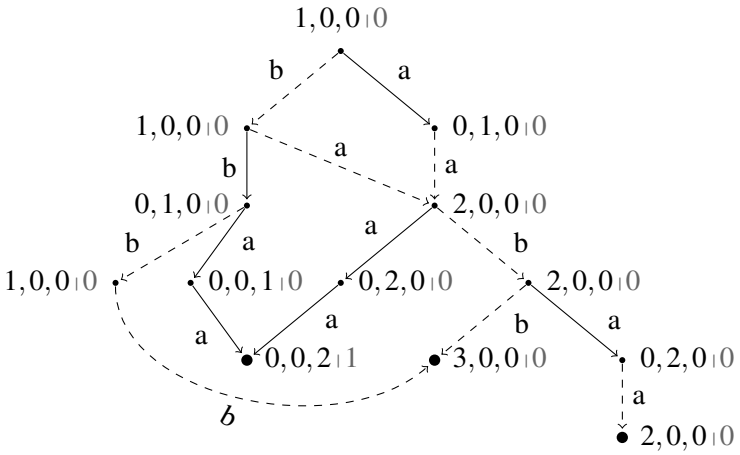


Figure 4.17: Illustration of the partial match indices defined in the proof of Lemma 4.67. The shown prefix DAG encodes the language 3-CONT$[S]$ for $S = \{(ababb,-1),(ababa,-1),(abbab,1))\}$, and the shown indices are computed with respect to the input string *abaab*. The left parts of the vertex labels denote the indices $P_i(v)$, and the right parts denote the index $C(v)$. For example, the label $0,0,2 \mid 1$ in the leftmost leaf vertex means $P_0(v) = 0, P_1(v) = 0, P_2(v) = 2$, and $C(v) = 1$. The definition in Lemma 4.67 can be understood by noting that along solid edges (which match the input string), the indices are "shifted" one position to the right (because all partial matches are extended by one symbol), while along dashed edges (which do not match), the indices "collapse" back to $P_0(v)$ (because all the partial matches are interrupted).

Corollary 4.68 (Computing the r-Contiguous Pattern Sampling Distance)**.** *There exists an algorithm that, given the input $S \subseteq \Sigma^L \times \{-\}, X \subseteq \Sigma^L, r \leq L$, outputs for each $x \in X$ the r-contiguous pattern sampling distance $\Delta_{r\text{-CONT}}(S,x)$ in time $O(|S| L r |\Sigma|)$.*

Moreover, because the prefix DAG D constructed in Lemma 4.62 allows to compute partial matches with respect to an *arbitrary* matching radius r, it is also possible to compute the entire matching profile (Section 4.5.6) for each $x \in X$ in polynomial time. This will be used in Chapter 5.

Corollary 4.69 (Computing the r-Contiguous Matching Profile)**.** *There exists an algorithm that, given the input $S \subseteq \Sigma^L \times \{-\}, X \subseteq \Sigma^L, r \leq L$, outputs for each $x \in X$ the matching profile $\Pi_{r\text{-CONT}}(S,x)$ in time $O(|S| L^3 |\Sigma|)$.*

4.8 Discussion

In contrast to the first part of this thesis where we proposed a new model, in this part we have investigated a model introduced almost two decades prior to the time

of writing [55], for which a large body of previous work exists. However, the perspective we have taken in our investigation differs from this previous work in three ways. (1) We considered the negative selection algorithm as a model of an immunological process rather than a general purpose machine learning technique. For this reason, (2) we formalized the negative selection algorithm using a notational framework akin to algorithmic learning theory. Finally, (3) our analysis distinguishes between the algorithm itself and the classification outcome of the algorithm.

We have seen that the negative selection algorithm is closely related to consistent learning [3] and version space learning [107]. We have also shown that a negative selection algorithm can generalize beyond the input data in two different ways: (1) by a parameterized restriction of the pattern class; (2) by pattern sampling and, more generally, via the matching profile. The second mechanism was previously unappreciated in the literature, and certainly plays a role in the real immune system, which generates T cells at random and in limited numbers. Generalization by restriction might be one of the factors that determine the length of the peptide chains that are presented on MHC molecules; at least for MHC class I, it has indeed been argued that this length provides just about enough information to discriminate self from nonself [107]. Our computational complexity analysis showed that negative selection algorithms using the classes of r-chunk and r-contiguous patterns, which are based on an abstract model of T cell antigen detection [116], can be simulated efficiently. This result will be exploited in the next chapter to generate testable quantitative predictions from the assumption that the r-contiguous matching rule is indeed a meaningful description of TCR-antigen binding.

We would like to point out two main directions for future work. First, we have shown the simulation of negative selection algorithms to be NP-hard for Hamming distance based patterns (Definition 4.12). These patterns are important for theoretical immunology as they have been used in more recent models like those of Košmrlj et al. [90, 91]. Thus it would be desirable to develop methods that allow to generate predictions efficiently despite the NP-completeness. An obvious approach would be to investigate whether the pattern sampling distance can be at least approximately computed. Moreover, for sampling negative selection one could drop the requirement that detector sampling has to be *exactly* uniform. For practical applications, *approximately* uniform samplers (with polynomially bounded error) are perfectly fine: the difference between both cannot be determined by an experiment of polynomially bounded duration [80]. We conjecture that if we allow for approximately uniform sampling, then the complexity bound provided by Proposition 4.44 would become a sharp one because *approximate* counting and *approximate* sampling are mutually reducible [80].

Second, our initial approach to non-consistent versions of negative selection algorithms (Section 4.5.6) should be investigated in more detail. Consistency implies that the *frequency* with which elements appear in the input sample is irrelevant: consistency with an input sample that occurs only once is considered just as important as consistency with an input sample that occurs many times. Most

biological processes are somehow stochastic, and it is reasonable to assume that this also holds for antigen presentation in the thymus – hence, it can be expected that a T cell does not always see each and every antigen in the thymus before it is allowed to leave. For this reason, we base our predictions in the next chapter on the matching profile rather than the sampling distance. Further refined negative selection algorithms could take the frequency with which samples appear in the input data explicitly into account, and relax the consistency requirement accordingly. A similar idea underlies the popular PAC learning model [152].

4.9 Epilogue: Negative Selection in Machine Learning

Thomas Stibor wrote in the conclusion to his PhD thesis [136, p. 118]:

> *"From the point of view of the author, the negative selection was thoroughly explored and has no potential for becoming a robust and useful [...] anomaly detection technique. We therefore believe that future work in this direction is not meaningful. "*

We wish to conclude this chapter by stating why we largely disagree with this conclusion. Although it is absolutely possible that negative selection may turn out to be less generic, less robust, or less widely applicable than other techniques such as one-class support vector machines [126], there is no firm theoretical basis to back up so broad a claim.

Let us review the evidence given by Stibor to support his conclusion. We focus on what he called "Hamming Negative Selection", which refers to negative selection with r-chunk and r-contiguous patterns. There appear to have been two major criticisms. (1) "The complexity to generate and to store a sufficient number of detectors is infeasibly high when applied to real-world (network) intrusion detection problems" [136, p. 116]. This statement is now disproved (Theorems 4.56 and 4.68), although we did see that there exist other pattern classes for which such a claim could be made. (2) The second issue raised was that negative selection algorithms generalize poorly on some example datasets. Although we have not explored this issue in great detail, we tend to agree that restriction based generalization (Section 4.5.5), which Stibor investigated, is presumably neither robust nor powerful. In Stibor's words, "the r-chunk length [restriction parameter] cannot be arbitrarily chosen, as it must capture the semantical representation of the elements" [136, p. 71]. In Section 4.5.5 we argue that the range of restriction parameters that give a meaningful generalization should typically indeed be very small. However, with sampling based generalization we have demonstrated a completely orthogonal mechanism, which has so far not been investigated in "applied" negative selection research. Judging from the results of our toy problem (Figure 4.7 and 4.8) and the results that will be presented in the next chapter, we would fathom a guess that sampling distance based generalization could turn out to be far more robust and flexible than exhaustive negative selection combined with restriction based generalization, which is simply an extreme special case of the former.

A last argument of Stibor that we wish to respond to is that learning methods based on statistical techniques should provide more robust results. We agree with this assessment; combinatorial learning techniques like negative selection usually have difficulties with erroneous labels, or when the underlying structure of the problem space is very complex or unknown, such that it is difficult to describe combinatorially (e.g., by a suitable pattern matching function). This problem could be addressed in part by relaxing the consistency requirement via the matching profile or other techniques – as mentioned previously, an *approximately* consistent version of negative selection would presumably be less prone to noise. One potential advantage of negative selection algorithms over many statistical anomaly detection techniques [26] is their ability to natively process string data – statistical techniques often require string data to be mapped to a metric space via appropriate feature extraction or kernelization techniques. Such a mapping can be difficult to construct, and some important information might be lost when projecting the strings to a real-valued space. Thus it may be more natural to think of the problem at hand as a pattern matching problem and construct an appropriate matching rule for negative selection.

In summary, we believe that our work has opened some promising new directions for applying negative selection to practical problems. Because such work would potentially also yield more insight into the model itself or more efficient algorithms for applying negative selection to large datasets, it would not be without purpose even if negative selection turns out to be an uncompetitive technique, because negative selection algorithms are in any case important in theoretical immunology. Finally, we point out that some of the most successful models of learning theory (e.g., inductive inference) have never yielded competitive machine learning techniques, but still their study has provided tremendously useful results. Even though we agree that "there is nothing more practical than a good theory" [154], it is too narrow a view that only a practical theory is a good one.

