

Seminar

Anwendungen und Algorithmen basierend auf Prinzipien eines
künstlichen Immunsystems

Überblick über das Immunsystem & Simulation des Immunsystems

vorgelegt von:
Uwe Schreiner

Betreuer:
Thomas Stibor

Inhaltsverzeichnis

	Seite
Inhaltsverzeichnis	1
Abkürzungsverzeichnis	2
1 Einleitung	3
1.1 Das Immunsystem	3
1.1.1 Die lymphatischen Organe	3
1.2 Die unspezifische, nicht adaptive Immunabwehr	4
1.2.1 Humorale Faktoren	4
1.2.2 Zelluläre Faktoren	5
1.3 Die spezifische, adaptive Immunabwehr	5
1.3.1 T-Lymphozyten	5
1.3.1.1 Differenzierung und Selektion von T-Lymphozyten	6
1.3.2 B-Lymphozyten	7
1.3.2.1 Aktivierung der B-Zelle	7
1.4 Immunglobuline (Antikörper)	8
1.4.1 Immunglobulinklassen	9
1.4.2 Generierung der Antikörpervielfalt	9
1.4.3 Antigenerkennung	11
1.4.4 Antikörperselektivität	11
1.5 Major histocompatibility complex (MHC)	12
1.6 Antigenpräsentierende Zelle	13
1.7 Natürliche Killer-Zellen (NK-Zelle)	14
1.8 Zielsetzung der Simulation	14
2 Das Computersimulationsmodell IMMSIM	15
2.1 Aufbau und Eigenschaften der zellulären Bestandteile des Modells	15
2.2 Repräsentation der Antikörper, Antigene und Antigen-Antikörperkomplexe im Modell	16
2.3 Interaktionen, Komplementarität und Affinität	16
2.4 Ablauf einer Simulation am Beispiel der B-Zell-Aktivierung	17
2.5 Modellierung lymphatischer Organe	18
2.6 Die Implementierung	19
2.6.1 Die Datenstrukturen	19
2.6.2 Initialisierung der Simulation	19
2.6.3 Die Methoden der Hauptschleife	20
2.6.3.1 Die Methode <code>birth()</code>	20
2.6.3.2 Die Methode <code>produceAb()</code>	20
2.6.3.3 Die Methode <code>iB_Ag()</code>	20
2.6.3.4 Die Methode <code>iAb_Ag()</code>	21
2.6.3.5 Die Methode <code>death()</code>	21
2.6.3.6 Die Methode <code>flow()</code>	22
2.6.3.7 Die Methode <code>diffusion()</code>	22
2.7 Simulationsergebnisse	23
3 Diskussion und Zusammenfassung	24
4 Literaturverzeichnis	25
5 Anhang	27

Abkürzungsverzeichnis

↑	Konzentrationszunahme
↓	Konzentrationsabnahme
[]	Array in der Simulation
[] (mit ↑ bzw. ↓)	Konzentration bei B-Zellen, Antigenen und Antikörper
3D	dreidimensional
A	Antigenpräsentierende Zelle
Ab	Antikörper
Ag	Antigen
Abb	Abbildung
APC	Antigenpräsentierende Zelle
B	B-Zelle
B-Zelle	B-Lymphozyt
C	durch eine Säuregruppe gekennzeichnetes Ende eines Polypeptids
CD	cluster of differentiation
CD4+	cluster of differentiation Typ 4 positiv
CD4/8	sowohl CD4 als auch CD8
CD8+	cluster of differentiation Typ 8 positiv
COOH	Säuregruppe
d.h.	das heisst
Fab	fragment antigen binding
Fc	fragment crystalline
H-Kette	schwere Kette eines Immunglobulins
HIV	human immunodeficiency virus → AIDS
HLA	human leucocyte anitgene → MHC
Ig	Immunglobulin
IgA	Immunglobulin Typ A
IgD	Immunglobulin Typ D
IgE	Immunglobulin Typ E
IgG	Immunglobulin Typ G
IgM	Immunglobulin Typ M
IMMSIM	immune simulation
INF	Interferon
INF γ	Interferon γ
L-Kette	leichte Kette eines Immunglobulins
MHC	Hapthistokompatibilitätskomplex
MHC I	Hapthistokompatibilitätskomplex Klasse I
MHC II	Hapthistokompatibilitätskomplex Klasse II
N	durch eine Aminogruppe gekennzeichnetes Ende eines Polypeptids
Naiv	undifferenzierte B-Zelle
NH ₂	Aminogruppe
NK-Zelle	natürliche Killerzelle
T	T-Zelle
T-Zelle	T-Lymphozyt
TCR	T-Zellrezeptor
TZR	T-Zellrezeptor
TH -Zelle	T-Helfer-Zelle
TH1-Zelle	T-Helfer-Zelle Typ 1
TH2-Zelle	T-Helfer-Zelle Typ 2
u.U.	unter Umständen

1 Einleitung

Wenn ein nicht körpereigenes Pathogen in den menschlichen Organismus eindringt, beginnt das intakte körpereigene Immunsystem den Eindringling erfolgreich zu eliminieren. An dieser Reaktion sind mehr als 10^{12} Zellen kollektiv und koordiniert beteiligt, vergleichbar mit der Anzahl der Synapsen im menschlichen Gehirn. Wegen seiner außerordentlichen Komplexität und Differenziertheit greift man heute bei der Interpretation von experimentellen immunologischen Daten auch auf Computersimulationen des Immunsystems zurück.

