Lecture — February 3rd, 2025

Artificial Immune Systems

Immunology's solutions to anomaly detection

inge.wortel@ru.nl

Two sides of "Natural Computing"

Two basic questions at the heart of this course:

1. Can we design algorithms for computing science problems that are inspired by the natural world?
2. Can we improve our understanding of the natural world by using computer science algorithms?

First: focus on question 1. After the break: some modelling with an AIS (question 2).

Today's Goals

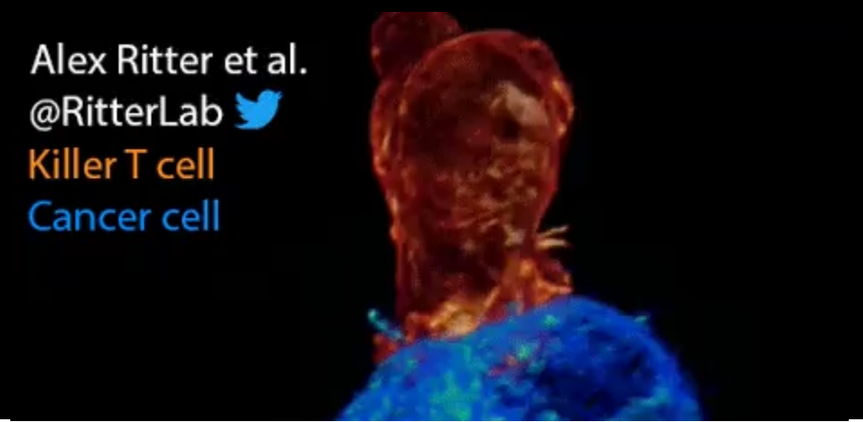
Lecture:

- explain what Artificial Immune Systems are
- describe how negative selection is an algorithm for anomaly detection
- adapt this algorithm to perform anomaly detection for simple datatypes

In the assignment, you will:

- apply the negative selection algorithm to a given dataset
- evaluate and tune the performance of your algorithm

Inspiration: natural anomaly detection

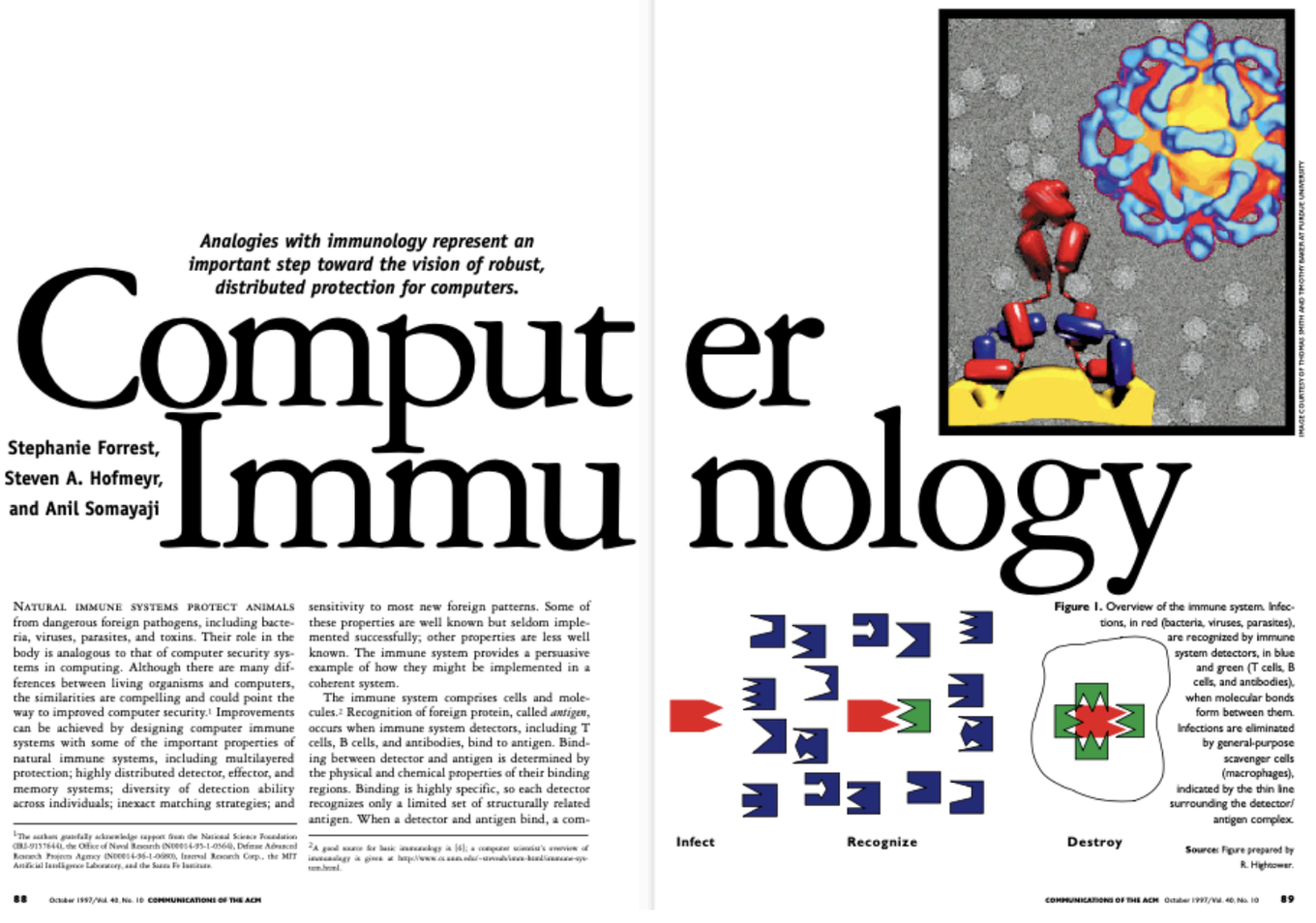


Why "immunology-inspired" computing?