Algorithm CONSTRUCT-CONTIGUOUS-PATTERN-DAG(S,r)
Input: Sample $S \subseteq \Sigma^l \times -$, number $r \in \{1,\dots,L\}$
Output: Prefix DAG D with roots $\rho_1,\dots,\rho_{L-r+1}$
such that $\mathcal{L}(D,\rho_j) \cap \Sigma^L = r\text{-CONT}[S][j\dots L]$

Construct prefix trees:
1 **for** $i = 1$ **to** $L-r+1$ **do**
2 $T_i \leftarrow$ prefix tree with $\mathcal{L}(T_i) \cap \Sigma^r = \Sigma^r \setminus S[i\dots i+r-1]$

Trim the trees in a right-to-left pass:
3 $T^{(\mathrm{R})}_{L-r+1} \leftarrow T_{L-r+1}$
4 **for** $i = L-r$ **down to** 1 **do**
5 $T^{(\mathrm{R})}_i \leftarrow$ empty prefix tree
6 **for each** string $s \in T_i$ **do**
7 **if** there exists $s' \in T^{(\mathrm{R})}_{i+1}$ such that $s[2\dots|s|]$ is a prefix of s'
8 **then** insert s into $T^{(\mathrm{R})}_i$

Trim the trees in a left-to-right pass:
9 $T^{(\mathrm{L})}_1 \leftarrow T^{(\mathrm{R})}_1$
10 **for** $i = 2$ **to** $L-r+1$ **do**
11 $T^{(\mathrm{L})}_i \leftarrow$ empty prefix tree
12 **for each** string $s \in T^{(\mathrm{R})}_i$ **do**
13 **if** there exists $s' \in T^{(\mathrm{L})}_{i-1}$ such that $s'[2\dots|s'|]$ is a prefix of s
14 **then** insert s into $T^{(\mathrm{L})}_i$

Weave the trees together into a prefix DAG:
15 $D_{L-r+1} \leftarrow T^{(\mathrm{L})}_{L-r+1}$
16 **for** $i = L-r$ **down to** 1 **do**
17 $D_i \leftarrow$ disjoint union of D_{i+1} and $T^{(\mathrm{L})}_i$; $\rho_i \leftarrow$ root of $T^{(\mathrm{L})}_i$
18 **for each** string $s \in T^{(\mathrm{L})}_i$ **do**
19 $(n,\sigma,n') \leftarrow$ last labeled edge on the s-path from ρ_i in $T^{(\mathrm{L})}_i$
20 $n'' \leftarrow$ end node of the $s[2\dots|s|]$-path from ρ_{i+1} in D_{i+1}
21 delete edge (n,σ,n') from D_i and insert edge (n,σ,n'')

Output final prefix DAG with roots $\rho_1,\dots,\rho_{L-r+1}$:
22 **output** $D \leftarrow D_1$

Figure 4.18: For a given negative sample $S \subseteq \Sigma^L$ and number $r \in \{1,\dots,L\}$, this procedure constructs a leveled prefix DAG D with roots $\rho_1,\dots,\rho_{L-r+1}$ such that $\mathcal{L}(D,\rho_i) \cap \Sigma^{L-i+1} = r\text{-CONT}[S][i\dots L]$ for every $i \in \{1,\dots,L-r+1\}$ in time $O(|S|Lr|\Sigma|)$. Thus, in particular we have $\mathcal{L}(D,\rho_1) \cap \Sigma^L = r\text{-CONT}[S]$. Pseudocode adapted from Elberfeld and Textor [43].

Algorithm CONSTRUCT-CONTIGUOUS-MEALY-AUTOMATON(S, r)
Input: Sample $S \subseteq \Sigma^L \times -$, number $r \in \{1, \ldots, L\}$
Output: A Mealy automaton M with $\mathcal{L}(M, r) \cap \Sigma^L = \mathcal{L}(r\text{-CONT}[S])$

```
1   M ← Finite automaton for r-CONT[S] from Corollary 4.64
2   assign output 1 to all transitions in M
3   ρ_1, ..., ρ_{L−r+1} ← root nodes of M's graph

Insert failure links with outputs in right-to-left pass:
4   for i = L − r down to 1 do
5       for each node n reachable from ρ_i but not from ρ_{i+1} do
6           for each σ ∈ Σ where n has no outgoing σ-edge do
7               p ← path from ρ_i to n ; s ← string on p ; s' ← sσ
8               if there exists a path p' for s'[2...|s'|] from ρ_{i+1} then
9                   w ← sum of outputs on p
10                  w' ← sum of outputs on p'
11                  n' ← end node of p'
12                  create a transition (n, σ, n') with output w' − w
13  output M
```

Figure 4.19: The algorithm sketched in the proof of Lemma 4.65, which transforms the finite automaton M constructed in Corollary 4.64 into a Mealy automaton with $\mathcal{L}(M, r) \cap \Sigma^L = \mathcal{L}(r\text{-CONT}[S])$. Note that the language $\mathcal{L}(M, r)$, formalized in Definition 4.54, depends solely on the output of M, and not on M's accepting states. Pseudocode adapted from Elberfeld and Textor [43].

Chapter 5

Predicting Recognition of CD8 T Cell Epitopes

Summary. *CD8 T cells (or "T killer cells") are essential for controlling viral infections. Infected cells present viral peptides (epitopes) on surface-bound MHC class I molecules, and CD8 T cells that recognize these peptide-MHC complexes kill the infected cells, preventing further spread of the virus. The development of an anti-viral immune response is subject to two rate-limiting steps: (1) not all viral peptides can be cleaved in the proteasome and bind to an MHC molecule; (2) not all MHC-presented viral peptides trigger a strong enough T cell signal to elicit a response. Knowing which peptides are able to elicit T cell responses is essential for vaccine design and for understanding control of infection. Computer algorithms have been developed that can predict the outcome of step (1) – peptide-MHC binding – with high accuracy. However, there has been limited success so far in identifying the factors that determine the outcome of step (2).*

Here, we investigate two predictors based on the hypothesis that self-similar epitopes are less likely to be recognized by T cells. The predictors measure self similarity in two different ways: directly, by comparing self and nonself epitope sequences, and indirectly, via a model of thymic negative selection. By analyzing publicly available data, we show that both methods give significantly accurate predictions of recognition status as well as recognition frequency of HLA-A2 restricted HIV epitopes. These findings demonstrate that combinatorial restriction of the CD8 T cell repertoire through thymic negative selection is an important determining factor for epitope recognition. Our methodology provides a quantitative framework for testing the biological validity of string-based models of T cell receptor cross reactivity.

5.1 Introduction

Anti-viral CD8 T cell mediated immune responses are triggered by small chunks of foreign protein, called *epitopes*, that are displayed on MHC class I molecules.

The need to understand disease progression and immune control of notorious viral epidemics like HIV, as well as ongoing efforts to design vaccines against such pathogens, have given rise to the problem of *epitope prediction*: Given a set of viral or other pathogenic protein sequences, identify those peptides that will bind to MHC class I molecules and, thus, be presented to CD8 T cells. With predictors reaching accuracies of >95% [96] and several tools freely and publicly available [96, 112, 22, 111, 117, 64], this can almost be considered a solved problem. Such tools can substantially reduce the amount of required experimental work for identifying new epitopes, since MHC class I molecules are highly specific and only bind to around 1% of all possible 9-mers [95]. Typically, epitope predictors are based on artificial neural networks trained on datasets of MHC binders and non-binders [22]; because peptide-MHC binding can be studied *in vitro*, a large amount of such data is available. However, only about half of all MHC-presented peptides are recognized by CD8 T cells [57], and these peptides are further subdivided into a hierarchy of immunodominant, weakly recognized, and hardly recognizable (cryptic) epitopes. Because T cell recognition and immunodominance are the outcome of a highly complex process compared to the "simple" chemical reaction that is peptide-MHC binding, the factors that determine recognition and immunodominance of T cell epitopes are still poorly understood.

One important early result on this subject was obtained by Frahm et al. [56], who found that sequence entropy and proteasomal cleavage likelihood significantly correlate with HIV peptide recognition frequency. However, these metrics cannot discern whether increased recognition is due to improved cognate detection by T cells, or can rather be explained by MHC binding alone. What was needed were *mechanistic* factors for predicting recognition, which would shed light into *why* certain peptides are recognized and others are not. One possibility to obtain such metrics is to consider biochemical properties of T cell receptors (TCRs) and epitopes. However, in a study of vaccinia infection in mice [4], Assarsson et al. concluded that "neither peptide-binding affinity, nor complex stability, nor TCR avidity, nor amount of processed epitope appeared to strictly correlate with immunodominance status"; similar negative results were very recently obtained by Leger at al. for HSV in mice [97]. Thus, the biophysical characteristics alone do not seem to provide enough information for predicting peptide recognition. Another approach is to consider the similarity of viral epitopes to the human proteome (self similarity), because more self-similar epitopes should presumably be more difficult to recognize. Rolland et al. [124] and Frankild et al. [57] investigated such similarity measures. The similarity measure by Rolland et al. [124] was based on simple string matching, while Frankild et al. [57] took amino acid similarity into account using a substitution matrix. However, neither of these studies demonstrated a convincing correlation between recognition status and the proposed metric.

In this chapter, we will re-consider the similarity measure defined by Frankild et al. [57], and show that it can be improved in a biologically motivated fashion. The improved measure will be shown to provide significant predictions for both status and frequency of TCR recognition of HLA-A2 restricted HIV epitopes.

Moreover, we propose a new metric based on a simple model of TCR cross reactivity combined with a simulation of thymic negative selection, which provides significant results in a mechanistic fashion.

5.2 Results

5.2.1 Improving the Metric by Frankild et al.

Frankild et al. [57] defined a similarity measure between amino acid sequences based on the BLOSUM matrix (Section 5.4.3), with which they investigated T cell cross-reactivity and "holes" in the T cell repertoire. For each protein from the HIV HXB2 consensus sequence, they predicted 9-mers that would be presented on the HLA-A2 molecule using the software *NetCTL* (Figure 5.1A). The resulting collection of 9-mers will be called the *nonself epitopes* in the following (Figure 5.1B). For each predicted epitope, Frankild et al. searched the Los Alamos National Laboratory (LANL) HIV database [165] to verify whether the epitope had been confirmed to be detectable by CD8 T cells. This was the case for around 1/3rd of the predicted epitopes (highlighted 9-mers in Figure 5.1B). Frankild et al. hypothesized that the non-confirmed epitopes should be more similar to *self epitopes* from the human proteome, and thus fall within "holes" of the T cell repertoire. To test this hypothesis, they generated a set of self epitopes by running the NetCTL software on the entire human proteome (Section 5.4.2). For each nonself epitope, they determined the maximum similarity to a self epitope. Their expectation was that, on average, this maximum self similarity should be lower for the confirmed epitopes than for the non-confirmed epitopes. However, they did not find a significant difference between the self similarity of confirmed and non-confirmed epitopes.

We repeated their experiment using the exact same methodology on current data (Section 5.4.2), and evaluated the predictive power of their self similarity measure using a "receiver operating characteristic" (ROC) analysis. A ROC curve is generated by determining sensitivity and specificity of a threshold-based predictor for all possible thresholds. In our case, this means that we set a self similarity threshold and define all epitopes above that threshold as detectable; then, we test this prediction against the information from the LANL database. The resulting sensitivity and specificity correspond to one point of the ROC curve, and the entire curve is obtained by repeating this process for all possible self similarity values. The power of a predictor can be quantified by integrating the area under the curve (AUC): if the AUC is 0.5 (corresponding to a diagonal ROC curve), then the prediction is equivalent to random guessing, while an AUC of 1.0 would correspond to perfect prediction. This fact forms the basis of a non-parametric statistical test whether the classifier is better than random guessing. For the self similarity measure proposed by Frankild et al., we obtained an AUC value of 0.43, and no significant difference to random guessing ($p = 0.26$; Figure 5.2A).