1.1 Das Immunsystem

Als Immunsystem werden Organe, Zellen und Eiweißkörper zusammengefasst, deren Funktion in der Erhaltung der Individualstruktur durch die Abwehr körperfremder Substanzen und Krankheitserreger wie Bakterien, Viren, Parasiten oder Pilzen besteht. Voraussetzung dafür ist, dass das Immunsystem zwischen körpereigenen und körperfremden Strukturen unterscheiden kann, so dass im Normalfall keine Immunreaktion gegen den eigenen gesunden Körper erfolgt ("immunologische Toleranz"). Allerdings ist es ebenfalls Aufgabe des Immunsystems, krankhafte körpereigene Zellen wie Tumorzellen zu erkennen und anzugreifen. Die im Laufe der Evolution entwickelte Abwehr des Körpers gliedert sich in zwei Hauptsysteme: Die **angeborene, unspezifische Immunität**, die vor allem bei der Bekämpfung bakterieller Infektionen von grundlegender Bedeutung ist, und die **erworbene, spezifische Immunität** gegen jeweils ganz bestimmte Krankheitserreger. Diese richtet sich gegen verkapselte Bakterien und Viren, die eine in der Evolution schnell veränderbare Oberflächenstruktur besitzen. Die spezifischen und unspezifischen Abwehrmechanismen sind eng miteinander vernetzt, wie Abbildung 1 (Abb.1) zeigt.

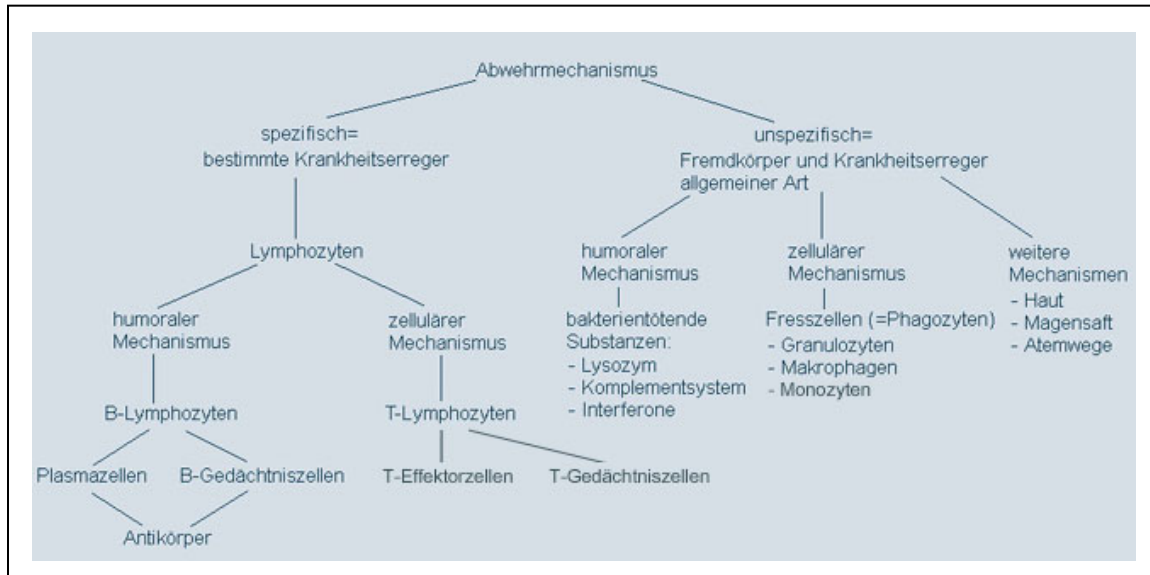


Abb. 1: Schematische Darstellung des Immunsystems

In diesem Graph ist die Struktur des Abwehrmechanismus, bestehend aus der unspezifischen, nicht adaptiven Immunabwehr und der spezifischen, adaptiven Immunabwehr dargestellt. Zu erkennen sind auch die Bestandteile der einzelnen Mechanismen.

1.1.1 Die lymphatischen Organe

Die verschiedenen Organe, die an der Entstehung der Immunantwort beteiligt sind, gliedern sich in die **zentralen, primären lymphatischen Organe**, das Knochenmark und den Thymus und in die **peripheren, sekundären lymphatischen Organe**, wie Lymphknoten, Milz

und die lymphatischen Gewebe des Magen-Darm-Traktes (Rachenmandeln, Blinddarm u.a.), der Lunge und anderer Schleimhäute (siehe Abb. 2). Während das **Knochenmark und Thymus** für die Bildung von Lymphozyten zuständig sind, die dann über das Blut zu den peripheren lymphatischen Organen transportiert werden, wird in den sekundären Organen die erworbene Immunabwehr eingeleitet.