The immune system has a number of attractive features worth replicating:

- **layered** protection (with multiple fail-safes)
- **distributed** protection (no single, "central" control to disable)
- **(bio)diverse** protection (no two persons are the same!)
- **broad** protection (even against previously unseen foreign material)

Why "immunology-inspired" computing?



Analogies with immunology represent an important step toward the vision of robust, distributed protection for computers.

Computer Immunology

Stephanie Forrest,
Steven A. Hofmeyr,
and Anil Somayaji

NATURAL IMMUNE SYSTEMS PROTECT ANIMALS from dangerous foreign pathogens, including bacteria, viruses, parasites, and toxins. Their role in the body is analogous to that of computer security systems in computing. Although there are many differences between living organisms and computers, the similarities are compelling and could point the way to improved computer security.[1] Improvements can be achieved by designing computer immune systems with some of the important properties of natural immune systems, including multilayered protection; highly distributed detector, effector, and memory systems; diversity of detection ability across individuals; inexact matching strategies; and sensitivity to most new foreign patterns. Some of these properties are well known but seldom implemented successfully; other properties are less well known. The immune system provides a persuasive example of how they might be implemented in a coherent system.

The immune system comprises cells and molecules.[2] Recognition of foreign protein, called *antigen*, occurs when immune system detectors, including T cells, B cells, and antibodies, bind to antigen. Binding between detector and antigen is determined by the physical and chemical properties of their binding regions. Binding is highly specific, so each detector recognizes only a limited set of structurally related antigen. When a detector and antigen bind, a com-

[1]The authors gratefully acknowledge support from the National Science Foundation (IRI-9157644), the Office of Naval Research (N00014-95-1-0364), Defense Advanced Research Projects Agency (N00014-96-1-0680), Interval Research Corp., the MIT Artificial Intelligence Laboratory, and the Santa Fe Institute.

[2]A good source for basic immunology is [6]; a computer scientist's overview of immunology is given at http://www.cs.unm.edu/~steves/imm-html/immune-system.html.

Figure 1. Overview of the immune system. Infections, in red (bacteria, viruses, parasites), are recognized by immune system detectors, in blue and green (T cells, B cells, and antibodies), when molecular bonds form between them. Infections are eliminated by general-purpose scavenger cells (macrophages), indicated by the thin line surrounding the detector/antigen complex.

Source: Figure prepared by R. Hightower.

Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji (1997). Communcations of the ACM, 40(10):88-96.

Artificial Immune Systems (AISs)

An **artificial immune system** is any algorithm/engineered system inspired by the real immune system.

The real immune system is incredibly **complex and diverse** — and so are AISs!

We will focus on one particular AIS: the **negative selection algorithm**. In the second half of this lecture, we will use it to answer a biological question, but first we will see how it can be used for natural computing in general.

Source of inspiration: biological immunity

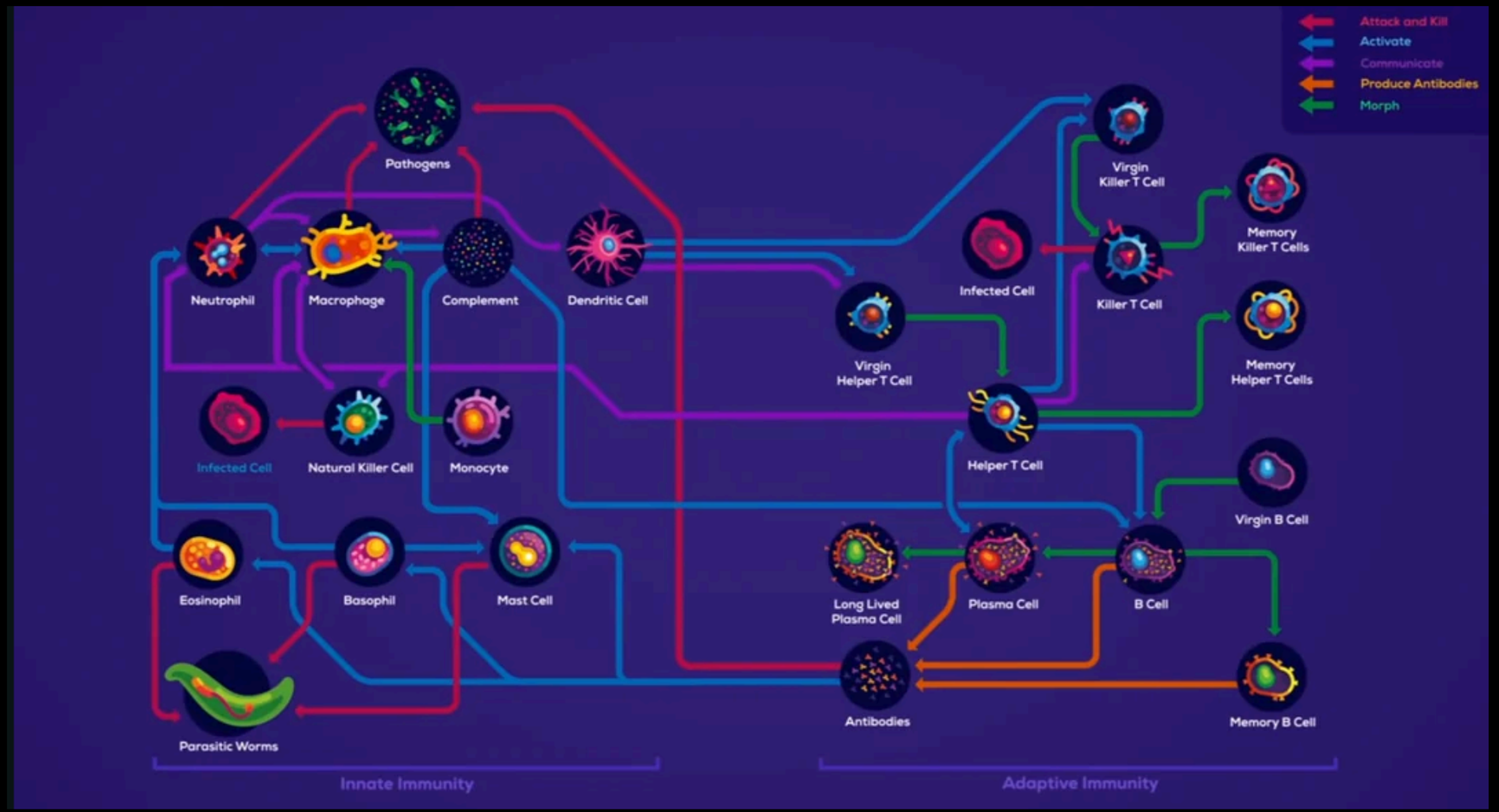
Let's first examine how the immune system accomplishes these features:

- **layered** protection (with multiple fail-safes)
- **distributed** protection (no single, "central" control to disable)
- **(bio)diverse** protection (no two persons are the same!)
- **broad** protection (even against previously unseen foreign material)

The immune system is *multi-layered*

Multiple layers of protection in the immune system:

- The **skin**: passive but crucial (prevent: "mopping up the floor with the tap open")
- The **complement** system: molecules in blood, generic & passive
- The **innate immune system**: cells that actively detect **generic** pathogenic patterns
- The **adaptive immune system**: cells that actively detect **specific** pathogenic patterns


Attack and Kill
Activate
Communicate
Produce Antibodies
Morph
Pathogens
Neutrophil
Macrophage
Complement
Dendritic Cell
Infected Cell
Natural Killer Cell
Monocyte
Eosinophil
Basophil
Mast Cell
Parasitic Worms
Virgin Killer T Cell
Infected Cell
Killer T Cell
Memory Killer T Cells
Virgin Helper T Cell
Memory Helper T Cells
Helper T Cell
Virgin B Cell
Long Lived Plasma Cell
Plasma Cell
B Cell
Antibodies
Memory B Cell
Innate Immunity
Adaptive Immunity

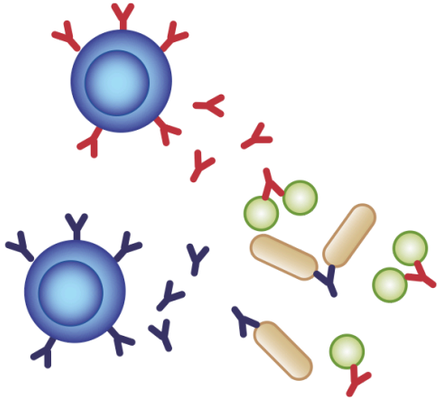
The (adaptive) immune system is *distributed*

The adaptive immune system is a population of **distributed detectors**. It has two major subsystems: B cells and T cells

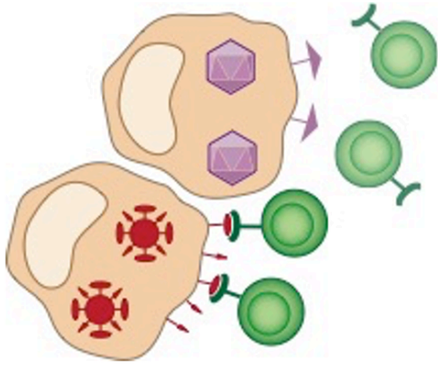
- Bursa (or bone marrow)-derived **B cells** make soluble **antibodies** to attack pathogens in the blood
- Thymus-derived **T cells** can attack and kill infected/cancerous cells
- Both are highly specialized or **"specific"** we have many different cells that detect different things.

Today's focus: T cells. What data do T cells see?