We hypothesized that the low predictive power of the metric by Frankild et al. [57] could be due to two issues with their similarity measure. First, not all

A
```
>sp|P04601|NEF_HV1H2 Nef protein
MGGKWSKSSVIGWPTVRERMRRAEPAADRVGAASRDLEKHGAITSSNTA
ATNAACAWLEAQEEEEVGFPVTPQVPLRPMTYKAAVDLSHFLKEKGGLE
GLIHSQRRQDILDLWIYHTQGYFPDWQNYTPGPGVRYPLTFGWCYKLVP
VEPDKIEEANKGENTSLLHPVSLHGMDDPEREVLEWRFDSRLAFHHVAR
ELHPEYFKNC
```

B
```
AAVDLSHFL AIVALVVAI ALFLGFLGA ALQDSGLEV ALTEVIPLT
ALVEICTEM ALVEMGVEM AMSQVTNSA AVAEGTDRV AVLSIVNRV
ELADQLIHL EMMTACQGV FIMIVGGLV FLGAAGSTM FLQSRPEPT
FLREDLAFL GAASMTLTV HLEGKVILV IIAIVVWSI IIVQLNTSV
ILDLWIYHT ILGQLQPSL ILKEPVHGV ILVESPTVL IVFAVLSIV
IVGAETFYV IVTDSQYAL KALTEVIPL KLLRGTKAL KLLWKGEGA
KLTPLCVSL KLTSCNTSV KLVGKLNWA KQMAGDDCV LIVTRIVEL
LIWDDLRSL LLLIVTRIV LLNATAIAV LLQLTVWGI LLQYWSQEL
LLTQIGCTL LLWKGEGAV LQYLALAAL LTFGWCYKL LVITTYWGL
LVVAIIIAI MIVGGLVGL NITNWLWYI NTVATLYCV NVWATHACV
QILVESPTV QIWNHTTWM QLQARILAV QLTEAVQKI QMHEDIISL
RAMASDFNL RILAVERYL RILQQLLFI RIVFAVLSI RLRDLLLIV
RLVNGSLAL RMYSPTSIL RVIEVVQGA SINNETPGI SLAEEEVVI
SLLNATAIA SLQTGSEEL SLVKHHMYV SLWNWFNIT SLYNTVATL
TLNAWVKVV TLNFPISPI TLYCVHQRI TMGAASMTL TMLLGMLMI
VITQACPKV VIYQYMDDL VLAEAMSQV VTVYYGVPV VVAIIIAIV
WLWYIKLFI WQVMIVWQV WTLELLEEL YIEAEVIPA YIKLFIMIV
YLALQDSGL YLGRSAEPV YMDDLYVGS YQLEKEPIV YQYMDDLYV
YTAFTIPSI
```

Figure 5.1: **Epitope prediction. (A)** For each investigated protein from the HIV and human proteomes (the sequence of the *Nef* protein from HIV is shown), the location of HLA-A2 epitope nonamers (highlighted) was predicted using the software *NetCTL* (Section 5.4.1). In the Nef protein, three epitopes were identified. **(B)** On average, around 1% of all nonamers bind to HLA-A2 [95]. For the entire HXB2 proteome, which consists of 14 proteins and around 3000 amino acids, 91 nonamers are predicted to be HLA-A2 epitopes. Less than half of these (highlighted) are known to be detectable by CD8 T cells according to the LANL HIV database [165]. Our task in this chapter is to determine factors that distinguish the recognized from the non-recognized epitopes.

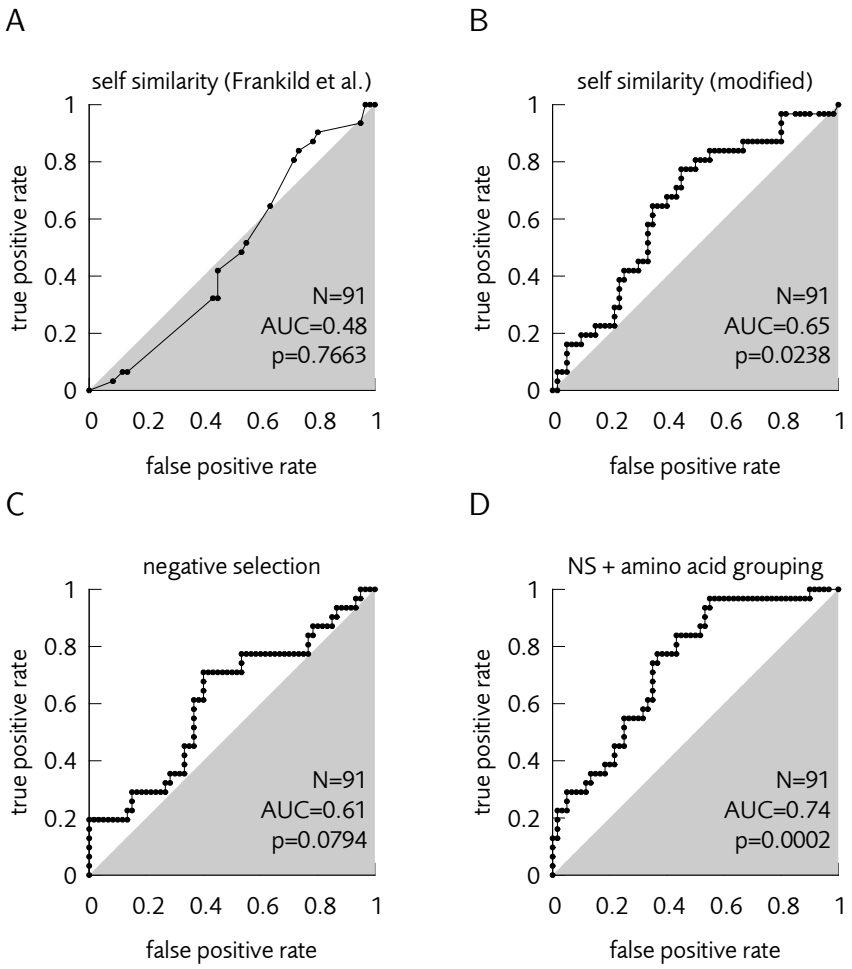


Figure 5.2: **Predicting T cell recognition status.** We used different metrics to predict which HLA-A2 epitopes can be recognized by CD8 T cells (i.e., our aim is to discern the emphasized strings in Figure 5.1B from the non-emphasized ones). This figure shows ROC curves for each predictor's performance. A diagonal ROC curve (gray triangles) would correspond to pure random guessing. The shown p-values are based on a statistical test of the null hypothesis that classification is purely random. **(A)** Prediction based on the method proposed by Frankild et al. [57] is not significantly different from random guessing. **(B)** With some biologically motivated improvements to the metric by Frankild et al. [57], one obtains a substantially improved prediction. **(C)** Using a negative selection (NS) algorithm, we can directly simulate combinatorial restriction of the T cell repertoire, and thereby predict a detection probability for each given 6-mer. However, the result is not significantly better than guessing. **(D)** Using amino acid grouping (Figure 5.3), the matching function employed by the NS algorithm can be made more realistic, and then it does give meaningful predictions.

residues in MHC-bound 9-mers carry the same amount of information: For HLA-A2 epitopes, the *anchor residues* at position 2 and 9 are bound to the "pockets" of the peptide binding groove of the MHC molecule. These positions in the 9-mer are restricted to only a few amino acids, and less likely to be in contact with the T cell receptor [101]. Thus, it is conceivable that the 6-mer from residue 3 to residue 8 is most important for determining TCR recognition. Although there is a small overlap at the 6-mer level between human and pathogenic proteomes [21], this overlap is almost non-existent if only HLA-A2 epitopes are considered rather than the entire proteome – in our case, only one of the nonself epitope 6-mers is also found in the set of self epitopes. Second, the way that Frankild et al. calculate peptide similarity leads to a heavy positional bias in sequences containing rarely substituting amino acids such as tryptophan. This positional bias can be removed by performing the similarity calculation in a simpler fashion, which weighs all positions equally (Section 5.4.3). Jointly, these modifications to the approach by Frankild et al. indeed lead to a significant[1] prediction (AUC$= 0.65, p = 0.0238$; Figure 5.2B).

5.2.2 The Negative Selection Algorithm

In contrast to the self similarity metric considered above, which directly compares the self and nonself epitope sequences, we will now investigate whether it is also possible to predict epitope recognition in an purely *mechanistic* fashion using a model of thymic negative selection, the so-called *negative selection algorithm* [55]. The negative selection algorithm is based on a string-based model of T cell cross-reactivity. While several such models are conceivable (Section 4.4.3), we limit our discussion here to an established model by Percus et al. [116], which is based on *r-contiguous* string matching.

5.2.3 The *r*-Contiguous Model of T Cell Cross-Reactivity

In the *r*-contiguous model, a TCR is represented as a pair of a string and a number. The string in this pair defines the amino acid sequences to which the receptor binds, and the number controls the degree of cross-reactivity: the lower the number, the more cross-reactive the TCR. In the simplest case, the number is equal to the length of the string, e.g.

(`ITTYWG`, 6)

would represent a TCR that binds *only* to the 6-mer `ITTYWG` (recall that since we consider HLA-A2 epitopes, we disregard the residues 1,2, and 9). More generally, a number k states that the TCR binds any epitope that is identical to the TCR string in at least k contiguous positions. For example, the receptor

(`ITTYWG`, 4)

[1] We use the significance threshold $p < 0.05$.

would bind to all epitopes of the form

$$\diamond\diamond\texttt{TYWG},\quad \diamond\texttt{TTYW}\diamond,\quad \text{or}\quad \texttt{ITTY}\diamond\diamond,$$

where $\diamond$ stands for an arbitrary amino acid.

Based on this matching function, we simulated thymic negative selection as follows[2]: We generated all possible TCR strings, and matched them against the self epitope strings using the matching parameter $k = 4$. Those receptors that matched a self epitope were deleted, and the remaining set of receptors formed the TCR repertoire. Now, we matched each nonself epitope to each TCR using parameter $k = 6$ (i.e., a perfect match), and ranked the nonself epitopes according to the number of matching TCRs. Ties were resolved by counting for each epitope the number of TCRs that match with parameter $k = 5$; remaining ties were then subsequently resolved by considering the parameter $k = 4$; and so forth until $k = 1$. The resulting rank of each epitope can be considered as a distance from self: the larger an epitope's rank, the stronger a TCR can bind to the epitope, and the more TCRs are able to bind to it[3].

A ROC analysis of this prediction method on the LANL HIV data resulted in $AUC = 0.61$ at $p = 0.0794$ (Figure 5.2C). Hence, although the result is slightly better than for the original Frankild et al. metric, the difference is not significant.