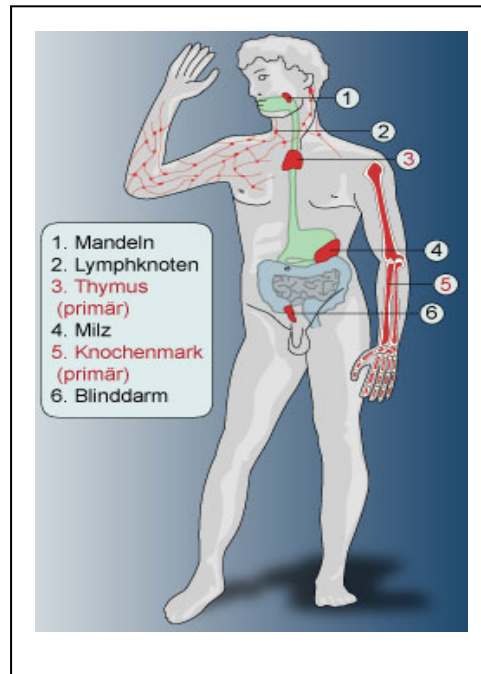


Abb. 2: Lage der lymphatischen Organe im menschlichen Körper
Gezeigt ist die Lage der primären, lymphatischen Organe (Knochenmark, Thymus), rot gekennzeichnet und der sekundären, lymphatischen Organe (schwarz gekennzeichnet) im menschlichen Körper.

1.2 Die unspezifische, nicht adaptive Immunabwehr

Die unspezifische, nicht antigenspezifische, d.h. nicht adaptive Abwehr ist in der Lage, Fremdkörper und viele allgemein vorkommende Krankheitserreger ohne lange Anlaufphase bereits beim ersten Kontakt unschädlich zu machen. Es wird deshalb von ihr auch als der angeborenen Immunität gesprochen. Sie ist für die Bekämpfung bakterieller Infektionen von großer Bedeutung. Zur unspezifischen Abwehr gehören humorale und zelluläre Mechanismen.

1.2.1 Humorale Faktoren

Die so genannten humoralen (lat. humor = Flüssigkeit), d.h. die in den Körperflüssigkeiten gelösten Faktoren des unspezifischen Abwehrsystems, sind bakterizid wirkende (=bakterien-tötende) Substanzen. Dazu gehört das Enzym Lysozym, das in verschiedenen Körpersekreten wie Tränenflüssigkeit und Speichel enthalten ist und die Zellwand zahlreicher Bakterien angreift. Daneben gibt es das so genannte **Komplementsystem**. Es handelt sich um ein von der Leber gebildetes Enzymsystem, das aus einer Gruppe von etwa 20 Bluteiweißkörpern besteht und zur Auflösung körperfremder Zellen führt. Darüber hinaus gehören auch so genannte Interferone, die eine vorwiegend gegen Viren gerichtete Wirkung haben, zur unspezifischen, humoralen Abwehr.

1.2.2 Zelluläre Faktoren

Die Abwehrzellen des unspezifischen Systems sind die Phagozyten, die den Erreger oder Fremdkörper aufnehmen und ihn "verdauen". Die Zellen werden deshalb auch "Fresszellen" genannt. Zu den Phagozyten gehören neutrophile und eosinophile Granulozyten, Makrophagen und Monozyten. Teilweise werden auch Lymphozyten, Mastzellen und Fibroblasten zu den Phagozyten gezählt. Diese nehmen zwar gelegentlich auch Fremdpartikel auf, verdauen diese aber nicht, sondern stoßen sie in Interzellulärräume wieder ab, wo sie von den eigentlichen Phagozyten vernichtet werden.

Neben den humoralen und zellulären Mechanismen unterstützen weitere Faktoren die unspezifische Immunabwehr. So bietet die gesunde Haut einen natürlichen Schutz gegen das Eindringen von Krankheitserregern. Magensaft vernichtet durch seinen hohen Säuregehalt Bakterien, die mit der Nahrung aufgenommen werden. Krankheitserreger, die durch die Atemluft in die Luftwege geraten, bleiben dort am von der Schleimhaut gebildeten Schleim hängen und werden durch den Schlag der Flimmerhaare aus dem Körper geschleust. Niesen oder Husten dienen dem gleichen Ziel.

1.3 Die spezifische, adaptive Immunabwehr

Die spezifische Abwehr entwickelt sich im Gegensatz zur unspezifischen erst in der direkten Auseinandersetzung mit einem bestimmten Krankheitserreger und auch dann erst, wenn die unspezifische Immunantwort erfolglos geblieben ist. Sie wird daher auch als erworbene, adaptive Immunität bezeichnet. Es kommt dabei zur Ausbildung besonderer Schutzmaßnahmen, für die eine Anlaufphase von 4 - 7 Tagen benötigt wird und die ganz gezielt gegen einen bestimmten Krankheitserreger gerichtet sind. Die während der Immunantwort gebildeten Gedächtniszellen beugen einem erneuten Angriff desselben Erregers vor. Darüber hinaus hat das spezifische Immunsystem die Fähigkeit, krankhafte körpereigene Zellen wie Tumorzellen zu erkennen und anzugreifen.