B cells



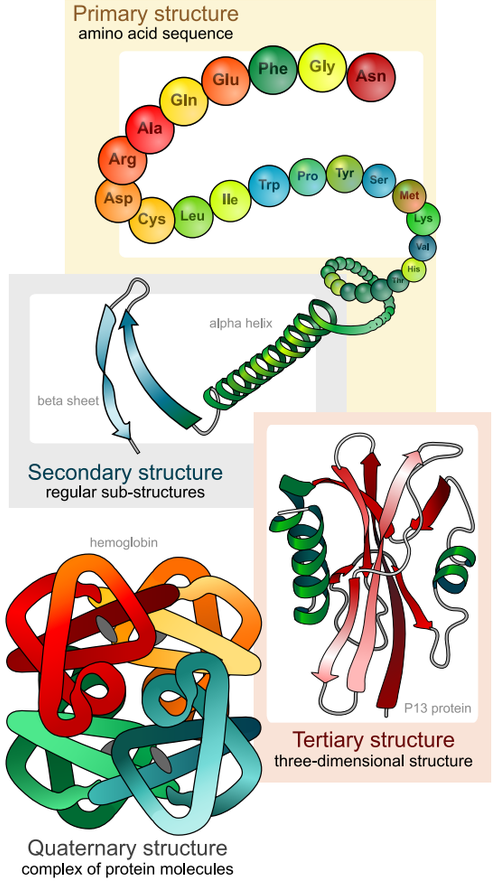
T cells



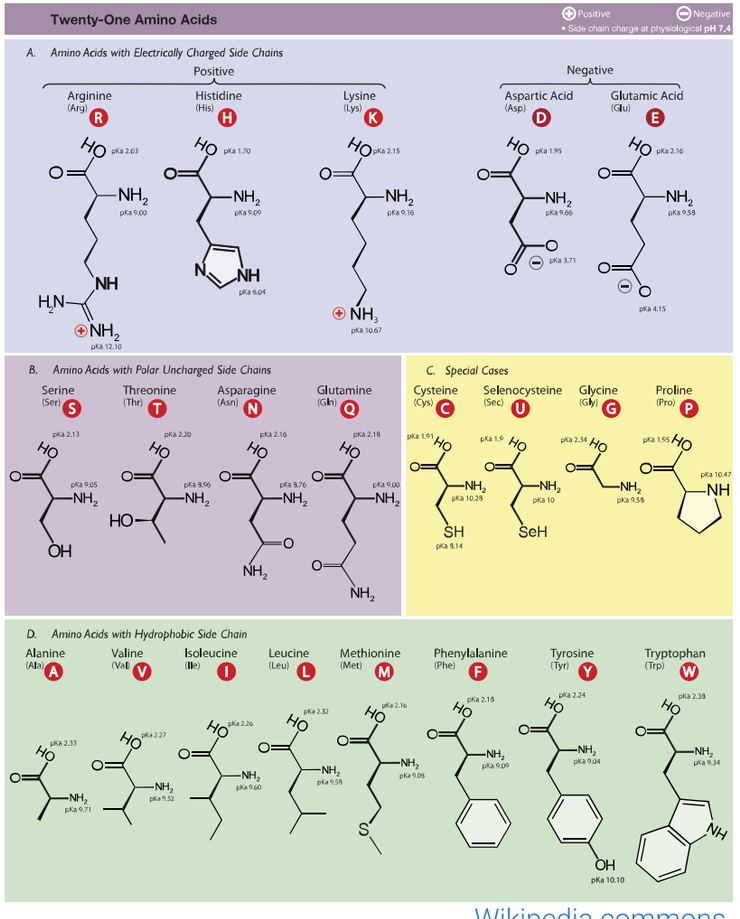
How T cells can see what is inside our cells

Human cells constantly make **proteins** and degrade them into fragments.

These fragments, **peptides**, are essentially **strings** of the amino acid "alphabet".



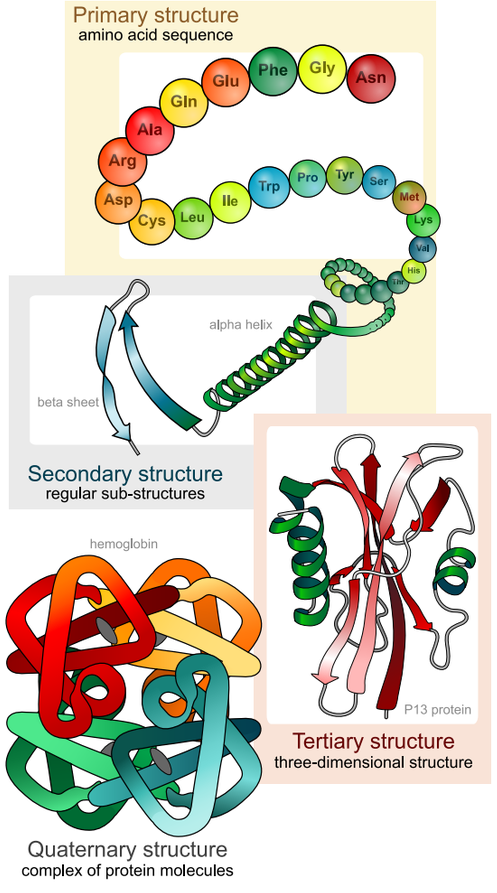
Wikipedia commons.



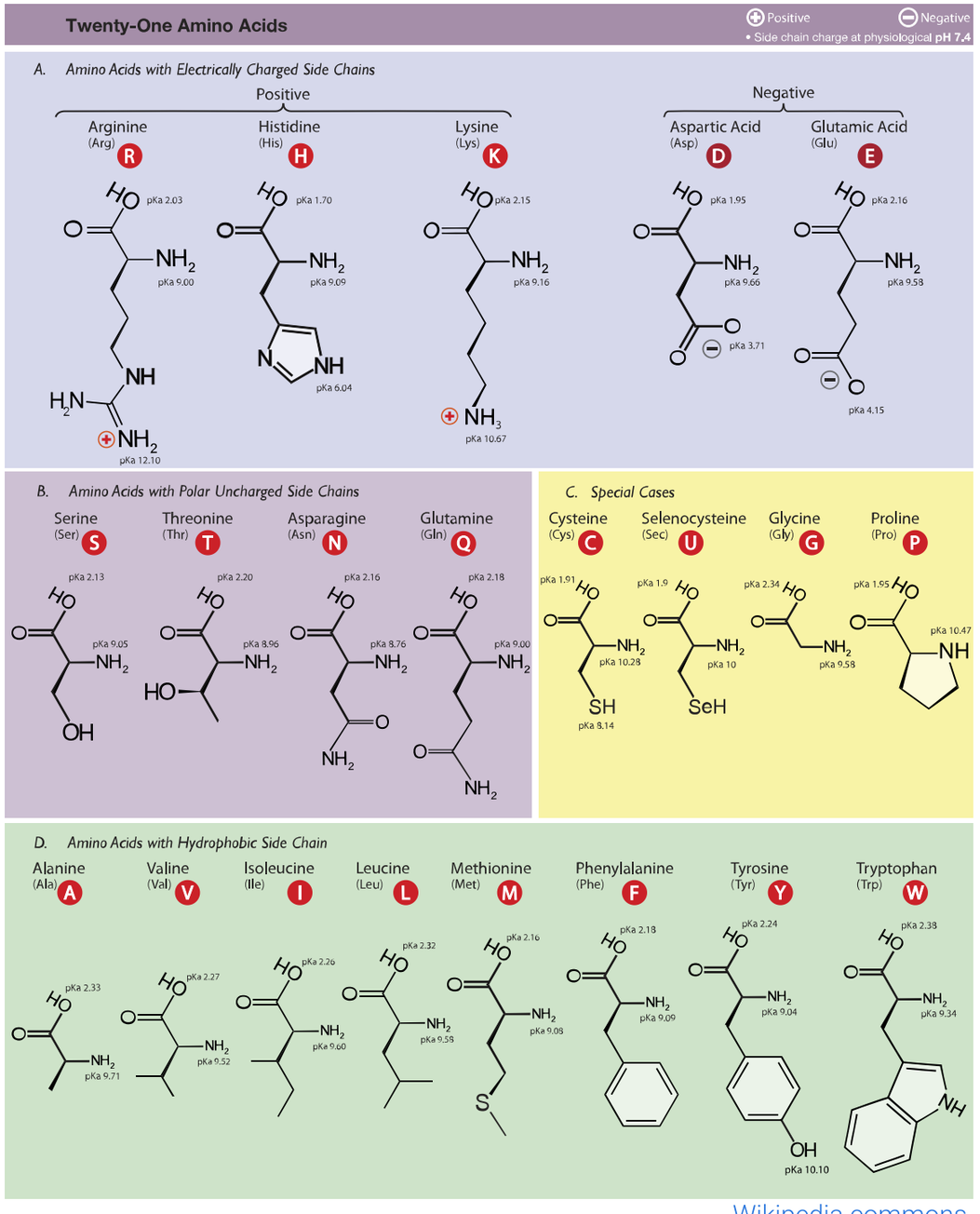
Wikipedia commons.