5.2.4 Amino Acid Grouping

We reasoned that a main problem with the r-contiguous model of TCR cross-reactivity is the fact that it does not take amino acid similarity into account. For example, it is conceivable that a 6-mer differing from another one only by substitution of a leucin (L) for an isoleucin (I) could still be recognized by the same TCR. One possibility to make the matching more realistic would be to use a scoring matrix, as does the Frankild et al. metric. However, the resulting negative selection algorithm is computationally very expensive[4]. A simpler method is to identify some pairs of amino acids that are roughly equivalent, and can thus be treated as equal. Using information from the literature on physicochemical properties and substitution likelihoods (Figure 5.3), we identified the following amino acid groups for this purpose: (D,E); (R,K); (T,S); (A,G); and (I,L,V). Hence, an alanine (A) and a glycine (G) at the same position of a 6-mer were considered identical, as were an arginine (R) and a lysine (K) and so forth. Indeed, this modification of the negative selection algorithm lead to a substantially improved predictive performance (Figure 5.2D) with an AUC of 0.74 ($p = 0.0002$).

[2] Note that not all these steps were in fact explicitly performed; this would have been computationally very demanding due to the large number of string comparisons required for generating the repertoire alone ($\approx 10^{12}$). However, some data compression tricks from computer science can be used to speed this process up, as discussed in Chapter 4.

[3] Formally, we ranked the epitope strings by lexicographically comparing their matching profiles $\Pi_{4\text{-CONT}}$ (Section 4.5.6).

[4] In the language of Chapter 4: The resulting pattern class would be a generalization of r-HAMMING, for which the consistency problem is NP-complete.

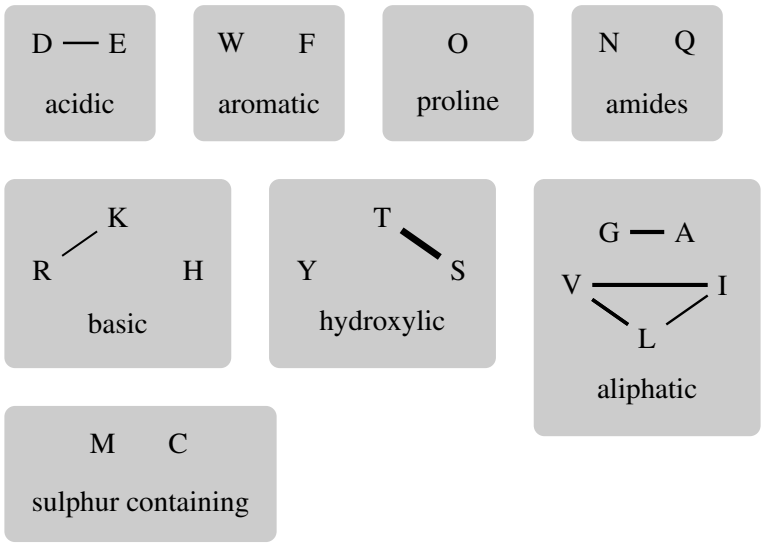


Figure 5.3: **Grouping the most similar amino acids.** To incorporate a notion of amino acid similarity into our negative selection algorithm, we aimed to identify the structurally and functionally most similar pairs of amino acids. To this end, we adopted a textbook categorization of the 20 amino acids into eight different groups according to their physicochemical properties [76]. Within each group, we identified pairs of amino acids that easily substitute according to data by Bordo and Argos [16]. Lines are drawn between each pair of acids that Bordo and Argos found to occur significantly more often in protein sequence alignments than it would be expected for random substitutions, and the thickness of the lines is proportional to the number of occurrences of each pair. All pairs linked in this fashion also have positive substitution scores in the BLOSUM matrices [57], which are based on similar data, but do not contain information on statistical significance.

5.2.5 Predicting Recognition Frequency

Because the immune response is a highly stochastic process, not all epitopes that can in principle be recognized by CD8 T cells will in fact be recognized in every infected individual. In a comprehensive *in vivo* study, Frahm et al. [56] tested CD8 T cell recognition of overlapping peptides (OLPs) from the 2001 HIV-B consensus sequence in 150 HIV-positive individuals of various ethnicities. The 410 tested OLPs varied in length from 15 to 20 amino acids and spanned the entire HIV-B proteome. The results of their study are freely available for download from the LANL HIV Immunology Database [165]. Their data associates each OLP with the percentage of patients in which the OLP elicited a detectable response. Frahm et al. [56] identified three factors that significantly correlated with recognition frequency: sequence entropy ($\rho = -0.25$), the fraction of amino acids that are rarely seen at the C termini of CTL epitopes (G, P, E, D, Q, N, T, S, and C; $\rho = -0.21$), and the proteasomal cleavage score predicted by the tool *NetChop* ($\rho = 0.18$; NetChop is also a part of NetCTL). However, note that the latter two factors are actually closely related. We were interested in whether our metrics would also be able to partly explain recognition frequency, and how the results would compare to those of Frahm et al.

To this end, we used NetCTL to generate a set of nonself epitopes from the 2001 HIV-B consensus sequence used by Frahm et al. (Section 5.4.2), which is slightly different to the version that we downloaded from UniProt. We then assigned to each epitope the average recognition frequency of the OLPs in which it occurs, and calculated our metrics as described above. This time we evaluated performance non-parametrically using the Spearman rank correlation coefficient ρ, as Frahm et al. did in their study; a ROC analysis was not appropriate as the prediction in this case is not verified against a binary value (i.e., recognition vs. non-recognition), but needs to be correlated with an ordinal value (i.e., the percentage of detection). The results are shown in Figure 5.4. Again, the original metric by Frankild provided no meaningful prediction ($\rho = -0.01, p = 0.9$), but the modified version did ($\rho = -0.46, p < 10^{-5}$). Notably, the latter correlation was significantly stronger ($p = 0.04$) than the correlation reported by Frahm et al. [56] for their entropy measure ($\rho = -0.25$), which to our knowledge had been the strongest correlation ever reported in the literature. Both versions of the negative selection algorithm also yielded significant correlations of $\rho = 0.21$ ($p = 0.045$) and $\rho = 0.3$ ($p = 0.004$), respectively, with the stronger correlation again provided by amino acid grouping.

It is important to note that these results are not simply a different statement of those in the previous section. Most importantly, in the previous section we did not distinguish at all between different recognition strengths. Moreover, only 14% of the predicted epitopes for the Frahm et al. dataset are found in OLPs that elicit no response at all, which is a much lower percentage than that of non-confirmed epitopes in the previous section (66%). This may be for two reasons: (1) Some of the non-confirmed epitopes may actually turn out to be epitopes in the

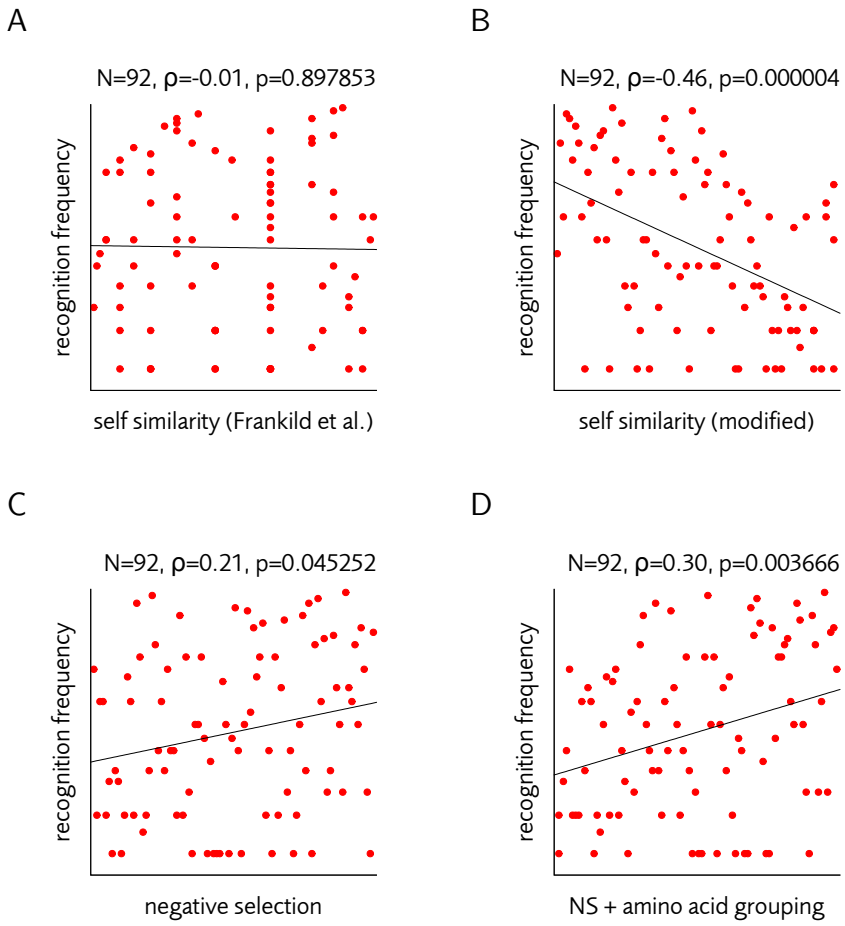


Figure 5.4: **Predicting epitope recognition frequency.** We investigated the correlation between the recognition frequency of HIV peptides in a study on 150 patients by Frahm et al. [56] and the two prediction measures defined in the text. Correlation was determined non-parametrically as the Spearman coefficient ρ, which measures linear correlation between *ranks* rather than absolute values. To enable comparison, the plots thus show rank-transformed measurements. In such a plot, a perfect correlation corresponds to a diagonal line with slope 1. **(A)** The self similarity metric in the version of Frankild et al. [57] yields no meaningful prediction. **(B)** The simplified version of the Frankild measure is inversely correlated with recognition status. The correlation in (C) is significantly stronger ($p = 0.04$) than the best predicting factor reported by Frahm et al. for their complete OLP dataset ($N = 401$), which was sequence entropy ($\rho = -0.25$). **(C)** The negative selection algorithm without amino acid grouping gives a moderate, but significant correlation. **(D)** Again, amino acid grouping improves the results of the negative selection algorithm.

future; and (2) the recognition frequency of the OLPs is aggregated information for the MHC supertypes in all patients rather than only HLA-A2. Finally, the virus sequence used by Frahm et al. is slightly different from the consensus sequence we downloaded from UniProt (Section 5.4.2), and thus 33 predicted 9-mer epitopes are not found in the nonself epitope dataset used in the previous section.

5.3 Discussion

Our results improve upon the previous work by Rolland et al. [124] and Frankild et al. [57], and now provide, to our knowledge for the first time, mechanistic predictions of T cell recognition that are significantly correlated with experimental findings. Importantly, our results are not the product of sophisticated machine learning algorithms or of careful parameter tuning, but are all grounded in relatively simple, reasonable biological hypotheses and assumptions.