Die spezifische Immunabwehr wird durch Immunzellen, d.h. Zellen, die zu immunologischen Reaktionen befähigt sind, vermittelt. Es handelt sich dabei um so genannte Lymphozyten (lat. *lympha* = klares Wasser, griech. *kytos* = Zelle). Lymphozyten sind die kleinsten weißen Blutkörperchen. Ihr Anteil an der Gesamtmenge der weißen Blutkörperchen im Blut beträgt etwa ein Viertel. Allerdings befinden sich 98 % der Lymphozyten nicht im Blut, sondern in den lymphatischen Organen (Lymphknoten, Lymphbahnen, Milz) und im Knochenmark. Von dort aus wird ständig ein kleiner Teil der Zellen ins Blut abgegeben. Die Lebensdauer der Lymphozyten beträgt zwischen zehn Tagen und mehreren Jahren. Sie entwickeln sich zunächst im Knochenmark und im Thymus, d.h. den primären Organen des Immunsystems, und besiedeln von dort aus die sekundären Immunorgane wie Lymphgewebe und Milz.

1.3.1 T-Lymphozyten

Im Thymus werden so genannte T-Lymphozyten (kurz: T-Zellen) gebildet und darauf geprägt, zwischen körpereigenen und körperfremden Strukturen zu unterscheiden. Der Thymus (griech. *thymos* = Brustdrüse) ist ein zweilappiges Organ, das hinter dem Brustbein liegt. Die T-Lymphozyten machen etwa 70-80 % aller Lymphozyten im Blut aus. Sie gehören zur spezifischen zellulären Immunabwehr. Bei Kontakt mit einem Fremdkörper entwickeln sie sich zu so genannten T-Effektorzellen, die verschiedene Immunreaktionen auslösen bzw. verstärken, oder zu länger lebigen **T-Gedächtniszellen**, die auch nach Jahren noch bei einem erneuten Eindringen des gleichen Fremdkörpers diesen erkennen und zu verstärkten Immunreaktionen führen. Die T-Effektorzellen lassen sich in zwei Gruppen einteilen: Die so genannten **T-Helferzellen (TH)**, die B-Zellen (**TH2-Zellen**) und Makrophagen (**TH1-Zellen**) aktivieren, und die so genannten **T-Killerzellen** (oder zytotoxische T-Lymphozyten), die durch Lyse (=Auflösung) infizierte Zellen töten. Unterscheidbar sind TH-Zellen von zytotoxi-

schen T-Zellen dadurch, dass TH-Zellen Differenzierungsantigene vom Typ CD4 und zytotoxische T-Zellen die des Typs CD8 jeweils auf ihrer Zelloberfläche besitzen. Ferner besitzen T-Lymphozyten auf ihrer Zelloberfläche einen T-Zellrezeptor, der kurze auf MHC-Molekülen präsentierte Antigen-Peptide erkennt, die in den MHC-Spalt eingebaut sind (siehe Abb. 3).

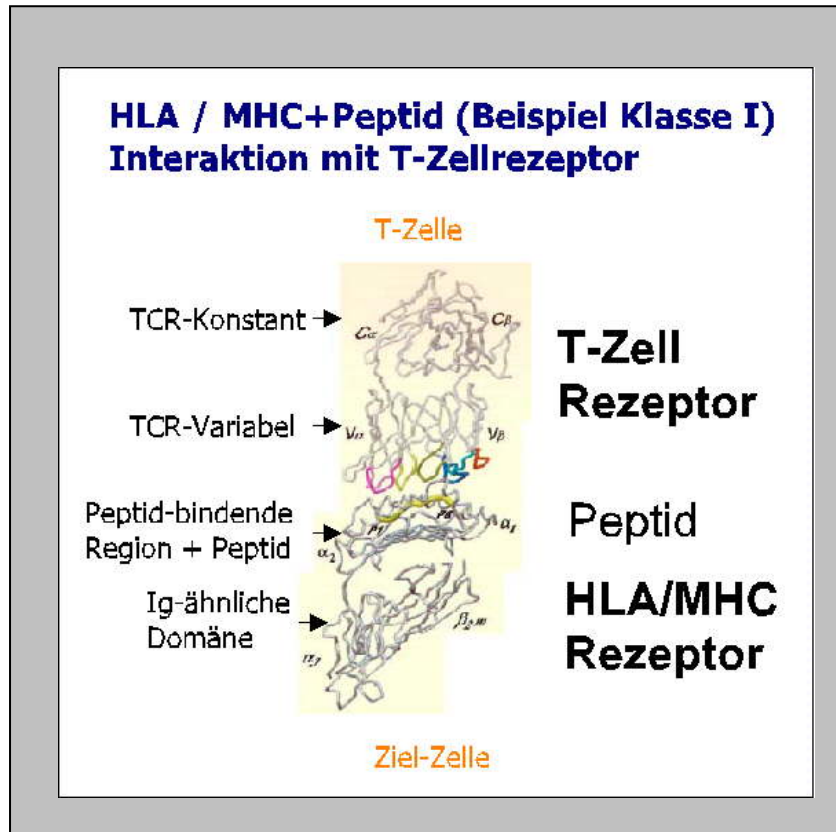


Abb. 3: Darstellung des HLA/MHC-Peptid-TCR-Komplexes

Gezeigt ist die Struktur, die der T-Zellrezeptor (TCR) mit dem Haupthistokompatibilitätskomplex (MHC), beim Menschen auch „human leucocyte antigene“ (HLA) bezeichnet, der ein Peptid trägt, bildet. Strukturell ähnelt er grob dem F_{ab}-Fragment eines Antikörpers: er besteht aus zwei Ketten (α:β bzw. γ:δ), und hat an der "Spitze" eine variable Region.