How T cells can see what is inside our cells

These fragments, **peptides**, are essentially **strings** of the amino acid "alphabet".



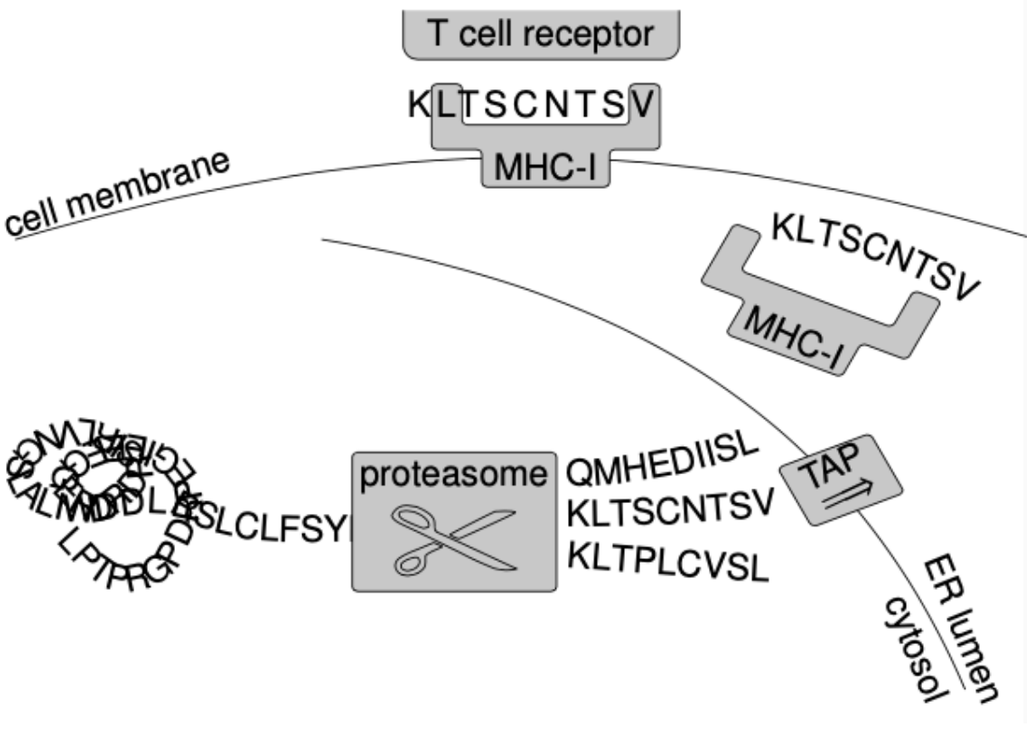
Wikipedia commons.



Wikipedia commons.

How T cells can see what is inside our cells

All cells (healthy or infected) **actively present** these peptides on their outer membrane, like little flags, using so-called **MHC molecules** ("flag-poles").



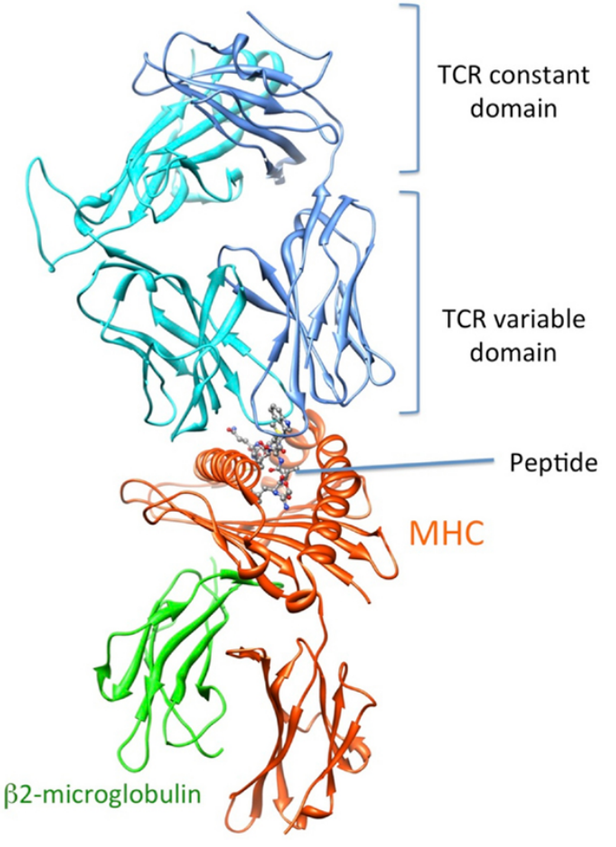
How T cells can see what is inside our cells

T cells constantly scan these MHC complexes for **anomalous ("non-self") peptides**:

- Healthy cells have only "self" peptides derived from "self" proteins
- Infected/cancerous/dying cells have "non-self" peptides from foreign/mutated proteins

Their task becomes:

- **kill** infected/cancerous cells presenting (any!!!) non-self peptides, while...
- ...**tolerating** healthy cells presenting self peptides



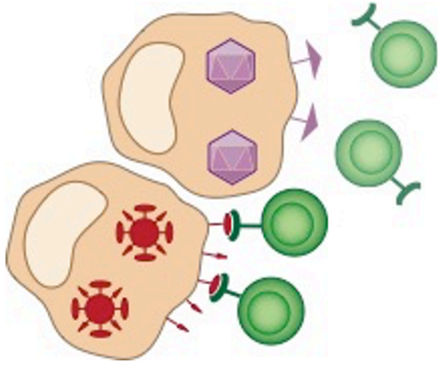
Zoete et al, Front. Immunol (2013).