Both the improved version of the Frankild et al. model and the negative selection algorithm assume some degree of TCR cross-reactivity, albeit in different manners (amino acid similarity vs. inexact matching). The fact that these two metrics give significant predictions, whereas the purely string-based calculations by Rolland et al. [124] do not, appears to confirm that this is indeed an essential property. This finding has some implications for interpreting previous studies that investigated self-nonself overlap at the proteome level. For example, Burroughs et al. [21] stated that the average overlap between the human and various pathogen proteomes is less than 0.2% at the 9-mer level. This most likely holds only at the string level, and due to TCR cross reactivity, the effective overlap should be much larger.

A key insight obtained from our findings is that the effects of combinatorial T cell repertoire shaping by thymic negative selection correlate not only with epitope recognition *status*, but even with recognition *frequency*. This is a remarkable result because, as mentioned previously, the capability to be detected by a TCR is a necessary, but not a sufficient condition for a peptide to elicit an immune response. One would expect biochemical factors, affinity-driven competition between T cell clones, and host-virus co-evolution (especially in HIV) to blur out the effects of thymic negative selection. Still, our negative selection based predictions are significant while a correlation between T cell recognition and *any* biochemical epitope property could, thus far, not be demonstrated. It is conceivable that a combination of biochemical epitope properties and repertoire shaping models could in the future be used to unravel the extent to which biochemical properties determine recognition status.

So far, our findings hold only for a single virus (the one for which most information is available) and only a single MHC supertype (the one that is most thoroughly researched). Obviously, it is highly desirable to generalize our results to more viruses and more epitope types, which could no longer be achieved within the scope of this thesis. Furthermore, our investigation on the predictive perfor-

mance of the negative selection algorithm only scratched the surface. Previous uses of similar algorithms have proved conclusive, most notably in studies by Košmrlj et al. [90, 91]; however, these studies used different matching functions than ours, which employ more sophisticated notions of amino acid similarity. Unfortunately, we have shown in the previous chapter that a generation of predictions from their similarity metric (which is a generalized version of the Hamming distance) is not straightforward. Further development of our algorithmic techniques would thus be important, because the predictive capacity of our approach could presumably benefit from a more fine-grained notion of amino acid similarity.

5.4 Methods

5.4.1 Epitope Prediction

Epitope prediction is the task of determining which chunks of a protein can be presented on an MHC class I molecule of a given supertype. The input to an epitope predictor is a set of protein sequences (e.g., given as a FASTA file), and its output is a set of substrings of fixed length from these proteins (k-mers). We used the software package *NetCTL*, version 1.2a, by the Technical University of Denmark [95, 96]. Because most MHC class I epitopes are 9-mers [165], NetCTL outputs only 9-mers as potential epitopes. Using artificial neural networks, NetCTL assigns a real-valued score to each 9-mer of the input protein. The NetCTL score combines predictions for proteasomal cleavage [87], TAP transport efficiency [117], and MHC binding [22, 112]. We used the default settings of NetCTL, which define all 9-mers with a score of at least 0.75 as epitopes. All predictions were performed for the HLA-A2 supertype of MHC class I, which is most common in humans in North America, Europe, and Northern Asia [56].

5.4.2 Datasets

Proteomes

The human proteome was downloaded from the Uniprot database [78, 30] on the 21st of June, 2011. Only proteins labeled as "reviewed" were used, which was the case for 20229 out of 51661 proteins. The epitope prediction method explained above identified 137499 9-mers from these proteins as potential HLA-A2 self epitopes. The nine proteins (counting Gag as one polyprotein) of the human immunodeficiency virus type 1 group M subtype B (isolate HXB2) were downloaded from the UniProt database on the same day. The HXB2 sequence commonly serves as the reference sequence for HIV [165].

Los Alamos HIV Dataset

The Los Alamos HIV Database list of HIV epitopes was accessed on the 21st of June, 2011, at which time the database contained 1305 epitopes with evidence for

T cell recognition. Out of these, 175 were human HLA-A2 epitopes.

Frahm et al. HIV Dataset

In a comprehensive *in vivo* study, Frahm et al. [56] tested CD8 T cell recognition of overlapping peptides (OLPs) from the 2001 HIV-B consensus sequence in 150 HIV-positive individuals of various ethnicities. The 410 tested OLPs spanned the entire HIV-B genome and were between 15 and 20 amino acids long. We downloaded their results from the LANL HIV Immunology Database [165]. To locate HLA-A2 epitopes within the OLPs, we applied NetCTL to their HIV-B sequence as described above. In this fashion, 92 nonamers were identified as epitopes, and each of those 9-mers was assigned the average recognition score of all OLPs (either 1 or 2) in which it occurred.

5.4.3 Self Similarity Measure by Frankild et al.

Frankild et al. [57] defined their similarity measure $\sigma_{\text{self}}(S,x)$ between a set of self epitopes S and a nonself epitope x as follows. For $a,b \in \Sigma$, where Σ is the alphabet of amino acids, let $W(a,b)$ be a function denoting the similarity of a and b (they used the BLOSUM 35 matrix). An unnormalized similarity score $\sigma(x,y)$ between two sequences $x,y \in \Sigma^L$ is given by

$$\sigma(x,y) = \sum_{i=1}^{N} W(x[i],y[i]) \; . \tag{5.1}$$

Frankild et al. defined a normalized version of σ as

$$\hat{\sigma}(x,y) = \frac{\sigma(x,y) - \sigma_{\min}(x)}{\sigma_{\max}(x) - \sigma_{\min}(x)} \tag{5.2}$$

where

$$\sigma_{\max}(x) = \max_{\bar{x} \in \Sigma^L} \sigma(x,\bar{x}) = \sigma(x,x)$$

and

$$\sigma_{\min}(x) = \min_{\bar{x} \in \Sigma^L} \sigma(x,\bar{x}) \; .$$

Finally, the self similarity $\sigma_{\text{self}}(S,x)$ is defined as

$$\sigma_{\text{self}}(S,x) = \max_{y \in S} \hat{\sigma}(x,y) \; . \tag{5.3}$$

This definition ensures that $0 \leq \sigma_{\text{self}}(S,x) \leq 1$. However, the normalization induces several problems. First, $\hat{\sigma}$ is not symmetric. Second, $\sigma_{\max}(x)$ and $\sigma_{\min}(x)$ may vary greatly with x, and especially the appearance of rarely substituting amino acids in x (where $W(x,x)$ is very large and $W(x,y)$ is very small for $y \neq x$) substantially weakens the contribution of the rest of the sequence to the score, inducing positional bias.

To remove these issues, we substituted $\hat{\sigma}$ for the simpler σ in Equation 5.3. Although we did not really see a need to normalize the resulting values to the interval $[0,1]$, this could be done in a non-biasing fashion using

$$\hat{\sigma}(x,y) = \frac{\sigma(x,y) - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}}$$

where

$$\sigma_{\max} = \max_{x,y \in \Sigma^L} \sigma(x,y)$$

and

$$\sigma_{\min} = \min_{x,y \in \Sigma^L} \sigma(x,y) \ .$$

Bibliography

[1] U. Aickelin. Special issue on artificial immune systems: editorial. *Evolutionary Intelligence*, 1(2):83–84, 2008. doi:10.1007/s12065-008-0007-7.

[2] M. S. Anderson, E. S. Venanzi, L. Klein, Z. Chen, S. P. Berzins, S. J. Turley, H. von Boehmer, R. Bronson, A. Dierich, C. Benoist, and D. Mathis. Projection of an immunological self shadow within the thymus by the AIRE protein. *Science*, 298:1395–1401, 2002. doi:10.1126/science.1075958.

[3] M. Anthony and N. Biggs. *Computational Learning Theory*, volume 30 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1992.

[4] E. Assarsson, J. Sidney, C. Oseroff, V. Pasquetto, H. Bui, N. Frahm, C. Brander, B. Peters, H. Grey, and A. Sette. A quantitative analysis of the variables affecting the repertoire of T cell specificities recognized after vaccinia virus infection. *Journal of Immunology*, 178(12):7890–901, 2007. URL: http://www.jimmunol.org/content/178/12/7890.long.

[5] M. Bajénoff, J. G. Egen, L. Y. Koo, J. P. Laugier, F. Brau, N. Glaichenhaus, and R. N. Germain. Stromal cell networks regulate lymphocyte entry, migration, and territoriality in lymph nodes. *Immunity*, 25:989–1001, 2006. doi:10.1016/j.immuni.2006.10.011.

[6] M. Bajénoff, J. G. Egen, H. Qi, A. Y. Huang, F. Castellino, and R. N. Germain. Highways, byways and breadcrumbs: directing lymphocyte traffic in the lymph node. *Trends in Immunology*, 28(8):346–352, 2007. doi:10.1016/j.it.2007.06.005.

[7] J. Balthrop, F. Esponda, S. Forrest, and M. R. Glickman. Coverage and generalization in an artificial immune system. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 1045–1050, 2002. URL: http://portal.acm.org/citation.cfm?id=646205.682310.

[8] J. Balthrop, S. Forrest, and M. R. Glickman. Revisiting LISYS: Parameters and normal behavior. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1045–1050, 2002. doi:10.1109/CEC.2002.1004387.

[9] C. Beauchemin, N. M. Dixit, and A. S. Perelson. Characterizing T cell movement within lymph nodes in the absence of antigen. *Journal of Immunology*, 178(9):5505–5512, 2007.

[10] J. B. Beltman, A. F. M. Marée, and R. J. de Boer. Analysing immune cell migration. *Nature Reviews Immunology*, 9:789–798, 2009.

[11] J. B. Beltman, A. F. M. Marée, J. N. Lynch, M. J. Miller, and R. J. de Boer. Lymph node topology dictates T cell migration behavior. *Journal of Experimental Medicine*, 204:771–780, 2007. doi:10.1084/jem.20061278.

[12] C. M. Bishop. Novelty detection and neural network validation. *IEEE Proceedings on Vision and Image Signal Processing*, 141:217–222, 1994. doi:10.1049/ip-vis:19941330.

[13] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[14] V. Blaschke, B. Micheel, R. Pabst, and J. Westermann. Lymphocyte traffic through lymph nodes and peyer's patches of the rat: B- and T-cell-specific migration patterns within the tissue, and their dependence on splenic tissue. *Cell and Tissue Research*, 282:377–386, 1995.

[15] J. N. Blattman, R. Antia, D. J. Sourdive, X. Wang, S. M. Kaech, K. Murali-Krishna, J. D. Altman, and R. Ahmed. Estimating the precursor frequency of naive antigen-specific CD8 T cells. *Journal of Experimental Medicine*, 195(5):657–664, 2002. doi:doi:10.1084/jem.20001021.

[16] D. Bordo and P. Argos. Suggestions for "safe" residute substitutions in site-directed mutagenesis. *Journal of Molecular Biology*, 217:721–729, 1991.

[17] P. Bousso. T-cell activation by dendritic cells in the lymph node: lessons from the movies. *Nature Reviews Immunology*, 8:675–684, 2008. doi:10.1038/nri2379.