1.3.1.1 Differenzierung und Selektion von T-Lymphozyten

Die Differenzierung von T-Zellen im Thymus umfasst zwei Selektionsprozesse, positive Selektion von unreifen Thymozyten, die den eigenen Haupt-Histokompatibilitätskomplex (MHC) erkennen, und negative Selektion von unreifen Thymozyten, die Selbstantigene erkennen und damit potentiell autoreaktiv sind. Die zellulären und molekularen Mechanismen beider Selektionsprozesse sind noch weitgehend unverstanden. Bekannt ist allerdings, dass bei der **positiven Selektion** zwei separierbare TCR-vermittelte Signalschritte erforderlich sind. Das erste Signal vermittelt "lineage commitment", d.h. die Differenzierung von bipotenten CD4/8 Vorläuferzellen in CD4 T-Zellen (T-Helferzellen) und eine transiente Rettung vor dem programmiertem Zelltod. Erst das zweite Signal vermittelt die irreversible Differenzierung in reife, langlebige CD4+ T-Zellen. Unklar ist derzeit, ob beide Differenzierungsschritte die gleichen intrazellulären Signalwege benutzen.

Bei der **negativen Selektion** werden die autoreaktiven Thymozyten per Apoptose eliminiert.

1.3.2 B-Lymphozyten

B-Lymphozyten reifen im Knochenmark und machen mit ihrer Konzentration von 10^{10} -Zellen etwa 15 % aller Lymphozyten im Blut aus. Sie gehören zur spezifischen humoralen Immunabwehr. Bei Kontakt mit einem Fremdkörper entwickelt sich ein Teil der B-Lymphozyten zu so genannten **Plasmazellen**, die Antikörper (=Immunglobuline, Ig) gegen diesen Fremdkörper bilden. Plasmazellen leben etwa 2-3 Tage. Aus dem anderen Teil der B-Lymphozyten werden nach Kontakt mit einem Fremdkörper langlebige **B-Gedächtniszellen**, die noch Jahre später, auch wenn der Körper nicht mehr diesem Fremdkörper ausgesetzt ist, die gleichen Antikörper bilden können, was eine schnellere Reaktivierung der adaptiven Immunabwehr induziert.

Jede reife B-Zelle besitzt auf ihrer Oberfläche einen für diese Zelle spezifischen Antikörper, der als Antigenrezeptor fungiert und für den diese Zelle für den Rest ihres Lebens zuständig ist (siehe Abb. 4). Diese oberflächenständigen Antikörper sind zunächst vom Typ IgM, später, d.h. bei einer zweiten Aktivierung durch ein Antigen, bedingt durch selektive Expansion, vom Typ IgG (vgl. 1.4.). IgGs besitzen eine grössere Affinität als IgMs, was eine Affinitätsreifung darstellt.

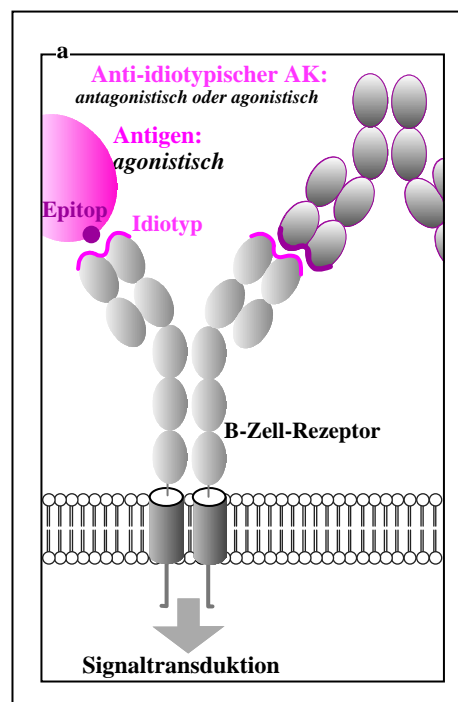


Abb. 4: Struktur des B-Zellrezeptor

Gezeigt ist die Bindung eines als B-Zelloberflächenrezeptors dienenden Antikörpers mit einem Antigen. Während antagonistisch die Hemmung beschreibt, steht agonistisch für das Gegenteil. Dadurch ist die entgegengesetzte Interaktion der Antigene mit den Antikörpern festgelegt. Der Idiotyp ist eine einzelne antigene Determinante auf der variablen Region eines Antikörpers.