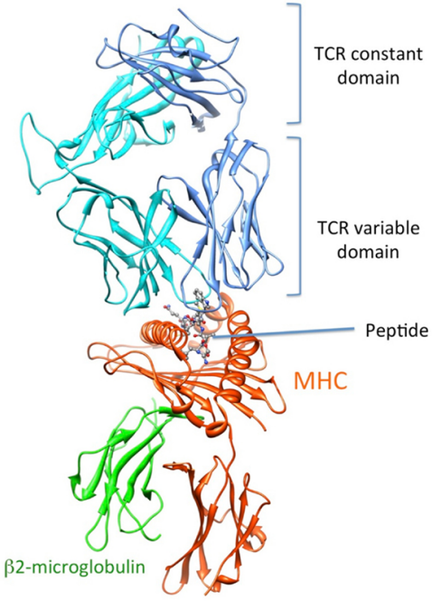
T-cell immunity (updated)

The immune system's T cells form a population of **distributed detectors**:

- The **data** they detect consists of **peptides** on the outside of cells, which are essentially short strings of the amino-acid "alphabet"
- Each T cell has a unique **T-cell receptor**, which serves as the molecular **detector** for anomalous peptide "data"
- Each T-cell receptor is **specific** and detects only 1 in every 100 000 peptides
- So protection against many different pathogens is **distributed** over many different T-cell receptors (and thus over many different T cells)

T cells





Broad yet specific immunity: a numbers problem

T cells must **discriminate** "self" vs "non-self" peptide strings:

- recognize **>10^{15}** potential (and continuously changing) non-self peptides ... [1]
- ... while tolerating **hundreds of thousands** of self peptides

Question

Can you see a reason why this might be difficult?

[1] Sewell (2012). *Nature Reviews Immunology.*

How do we get that many T cells?!

The problem:

- T-cell receptors are themselves proteins, encoded in our DNA
- We need (many) millions of T-cell receptors to "cover" all pathogens
- ... but our DNA only has space for ~20,000 genes...
- We would need 11 orders of magnitude more DNA to encode all those receptors!

How do we get that many T cells?!

The problem:

- T-cell receptors are themselves proteins, encoded in our DNA
- We need (many) millions of T-cell receptors to "cover" all pathogens
- ... but our DNA only has space for ~20,000 genes...
- We would need 11 orders of magnitude more DNA to encode all those receptors!

The solution: randomize!

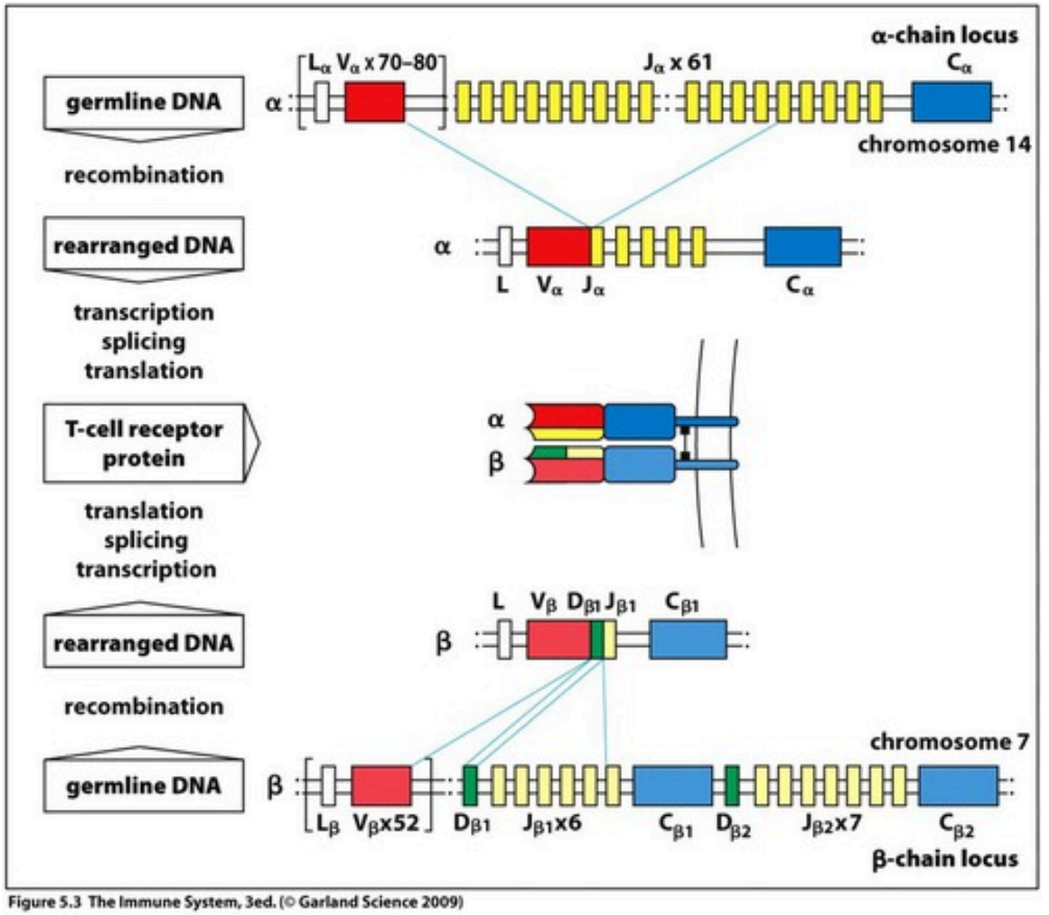


Figure 5.3 The Immune System, 3ed. (© Garland Science 2009)

The immune system is *diverse*

Randomly generated T cells are **diverse**:

- *within* individuals
- *between* individuals

Question

Can you see both a benefit and a problem of this solution?