[18] P. Bousso, N. R. Bhakta, R. S. Lewis, and E. Robey. Dynamics of thymocyte-stromal cell interactions visualized by two-photon microscopy. *Science*, 296:1876–1880, 2002. doi:10.1126/science.1070945.

[19] R. Bubley, M. Dyer, C. Greenhill, and M. Jerrum. On approximately counting colourings of small degree graphs. *SIAM Journal on Computing*, 29:387–400, 1998. doi:10.1137/S0097539798338175.

[20] R. Bujdoso, P. Young, J. Hopkins, D. Allen, and I. McConnell. Non-random migration of CD4 and CD8 T cells: changes in the CD4:CD8 ratio and interleukin 2 responsiveness of efferent lymph cells following in vivo antigen challenge. *European Journal of Immunology*, 19:1779–1784, 1989.

[21] N. J. Burroughs, R. J. de Boer, and C. Keşmir. Discriminating self from nonself with short peptides from large proteomes. *Immunogenetics*, 56:311–320, 2004. doi:10.1007/s00251-004-0691-0.

[22] S. Buus, S. Lauemoller, P. Worning, C. Keşmir, T. Friumurer, S. Corbet, A. Fomsgaard, J. Hilden, A. Holm, and S. Brunak. Sensitive quantitative predictions of peptide-MHC binding by a "query by committee" artificial neural network approach. *Tissue Antigens*, 62:378–384, 2003. doi:10.1034/j.1399-0039.2003.00112.x.

[23] M. D. Cahalan and I. Parker. Choreography of cell motility and interaction dynamics imaged by two-photon microscopy in lymphoid organs. *Annual Reviews in Immunology*, 26:585–626, 2008.

[24] R. N. P. Cahill, H. Frost, and Z. Trnka. The effects of antigen on the migration of recirculating lymphocytes through single lymph nodes. *Journal of Experimental Medicine*, 143:870–888, 1976.

[25] F. Castellino, A. Y. Huang, G. Altan-Bonnet, S. Stoll, C. Scheinecker, and R. N. Germain. Chemokines enhance immunity by guiding naive CD8+ T cells to sites of CD4+ T cell–dendritic cell interaction. *Nature*, 440:890–895, 2006.

[26] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009. doi:10.1145/1541880.1541882.

[27] D. L. Chao, M. P. Davenport, S. Forrest, and A. S. Perelson. A stochastic model of cytotoxic T cell responses. *Journal of Theoretical Biology*, 228:227–240, 2004. doi:10.1016/j.jtbi.2003.12.011.

[28] D. L. Chao, M. P. Davenport, S. Forrest, and A. S. Perelson. The effects of thymic selection on the range of T cell cross-reactivity. *European Journal of Immunology*, 35:3452–3459, 2005. doi:10.1002/eji.200535098.

[29] E. L. Charnov. Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9:129–136, 1976.

[30] T. U. Consortium. Ongoing and future developments at the universal protein resource. *Nucleic Acids Research*, 39:D214–D219, 2011. doi:10.1093/nar/gkq1020.

[31] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[32] S. Cose. T-cell migration: a naive paradigm? *Immunology*, 120:1–7, 2007.

[33] M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on Strings*. Cambridge University Press, Cambridge, 1 edition, 2007.

[34] D. Crowther and J. Wagstaff. Lymphocyte migration in malignant disease. *Clinical and Experimental Immunology*, 51:413–420, 1983.

[35] D. Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, 1999.

[36] M. M. Davis and P. J. Bjorkman. T-cell antigen receptor genes and T-cell recognition. *Nature*, 334:395–402, 1988. doi:10.1038/334395a0.

[37] V. Detours and A. S. Perelson. Explaining high alloreactivity as a quantitative consequence of affinity-driven thymocyte selection. *Proceedings of the National Academy of Sciences of the United States of America*, 96:5153–5158, 1999. doi:10.1073/pnas.96.9.5153.

[38] P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: Algorithms, analysis, and implications. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 110–119. IEEE Computer Society, 1996.

[39] A. Di Sabatino, R. Carsetti, and G. R. Corazza. Post-splenectomy and hyposplenic states. *Lancet*, 2011. doi:10.1016/S0140-6736(10)61493-6.

[40] T. Dunning. Statistical identification of language. Technical Report CRL MCCS-94-273, New Mexico State University, 1994. URL: http://www.comp.lancs.ac.uk/computing/users/paul/ucrel/papers/lingdet.ps.

[41] S. Efroni, D. Harel, and I. R. Cohen. Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation. *Genome Research*, 13:2485–2497, 2003.

[42] M. Elberfeld and J. Textor. Efficient algorithms for string-based negative selection. In *Proceedings of the 8th International Conference on Artificial Immune Systems (ICARIS 2009)*, volume 5666 of *Lecture Notes in Computer Science*, pages 109–121. Springer, 2009. doi:10.1007/978-3-642-03246-2_14.

[43] M. Elberfeld and J. Textor. Negative selection algorithms on strings with efficient training and linear-time classification. *Theoretical Computer Science*, 412:534–542, 2011. doi:10.1016/j.tcs.2010.09.022.

[44] F. Esponda. *Negative Representations of Information*. PhD thesis, University of New Mexico, 2005. URL: http://cs.unm.edu/~forrest/projects/ndb/papers/fernando-dissertation.pdf.

[45] F. Esponda, S. Forrest, and P. Helman. The crossover closure and partial match detection. In *Proceedings of the 3rd International Conference on Artificial Immune Systems (ICARIS 2003)*, volume 2787 of *Lecture Notes in Computer Science*, pages 249–260. Springer, 2003. doi:10.1007/978-3-540-45192-1_24.

[46] L. D. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[47] K. A. Fichthorn and W. H. Weinberg. Theoretical foundations of dynamical monte carlo simulations. *The Journal of Chemical Physics*, 95(2):1090–1096, 1991. doi:10.1063/1.461138.

[48] M. T. Figge et al. Deriving a germinal center lymphocyte migration model from two-photon data. *Journal of Experimental Medicine*, 205(13):3019–3029, 2008.

[49] W. L. Ford. The immunological and migratory properties of lymphocytes recirculating through the rat spleen. *British Journal of Experimental Pathology*, 50:257–269, 1969.

[50] W. L. Ford. The kinetics of lymphocyte recirculation within the rat spleen. *Cell & Tissue Kinetics*, 2(3):171–191, 1969. doi:10.1111/j.1365-2184.1969.tb00229.x.

[51] W. L. Ford. The recirculating lymphocyte pool of the rat: a systematic description of the migratory behaviour of recirculating lymphocytes. *Immunology*, 49:83–94, 1983. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1454101/.

[52] S. Forrest and C. Beauchemin. Computer immunology. *Immunological Reviews*, 216:176–197, 2007.

[53] S. Forrest, S. A. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40:88–96, 1997. doi:10.1145/262793.262811.

[54] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 120–128, Washington, DC, USA, 1996. IEEE Computer Society. doi:10.1109/SECPRI.1996.502675.

[55] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 202–212. IEEE Computer Society Press, 1994. doi:10.1109/RISP.1994.296580.

[56] N. Frahm, B. T. Korber, C. M. Adams, J. J. Szinger, R. Draenert, M. M. Addo, M. E. Feeney, K. Yusim, K. Sango, N. V. Brown, D. SenGupta, A. Piechocka-Trocha, T. Simonis, F. M. Marincola, A. G. Wurcel, D. R. Stone, C. J. Russell, P. Adolf, D. Cohen, T. Roach, A. StJohn, A. Khatri, K. Davis, J. Mullins, P. J. Goulder, B. D. Walker, and C. Brander. Consistent cytotoxic-T-lymphocyte targeting of immunodominant regions in human immunodeficiency virus across multiple ethnicities. *Journal of Virology*, 78:2187–2200, 2004. doi:10.1128/JVI.78.5.2187-2200.2004.

[57] S. Frankild, R. J. de Boer, O. Lund, M. Nielsen, and C. Keşmir. Amino acid similarity accounts for T cell cross-reactivity and for "holes" in the T cell repertoire. *PLoS one*, 3(3):e1831, 2008. doi:10.1371/journal.pone.0001831.

[58] V. V. Ganusov and R. J. De Boer. Do most lymphocytes in humans really reside in the gut? *Trends in Immunology*, 28:514–518, 2007. doi:10.1016/j.it.2007.08.009.

[59] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.

[60] M. L. Glasser and I. J. Zucker. Extended watson integrals for the cubic lattices. *Proceedings of the National Academy of Sciences of the USA*, 74:1800–1801, 1977.

[61] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[62] C. Halin, M. L. Scimone, R. Bonasio, J. M. Gauguet, T. R. Mempel, E. Quackenbush, R. L. Proia, S. Mandala, and U. H. von Andrian. The S1P-analog FTY720 differentially modulates T-cell homing via HEV: T-cell-expressed S1P1 amplifies integrin activation in peripheral lymph nodes but not in Peyer patches. *Blood*, 106:1314–1322, 2005.

[63] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont. An artificial immune system architecture for computer security applications. *IEEE Transactions on Evolutionary Computation*, 6:252–280, 2002. doi:10.1109/TEVC.2002.1011540.

[64] D. Heckerman, C. Kadie, and J. Listgarten. Leveraging information across HLA alleles/supertypes improves HLA-specific epitope prediction. *Journal of Computational Biology*, 14(6):736–746, 2007. doi:10.1089/cmb.2007.R013.

[65] S. E. Henrickson, T. R. Mempel, I. B. Mazo, B. Liu, M. N. Artyomov, H. Zheng, A. Peixoto, M. P. Flynn, B. Senman, T. Junt, H. C. Wong, A. K. Chakraborty, and U. H. von Andrian. T cell sensing of antigen dose governs interactive behavior with dendritic cells and sets a threshold for T cell activation. *Nature Immunology*, 9:282–291, 2008.

[66] M. Hilker and K. Luther. Artificial cell communication in distributed systems. In *AINA '08*, pages 1034–1041, Washington, DC, USA, 2008. IEEE Computer Society. doi:10.1109/AINA.2008.12.

[67] H. Hirsh, N. Mishra, and L. Pitt. Version spaces and the consistency problem. *Artificial Intelligence*, 156:115–138, 2004. doi:10.1016/j.artint.2003.04.003.

[68] S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 7(1):1289–1296, 2000.

[69] S. A. Hofmeyr. *An Immunological Model of Distributed Detection and Its Application to Computer Security*. PhD thesis, University of New Mexico, 1999. URL: http://www.cs.unm.edu/~steveah/steve_diss.ps.gz.

[70] S. A. Hofmeyr. An interpretative introduction to the immune system. In L. A. Sagel and I. R. Cohen, editors, *Design Principles for the Immune System and Other Distributed Autonomous Systems*, pages 3–26. Oxford University Press, Oxford, 2000.