1.3.2.1 Aktivierung der B-Zelle

Im Lymphknoten kommen B-Zellen mit den eingespülten Bakterien in Berührung. Tritt das seltene Ereignis ein, daß eine B-Zelle einen passenden B-Zellrezeptor für ein bakterielles Antigen besitzt, nimmt die B-Zelle den Bakterienteil auf, verarbeitet das Antigen und präsentiert es ebenfalls auf MHC-II-Molekülen. Die B-Zelle wird aber noch nicht aktiviert: es fehlt noch die "Entsicherung". Diese tritt ein, wenn im Lymphknoten eine Zelle aus dem aktivierten TH2-Klon auf diese B-Zelle trifft: die TH2-Zelle erkennt das auf dem B-Zell-MHC-II-Komplex präsentierte Antigen-Peptid, für das sie aktiviert ist, und reagiert. Nun sind alle Bedingungen

erfüllt, um die B-Zelle zu aktivieren: sie beginnt ihrerseits zu proliferieren, wie auch die in die Aktivierung involvierten T-Zellen.

Dendritische Zellen fixieren bakterielles Antigen außen an ihrer Zelloberfläche (vgl. 1.6), um den entstehenden B-Zellklon mit ausreichender Stimulation über den B-Zellrezeptor zu versorgen.

Die B-Zellen differenzieren in wenigen Tagen zu Plasmazellen und beginnen, zunächst IgM zu erzeugen; durch class switch in einem Teil der Zellen wird später IgG produziert. Bei Schleimhautinfektionen erfolgt oft auch schon früh ein class switch zu IgA.

Die entstehenden Antikörper tragen dazu bei, daß die schon laufenden nicht-adaptiven Abwehrmechanismen wesentlich effizienter eingesetzt werden.

Wenn die Infektion erfolgreich bekämpft wurde, produzieren ausgereifte Plasmazellen noch unterschiedlich lange Antikörper, so daß eine gewisse Zeit lang ein Schutz vor Reinfektion besteht.

1.4 Immunglobuline (Antikörper)

Ein Hauptproblem der Abwehr ist, dass verschiedene Erregertypen **intrazellulär** (im Zytoplasma), in **Vesikeln** (z.B. Viren) und **extrazellulär** (im Gewebe, auf Epithelzellen) auftreten. Zur Bekämpfung **extrazellulär auftretender Erreger** dienen in erster Linie Antikörper, auch als Immunglobuline bezeichnet.

Wie in Abbildung 5 gezeigt, besteht ein Antikörpermolekül (=Immunglobulin) aus: zwei leichten (vom Typ κ oder λ) und zwei schweren Polypeptidketten (vom Typ μ , δ , γ , α oder ϵ), die durch Disulfidbrücken verbunden sind. Funktionelle Antikörper haben eine variable und eine konstante Region. Während die konstante Region unveränderbar ist, wird die variable Region in einem Prozess, das Rearrangement, neu gebildet. Die variable Region dient der Bindung des Antigens.

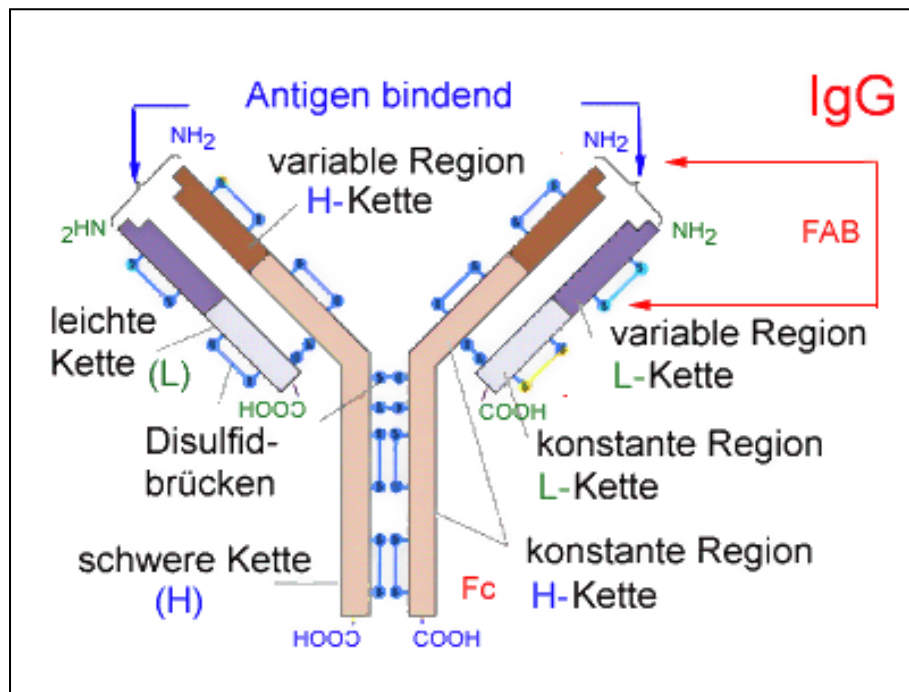


Abb. 5: Struktur des Antikörpers

In dieser IgG-Struktur ist neben der Lage der einzelnen Ketten und Regionen auch die Position der Disulfidbrücken (blau) zu sehen. Die N - terminalen Enden der L- bzw. H-Kette ist mit NH_2 , die c-terminalen Enden mit COOH gekennzeichnet. Auch sind die F_{ab} - F_{c} -Teile, die aus der proteolytischen Spaltung des Antikörpers hervorgehen, angegeben.