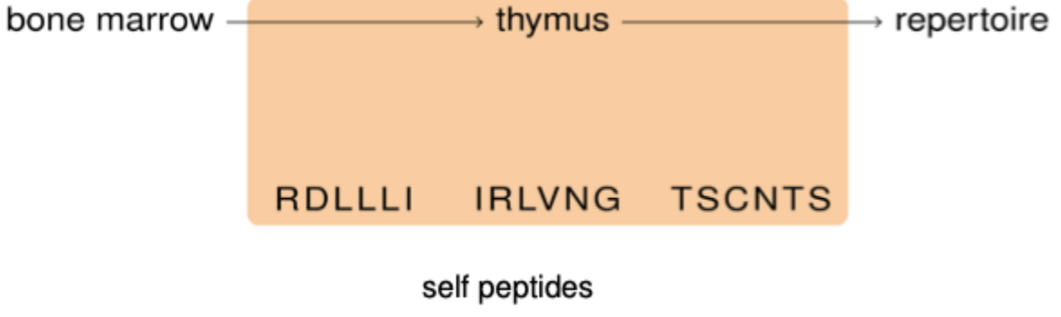
Negative selection

The (new) problem:

- Randomly generated T-cell receptors might detect also normal "self" peptides
- Such T cells can then attack our healthy cells; this is dangerous (auto-immunity)!

Solution — **negative selection**:

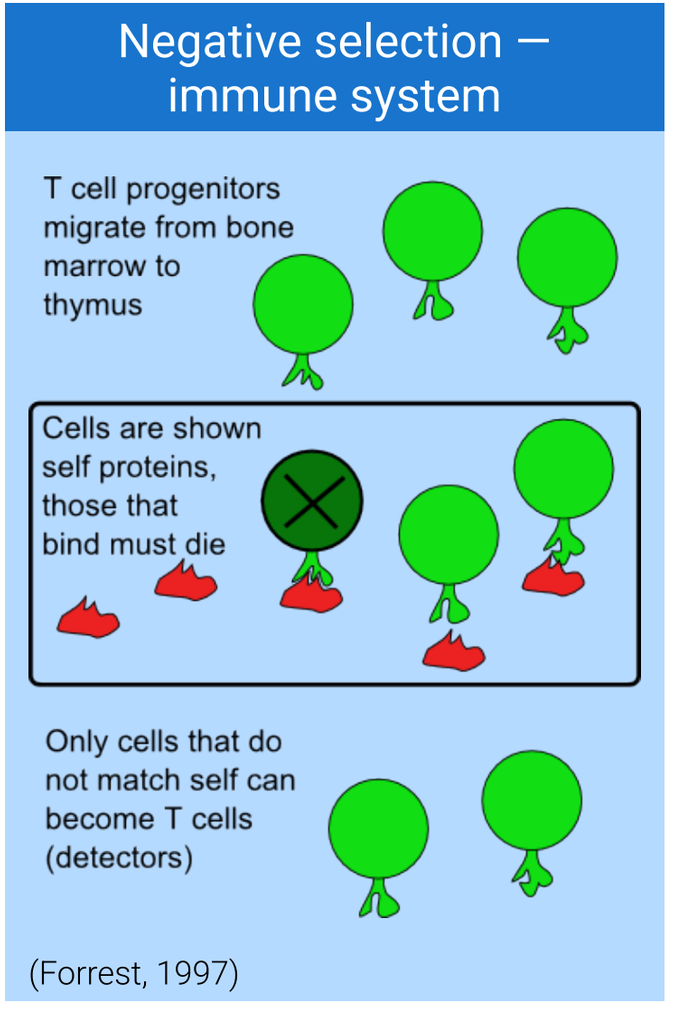
- T cells are born in the bone marrow, then go to the thymus for "training"
- There, they are then exposed to normal, "self" peptides...
- ..and they die if they recognize those.
- We are left with a **T-cell repertoire** where "self-reactive" cells have been **deleted**



Diversity, but safe: solution?

Two-step algorithm:

1. **Make a "repertoire" of many T cells.**
   A **stochastic** process ensures that this repertoire is both broad and diverse.
   → But: such a stochastic process also generates **self-reactive** T cells.
2. **Negatively select (i.e., kill) harmful cells.**
   Result: a "self-tolerant" T-cell repertoire.



T-cell immunity: back to our features

- **layered protection**:
  (T-cell) immunity is much more complex than what we covered here.
- **distributed protection**:
  distributed over many unique "detectors" (cells, T-cell receptors).
- **(bio)diverse protection**:
  Diverse T cells can be generated, and no two individuals have the same T-cell repertoire.
- **broad protection**:
  T cells can protect even against previously unseen foreign material if receptor recombination can generate receptors for (almost) any peptide.