[71] S. A. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 7(1):1289–1296, 2000. URL: http://cs.unm.edu/~forrest/publications/hofmeyr_forrest.pdf.

[72] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151–180, 1998.

[73] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.

[74] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*, pages 313–329. Academic Press, Orlando, FL, USA, 1978.

[75] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[76] F. Horn, G. Lindenmeier, I. Moc, C. Grillhösl, S. Berghold, N. Schneider, and B. Münster. *Biochemie des Menschen*. Thieme, Stuttgart, 2002.

[77] M. Huber. Exact sampling and approximate counting techniques. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 31–40, New York, NY, USA, 1998. ACM. doi:10.1145/276698.276709.

[78] E. Jain, A. Bairoch, S. Duvaud, I. Phan, N. Redaschi, B. E. Suzek, M. J. Martin, P. McGarvey, and E. Gasteiger. Infrastructure for the life sciences: design and implementation of the UniProt website. *BMC Bioinformatics*, 10:136, 2009. doi:10.1186/1471-2105-10-136.

[79] C. Janeway, P. Travers, M. Walport, and M. Shlomchick. *Immunobiology*. Garland Science, 2005.

[80] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X.

[81] Z. Ji. *Negative Selection Algorithms: from the Thymus to V-detector*. PhD thesis, University of Memphis, 2006. URL: http://www.zhouji.net/prof/380318.pdf.

[82] Z. Ji and D. Dasgupta. Revisiting negative selection algorithms. *Evolutionary Computation*, 15(2):223–251, 2007. doi:10.1162/evco.2007.15.2.223.

[83] Z. Ji and D. Dasgupta. V-detector: An efficient negative selection algorithm with "probably adequate" detector coverage. *Inf. Sci.*, 179:1390–1406, 2009. doi:10.1016/j.ins.2008.12.015.

[84] Y. Kawashima, M. Sugimura, Y.-C. Hwang, and N. Kudo. The lymph system in mice. *Japanese Journal of Veterinary Research*, 12(4):69–78, 1964.

[85] M. Kearns, M. Li, and L. Valiant. Learning boolean formulas. *Journal of the ACM*, 41:1298–1328, November 1994. doi:10.1145/195613.195656.

[86] J. O. Kephart. A biologically inspired immune system for computers. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139. MIT Press, 1994.

[87] C. Keşmir, A. K. Nussbaum, H. Schild, V. Detours, and Brunak. Prediction of proteasome cleavage motifs by neural networks. *Protein Engineering*, 15(4), 2002.

[88] J. Kim and P. J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1330–1337. Morgan Kaufmann, 2001.

[89] D. E. Knuth, J. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977. doi:10.1137/0206024.

[90] A. Košmrlj, A. K. Jha, E. S. Huseby, M. Kardar, and A. K. Chakraborty. How the thymus designs antigen-specific and self-tolerant T cell receptor sequences. *Proceedings of the National Academy of Sciences of the USA*, 105(43):16671–16676, 2008. doi:10.1073/pnas.0808081105.

[91] A. Košmrlj, E. L. Read, Y. Qi, T. M. Allen, M. Altfeld, S. G. Deeks, F. Pereyra, M. Carrington, B. D. Walker, and A. K. Chakraborty. Effects of thymic selection of the T-cell repertoire on HLA class I–associated control of HIV infection. *Nature*, 465:350–354, 2010. doi:10.1038/nature08997.

[92] Y. Kumamoto, L. M. Mattei, S. Sellers, G. W. Payne, and A. Iwasaki. CD4+ T cells support cytotoxic T lymphocyte priming by controlling lymph node input. *Proceedings of the National Academy of Sciences of the USA*, 108:8749–8754, 2011. doi:10.1073/pnas.1100567108.

[93] J. K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Information and Computation*, 185:41–55, 2003. doi:10.1016/S0890-5401(03)00057-9.

[94] E. S. Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001. doi:10.1038/35057062.

[95] M. V. Larsen, C. Lundegaard, K. Lamberth, S. Buus, S. Brunak, O. Lund, and M. Nielsen. An integrative approach to CTL epitope prediction. a combined algorithm integrating MHC-I binding, TAP transport efficiency, and proteasomal cleavage predictions. *European Journal of Immunology*, 35(8):2295–2303, 2005. doi:10.1002/eji.200425811.

[96] M. V. Larsen, C. Lundegaard, K. Lamberth, S. Buus, O. Lund, and M. Nielsen. Large-scale validation of methods for cytotoxic T-lymphocyte epitope prediction. *BMC Bioinformatics*, 8:424, 2007. doi:10.1186/1471-2105-8-424.

[97] A. J. S. Leger, B. Peters, J. Sidney, A. Sette, and R. L. Hendricks. Defining the herpes simplex virus-specific CD8+ T cell repertoire in C57BL/6 mice. *Journal of Immunology*, 186:3927–3933, 2011. doi:10.4049/jimmunol.1003735.

[98] J. J. Linderman et al. Characterizing the dynamics of CD4+ T cell priming within a lymph node. *J Immunol*, 184(6):2873–2885, 2010.

[99] M. Liśkiewicz and J. Textor. Negative selection algorithms without generating detectors. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'10)*, pages 1047–1054. ACM, 2010. doi:10.1145/1830483.1830673.

[100] M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of las vegas algorithms. *Information Processing Letters*, 47:173–180, 1993. doi:10.1016/0020-0190(93)90029-9.

[101] O. Lund, C. Keşmir, M. Nielsen, C. Lundegaard, and S. Brunak. *Immunological Bioinformatics*. MIT Press, Cambridge, MA, USA, 2005.

[102] C. R. Mackay. Chemokines: Immunology's high impact factors. *Nature Immunology*, 2:95–101, 2001. doi:10.1038/84298.

[103] C. R. Mackay, W. Marston, and L. Dudler. Altered patterns of T cell migration through lymph nodes and skin following antigen challenge. *European Journal of Immunology*, 22:2205–2210, 1992.

[104] H. Melville. *Moby-Dick, or, The Whale*. Hendricks House, New York, 1952.

[105] T. R. Mempel, S. E. Henrickson, and U. H. von Andrian. T-cell priming by dendritic cells in lymph nodes occurs in three distinct phases. *Nature*, 427:154–159, 2004.

[106] M. J. Miller, S. H. Wei, I. Parker, and M. D. Cahalan. Two-photon imaging of lymphocyte motility and antigen response in intact lymph node. *Science*, 296:1869–1873, 2002. doi:10.1126/science.1070051.

[107] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982. doi:10.1016/0004-3702(82)90040-6.

[108] J. J. Moon, H. H. Chu, M. Pepper, S. J. McSorley, S. C. Jameson, R. M. Kedl, and M. K. Jenkins. Naive CD4+ T cell frequency varies for different epitopes and predicts repertoire diversity and response magnitude. *Immunity*, 27:203–213, 2007. doi:10.1016/j.immuni.2007.07.007.

[109] S. N. Mueller and R. N. Germain. Stromal cell contributions to the homeostasis and functionality of the immune system. *Nature Reviews Immunology*, 9(9):618–629, 2009. doi:10.1038/nri2588.

[110] J. D. Murray. *Mathematical Biology: I. An Introduction – 3rd Edition*. Springer, 2002.

[111] M. Nielsen, C. Lundegaard, O. Lund, and C. Keşmir. The role of the proteasome in generating cytotoxic T-cell epitopes: insights obtained from improved predictions of proteasomal cleavage. *Immunogenetics*, 57:33–41, 2005.

[112] M. Nielsen, C. Lundegaard, P. Worning, S. L. Lauem?ller, K. Lamberth, S. Buus, S. Brunak, and O. Lund. Reliable prediction of T-cell epitopes using neural networks with novel sequence representations. *Protein Science*, 12:1007–1017, 2003. doi:10.1110/ps.0239403.

[113] J. J. Obar, K. M. Khanna, and L. Lefrancois. Endogenous naive CD8+ T cell precursor frequency regulates primary and memory responses to infection. *Immunity*, 28:859–869, 2008. doi:10.1016/j.immuni.2008.04.010.

[114] C. M. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[115] T. C. Pellas and L. Weiss. Deep splenic lymphatic vessels in the mouse: a route of splenic exit for recirculating lymphocytes. *American Journal of Anatomy*, 187:347–354, 1990.

[116] J. K. Percus, O. E. Percus, and A. S. Perelson. Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. *Proceedings of the National Academy of Sciences of the United States of America*, 90(5):1691–1695, 1993. URL: http://www.pnas.org/content/90/5/1691.

[117] B. Peters, S. Bulik, R. Tampe, P. M. Van Endert, and H. G. Holzhutter. Identifying MHC class I epitopes by predicting the TAP transport efficiency of epitope precursors. *Journal of Immunology*, 171:1741–1749, 2003.

[118] G. J. Pierce and J. G. Ollason. Eight reasons why optimal foraging theory is a complete waste of time. *Oikos*, 49(1):111–117, 1987. URL: http://www.vliz.be/imisdocs/publications/115603.pdf.

[119] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35:965–984, 1988. doi:10.1145/48014.63140.

[120] G. H. Rannie and K. J. Donald. Estimation of the migration of thoracic duct lymphocytes to non-lymphoid tissues. A comparison of the distribution of radioactivity at intervals following i. v. transfusion of cells labelled with 3H, 14C, 75Se, 99mTc, 125I, and 51Cr in the rat. *Cell & Tissue Kinetics*, 10:523–541, 1977.

[121] N. Rapin, O. Lund, M. Bernaschi, and F. Castiglione. Computational immunology meets bioinformatics: the use of prediction tools for molecular binding in the simulation of the immune system. *PLoS ONE*, 5:e9862, 2010. doi:10.1371/journal.pone.0009862.

[122] R. Reischuk and J. Textor. Stochastic search with locally clustered targets: Learning from T cells. In *Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*, volume 6825 of *Lecture Notes in Computer Science*, pages 146–159. Springer, 2011.

[123] T. Riggs, A. Walts, N. Perry, L. Bickle, J. N. Lynch, A. Myers, J. Flynn, J. J. Linderman, M. J. Miller, and D. E. Kirschner. A comparison of random vs. chemotaxis driven contacts of T cells with dendritic cells during repertoire scanning. *Journal of Theoretical Biology*, 250(4):732–751, 2008. doi:10.1016/j.jtbi.2007.10.015.

[124] M. Rolland, D. C. Nickle, W. Deng, N. Frahm, C. Brander, G. H. Learn, D. Heckerman, N. Jojic, V. Jojic, B. D. Walker, and J. I. Mullins. Recognition of HIV-1 peptides by host CTL is related to HIV-1 similarity to human proteins. *PLoS one*, 2(9):e828, 2007. doi:10.1371/journal.pone.0000823.