Die Antikörper werden von Plasmazellen auf den Reiz eines Antigens hin gebildet (siehe dazu 1.3.2.1). Die meisten Typen zirkulieren in bestimmten Bereichen des Körpers, einige sind an Lymphozyten gebunden. Viele Zellen des Immunsystems haben Rezeptoren, die den Fc-Teil des Antikörpers binden: **so genannte Fc-Rezeptoren**. Mit Fc-Rezeptoren erkennen und binden diese Zellen Antigen-Antikörperkomplexe.

Freie Antikörper werden von den Fc-Rezeptoren nicht gebunden. Eine Ausnahme sind Mastzellen, die auch über freie (nicht an ein Antigen gebundene) IgEs an ihre Oberfläche binden. Eine weitere Ausnahme bilden die von einer reifen B-Zelle produzierten als Membranrezeptoren wirkenden Immunglobuline vom Typ M und Typ D.

1.4.1 Immunglobulinklassen

Je nach Typ der schweren Kette spricht man von IgM, IgG, IgD, IgA oder IgE (Antikörperklassen). Bei IgM sind zusätzlich fünf solcher Grundeinheiten zu einem sehr großen Molekül zusammengefaßt. Eine strukturelle Darstellung der einzelnen Immunglobulinklassen ist in Abbildung 6 aufgeführt.

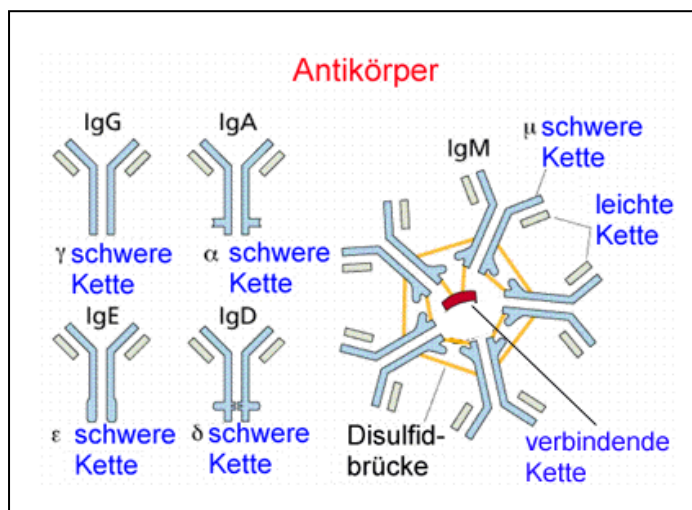


Abb. 6: Immunglobulintypen

Die strukturelle Auflistung der Immunglobulintypen A,D,E,G und M ist um die Spezifikationen der schweren Ketten, die den jeweiligen Typ bestimmen sowie um die Lage der IgM-Pentamer stabilisierenden Disulfidbrücken ergänzt.

Das **Immunglobulin G (IgG)** ist mit 80% die häufigste Antikörperklasse. Sie besitzt eine dimere Struktur. Das **Immunglobulin A (IgA)** hat ebenfalls eine dimere Struktur. Ca. 10-15% der Immunglobuline sind vom IgA-Typ. **IgM** (Immunglobulin M) kommt zu 5 -10% fast nur im Blut vor und hat eine typische, pentamere Struktur. Als Monomer (sIgM) ist er membranständiger "Oberflächenantikörper" der B-Lymphozyten. Dagegen werden vom **Immunglobulin Typ D (IgD)** nur wenige gebildet und seine Funktion ist unklar. Man findet ihn auf der Oberfläche von B-Lymphozyten. Möglicherweise wirkt er als Antigenrezeptor. Die **IgE**- Konzentration ist meist sehr gering, da IgEs meist an Rezeptoren auf **basophilen Zellen** und **Mastzellen** gebunden sind.

1.4.2 Generierung der Antikörpervielfalt

Die etwa 25.000 Gene, die die Antikörper im menschlichen Genom kodieren, sind wie in Abbildung 7 dargestellt, in der Keimbahn in Segmenten angeordnet. Allerdings werden etwa 10^{16} bis 10^{18} spezifische Antikörper theoretisch benötigt, um das pathogene Spektrum mögli-

cher Antigene abzudecken. Diese notwendige Vielfalt wird einerseits durch die **rekombinatorische Diversität** innerhalb der Antikörper bildenden Polypeptidketten erreicht. Der Ablauf einer solchen durch freie Rekombination von Immunglobulin-Gensegmenten induzierten somatischen Rekombination ist in Abbildung 8 dargestellt.