From "inspiration" to generic algorithm(s)

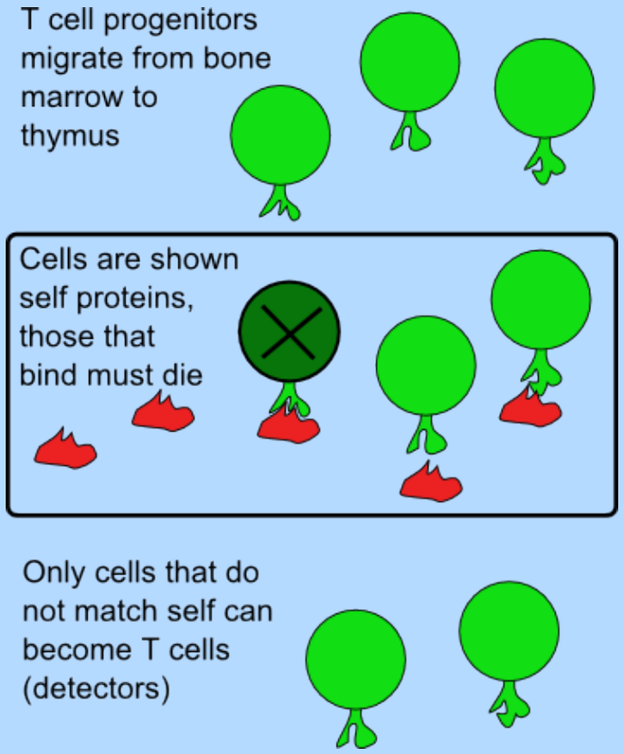
The immune system must solve a **one-class classification problem** (anomaly detection)

- We need to solve a binary classification problem ("normal" vs "anomalous" peptides)...
- ...but have only training examples from one class ("normal" peptides during negative selection)

Negative selection is an **algorithm** that solves this problem, and we will see that this algorithm can **generalize**.

Negative selection algorithm: general structure

Negative selection — IS



Negative selection algorithm (training)

Let X be the **learning domain** containing both SELF and NON-SELF, and D_0 an initial repertoire of **detectors** d each matching a subset of X.

input: training set $S \subseteq$ SELF, repertoire D_0

output: Trained repertoire D^*

initialize $D^* \leftarrow D_0$
for each $d \in D^*$ **do**
  if d **matches** any $s \in S$ **then**
    delete d from D^*

Negative selection algorithm (usage)

input: trained D^*, monitoring set $M \subseteq X$

output: For each $m \in M$: class SELF or NON-SELF

for each $m \in M$ **do**
  if m **matches** any $d \in D$ **then**
    output "m is NON-SELF"
  else
    output "m is SELF"

(Forrest, 1997)

Negative selection algorithm: general structure

Key ingredients:

- **Learning domain** of instances to classify as self/non-self (or normal/anomalous)
- Some definition of a **detector** that "covers" a small subset of the learning domain
- **Matching rule** to decide if a given detector *d* matches a given instance from the learning domain

We can apply this generic scheme to diverse data types (learning domains) as long as we can define a detector and matching rule.

Example 1: String-based negative selection

Suppose we want to distinguish "normal" from "anomalous" (binary) strings of length ℓ. We have only normal examples to train on.

- What is the learning domain? $\{0,1\}^\ell$ (strings of length ℓ)
- How could we define a detector? $\{0,1\}^\ell$ (also strings of length ℓ)
- How could we define a matching rule? Any rule to set a minimal "similarity" between two given strings, e.g. minimal number of matches and/or specific positions that must match.

Example 1: String-based negative selection

Negative selection

input: self-set $S \subseteq$ SELF

output: final detector set D

$D \leftarrow$ random set of **patterns** $\subseteq \{0,1\}^\ell$
for each $s \in S$ **do**
  for each $d \in D$ **matching** s **do**
    delete d

Learning domain X: $\{0,1\}^\ell$
Detectors d: patterns of length ℓ with parameter r
Generalization occurs, with a strength depending on r.

Matching rule: e.g. r-contiguous

Detector d of length ℓ matches string s of length ℓ if d and s contain the same symbol at r subsequent positions.

Assignment: anomaly detection in Unix processes

Idea: trace unix system calls of a daemon process like sendmail, coding each system call as an alphabet symbol. Generate a database of system call sequences in normal circumstances.

Apply negative selection to these sequences and generate detectors, which may be capable to detect anomalous operations.

Example	
iKEKEEFEEgKEgKEEFEggFKvafvaeggggggj	normal
6Lgggggngngngg LLnnPPggnnggLggiKEKE	exploit

So our strings don't have to be binary! In your assignment, you will apply negative selection to such data.

S. Forrest, S. A. Hofmeyr, A. Somayaji, T. A. Longstaff (1996). A sense of self for unix processes. IEEE Symposium on Security and Privacy.

Example 2: Real-valued negative selection (2D)

How could we define:

- Detectors covering a subset of the domain?
- Matching rule deciding which detectors match which points in space X?

Learning domain *X*: $\mathbb{R}^2$

Example 2: Real-valued negative selection (2D)

Negative selection

input: self-set $S \subseteq \text{SELF}$

output: final detector set D

$D \leftarrow$ random set of **circles**
for each $s \in S$ **do**
  for each $d \in D$ **matching** s **do**
    delete d

Learning domain X: $\mathbb{R}^2$
Detectors d: circles with fixed radius r
Generalization occurs, with a strength depending on r.

Matching rule

Detector d matches any point s contained within its circle of radius r.

How useful is such an AIS?

Back to our desirable features:

- **layered** protection (with multiple fail-safes)
- **distributed** protection (no single, "central" control to disable)
- **(bio)diverse** protection (no two persons are the same!)
- **broad** protection (even against previously unseen foreign material)

Our negative selection algorithm fulfils at least the last three. So why are they not more broadly used?

- This only works with **large repertoires** of detectors...
- ...requiring many string comparisons, which has been too inefficient (so far).
- Improved implementations may solve this issue — but those are beyond the scope of this lecture.

Summary

The immune system's T cells are negatively selected to match anything but self. This biological learning algorithm allows the immune system to generalize from examples of "self".

A generic one-class classifier can be based on the negative selection principle:

- applicable to many data domains (but mostly strings)
- requires a population ("repertoire") of distributed detectors, each specializing in only a part of our learning domain X
- implementing such an algorithm *efficiently* requires substantial work, but that is out of the scope of this lecture

After the break, we will apply this algorithm to (hopefully) learn something about the biological version of negative selection.