[125] K. Schepers, E. Swart, J. W. van Heijst, C. Gerlach, M. Castrucci, D. Sie, M. Heimerikx, A. Velds, R. M. Kerkhoven, R. Arens, and T. N. Schumacher. Dissecting T cell lineage relationships by cellular barcoding. *Journal of Experimental Medicine*, 205:2309–2318, 2008. doi:10.1084/jem.20072462.

[126] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001. doi:10.1162/089976601750264965.

[127] B. Schölkopf and A. J. Smola. *Learning With Kernels*. MIT Press, 2002.

[128] K. Shortman, M. Egerton, G. J. Spangrude, and R. Scollay. The generation and fate of thymocytes. *Seminars in Immunology*, 2:3–12, 1990.

[129] J. E. Slansky. Antigen-specific T cells: analyses of the needles in the haystack. *PLoS Biology*, 1:E78, 2003. doi:10.1371/journal.pbio.0000078.

[130] K. A. Soderberg, G. W. Payne, A. Sato, R. Medzhitov, S. S. Segal, and A. Iwasaki. Innate control of adaptive immunity via remodeling of lymph node feed arteriole. *Proceedings of the National Academy of Sciences of the USA*, 102(45):16315–16320, 2005. doi:10.1073/pnas.0506190102.

[131] A. Somayaji. *Operating System Stability and Security through Process Homeostasis*. PhD thesis, University of New Mexico, 2002. URL: http://people.scs.carleton.ca/~soma/pubs/soma-diss-2sided.pdf.

[132] W. Srikusalanukul, F. De Bruyne, and P. McCullagh. Modelling of peripheral lymphocyte migration: system identification approach. *Immunology and Cell Biology*, 78:288–293, 2000. doi:10.1046/j.1440-1711.2000.00907.x.

[133] D. J. Stekel. The simulation of density-dependent effects in the recirculation of T lymphocytes. *Scandinavian Journal of Immunology*, 47:426–430, 1998. doi:10.1046/j.1365-3083.1998.00329.x.

[134] D. J. Stekel, C. E. Parker, and M. A. Nowak. A model of lymphocyte recirculation. *Immunology Today*, 18(5):217–221, 1997. doi:10.1016/S0167-5699(97)01036-0.

[135] D. W. Stephens and J. R. Krebs. *Foraging Theory*. Princeton University Press, 1987.

[136] T. Stibor. *On the Appropriateness of Negative Selection for Anomaly Detection and Network Intrusion Detection*. PhD thesis, Technische Universität Darmstadt, 2006. URL: http://www.sec.in.tum.de/~stibor/papers/2006/dissertationstiborce4.pdf.

[137] T. Stibor. Discriminating self from non-self with finite mixtures of multivariate bernoulli distributions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 127–134. ACM Press, 2008. doi:10.1145/1389095.1389113.

[138] T. Stibor. An empirical study of self/non-self discrimination in binary data with a kernel estimator. In *Proceedings of the 7th International Conference on Artificial Immune Systems (ICARIS)*, pages 352–363. Springer, 2008. doi:10.1007/978-3-540-85072-4_31.

[139] T. Stibor. Foundations of r-contiguous matching in negative selection for anomaly detection. *Natural Computing*, 8:613–641, 2009. doi:10.1007/s11047-008-9097-5.

[140] T. Stibor, K. M. Bayarou, and C. Eckert. An investigation of r-chunk detector generation on higher alphabets. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, volume 3102 of *Lecture Notes in Computer Science*, pages 299–307. Springer, 2004. doi:10.1007/978-3-540-24854-5_31.

[141] T. Stibor, P. Mohr, J. Timmis, and C. Eckert. Is negative selection appropriate for anomaly detection? In *Proceedings of GECCO 2005*, pages 321–328. ACM, 2005. doi:10.1145/1068009.1068061.

[142] T. Stibor, J. Timmis, and C. Eckert. Generalization regions in hamming negative selection. In *Intelligent Information Systems (IIS-2006)*, volume 35

of *Advances in Soft Computing*, pages 447–456. Springer-Verlag, 2006. doi:10.1007/3-540-33521-8_49.

[143] T. Stibor, J. Timmis, and C. Eckert. The link between r-contiguous detectors and k-CNF satisfiability. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 491–498. IEEE Press, 2006. doi:10.1109/CEC.2006.1688350.

[144] A. T. Stock, C. M. Jones, W. R. Heath, and F. R. Carbone. Rapid recruitment and activation of CD8+ T cells after herpes simplex virus type 1 skin infection. *Immunology and Cell Biology*, 89:143–148, 2011. doi:10.1038/icb.2010.66.

[145] S. Stoll, J. Delon, T. M. Brotz, and R. N. Germain. Dynamic imaging of T cell-dendritic cell interactions in lymph nodes. *Science*, 296:1873–1876, 2002. doi:10.1126/science.1071065.

[146] C. Sumen, T. R. Mempel, I. B. Mazo, and U. H. von Andrian. Intravital microscopy: Visualizing immunity in context. *Immunity*, 21, 2004. doi:10.1016/j.immuni.2004.08.006.

[147] J. Textor, A. Peixoto, S. E. Henrickson, M. Sinn, U. H. von Andrian, and J. Westermann. Defining the quantitative limits of intravital two-photon lymphocyte tracking. *Proceedings of the National Academy of Sciences of the United States of America*, 108(30):12401–12406, 2011. doi:10.1073/pnas.1102288108.

[148] J. Textor and J. Westermann. Modeling migration, compartmentalization, and exit of naive T cells in lymph nodes without chemotaxis. In *Proceedings of the 6th International Conference on Artificial Immune Systems (ICARIS 2007)*, volume 4628 of *Lecture Notes in Computer Science*, pages 228–239. Springer, 2007. doi:10.1007/978-3-540-73922-7_20.

[149] J. Timmis, A. Hone, T. Stibor, and E. Clark. Theoretical advances in artificial immune systems. *Theoretical Computer Science*, 403:11–32, 2008. doi:10.1016/j.tcs.2008.02.011.

[150] F. Trepel. Number and distribution of lymphocytes in man. A critical analysis. *Klinische Wochenschrift*, 52:511–515, 1974. URL: http://www.springerlink.com/content/kj22334977mv3403/.

[151] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8:410–421, 1979. doi:10.1137/0208032.

[152] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

[153] J. W. van Heijst, C. Gerlach, E. Swart, D. Sie, C. Nunes-Alves, R. M. Kerkhoven, R. Arens, M. Correia-Neves, K. Schepers, and T. N. Schumacher. Recruitment of antigen-specific CD8+ T cells in response to infection is markedly efficient. *Science*, 325:1265–1269, 2009. doi:10.1126/science.1175455.

[154] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, Berlin, 2000.

[155] U. H. von Andrian. Intravital microscopy of the peripheral lymph node mirocirculation in mice. *Microcirculation*, 3:287–300, 1996.

[156] S. H. Wei, I. Parker, M. J. Miller, and M. D. Cahalan. A stochastic view of lymphocyte motility and trafficking within the lymph node. *Immunological Reviews*, 195:136–159, 2003. doi:10.1034/j.1600-065X.2003.00076.x.

[157] G. H. Weiss. Asymptotic form for random walk survival probabilities on three-dimensional lattices with traps. *Proceedings of the National Academy of Sciences of the USA*, 77(8):4391–4392, 1980. URL: http://www.pnas.org/content/77/8/4391.

[158] J. Westermann, B. Engelhardt, and J. Hoffmann. Migration of T cells in vivo: Molecular mechanisms and clinical implications. *Annals of Internal Medicine*, 135:279–295, 2001.

[159] J. Westermann and R. Pabst. Distribution of lymphocyte subsets and natural killer cells in the human body. *Clinical Investigator*, 70:539–544, 1992. doi:10.1007/BF00184787.

[160] J. Westermann, S. Persin, J. Matyas, P. van der Meide, and R. Pabst. IFN-γ influences the migration of thoracic duct B and T lymphocyte subsets in vivo. *Journal of Immunology*, 150:3843–3852, 1993.

[161] J. Westermann, Z. Puskas, and R. Pabst. Blood transit and recirculation kinetics of lymphocyte subsets in normal rats. *Scandinavian Journal of Immunology*, 28:203–210, 1988. doi:10.1111/j.1365-3083.1988.tb02432.x.

[162] J. Westermann, S. Söllner, E.-M. Ehlers, K. Nohroudi, M. Blessenohl, and K. Kalies. Analyzing the migration of labeled T cells in vivo: An essential approach with challenging features. *Laboratory Investigation*, 83(4):459–469, 2003. doi:10.1097/01.LAB.0000062852.80567.90.

[163] S. T. Wierzchoń. Generating optimal repertoire of antibody strings in an artificial immune system. In *Intelligent Information Systems*, Advances in Soft Computing, pages 119–133. Physica-Verlag, 2000.

[164] R. S. Wildin, S. Smyk-Pearson, and A. H. Filipovich. Clinical and molecular features of the immunodysregulation, polyendocrinopathy, enteropathy, X linked (IPEX) syndrome. *Journal of Medical Genetics*, 39:537–545, 2002. doi:10.1136/jmg.39.8.537.

[165] K. Yusim, B. T. M. Korber, C. Brander, B. F. Haynes, R. Koup, J. P. Moore, B. D. Walker, and D. I. Watkins. HIV molecular immunology 2009. Technical Report LA-UR 09-05941, Los Alamos National Laboratory, Theoretical Biology and Biophysics, Los Alamos, New Mexico, 2009.

List of Figures

List of Tables

Acknowledgements

Like every thesis, the present one would not have been possible without the advice, help, encouragement, and all kinds of other support from many important human beings. My advisor Rüdiger Reischuk put great faith in me and let me work for several years on ideas that must have seemed somewhat wild to him. I greatly owe to Jürgen Westermann, who has continuously supported and encouraged my interest in immunology since we first met in my first year. Our regular interactions and discussions have been invaluable in shaping my perception and appreciation of that fabulous field. On the computer science side, the patient and critical advice of Maciej Liśkiewicz, from whom I learnt most of what I know about machine learning and learning theory, have provided excellent guidance and support, especially during the last two years. In particular, I wish to thank him for his thorough and careful reading of Chapter 4.

I am indebted to Michael Elberfeld for developing together with me some important initial results on negative selection algorithms. Him and my other long-term colleagues Markus Hinkelmann and Till Tantau I also thank for a pleasant and inspiring work environment and day-to-day routine.

Outside my *alma mater*, I want to express my deep gratitude to Joost Beltman and Rob de Boer from the Institute of Theoretical Biology and Bioinformatics at the University of Utrecht. I greatly enjoyed the productive and inspiring time I had in Utrecht, and the interactions and discussions with both of them gave me confidence that I was on a good track. I also thank Jorg Calis and Can Keşmir for some inspiring discussions on epitope prediction, which raised many more questions than this thesis could answer.

My most personal thoughts go out to Gesa, whose caring companionship carried me across these years; and to my family, whose support made this all possible.