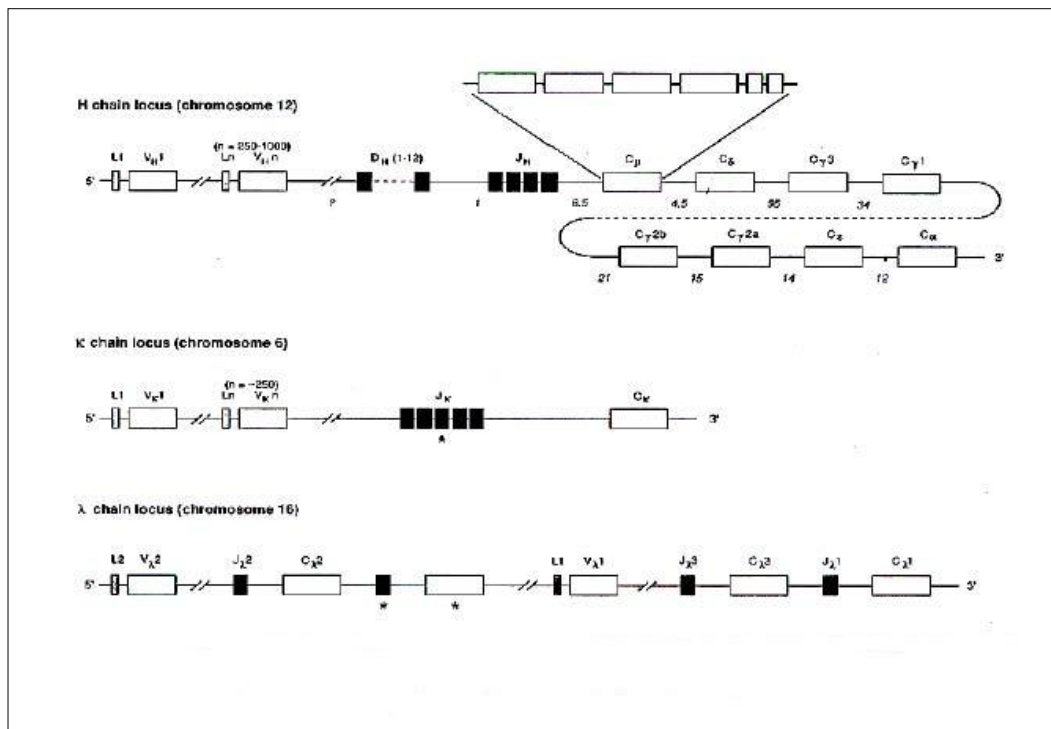


Abb. 7: Organisation der Maus-Ig-Gene in der Keimbahn

Exons und dazwischen liegende DNA-Segmente sind nicht vergrößert. Die kursiv geschriebenen Zahlen beziehen sich auf die ungefähren Längen der DNA-Segmente in Kilobasen (kb). Die mit Sternchen markierten Gene stellen nicht funktionale Pseudogene dar. Jedes Ch-Gen ist als eine einzelne Box dargestellt, die ihrerseits aus einigen Exons aufgebaut ist. Die aktuelle Exonzusammensetzung ist nur für die μ -Kette gezeigt. Die Gensegmente sind wie folgt gekennzeichnet: L, Leader; V, variable; J, verbindende; C, konstant

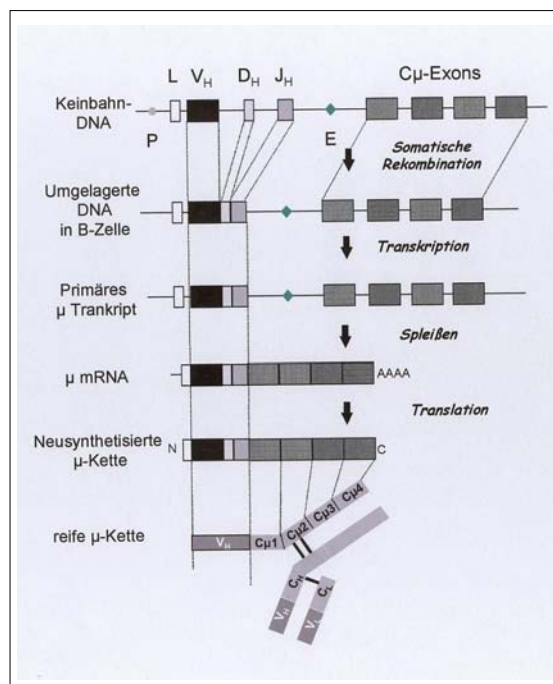


Abb. 8: Beispiel für eine somatische Rekombination

Dieses Schema zeigt die einzelnen Schritte innerhalb des Ablaufs einer somatischen Rekombination am Schwerkettenlocus.

Andererseits erfolgt eine freie Paarung einer H- mit einer L-Kette, so dass durch diese **kombinatorische Diversität** von schweren mit leichten Ketten verbunden mit der somatischen Rekombination etwa 10^7 verschiedene Antikörper gebildet werden können. Die Anzahl der spezifischen Antikörper wird aber auch durch die junktionale Variabilität (siehe Abbildung 9) der Struktursegmente weiter gesteigert.

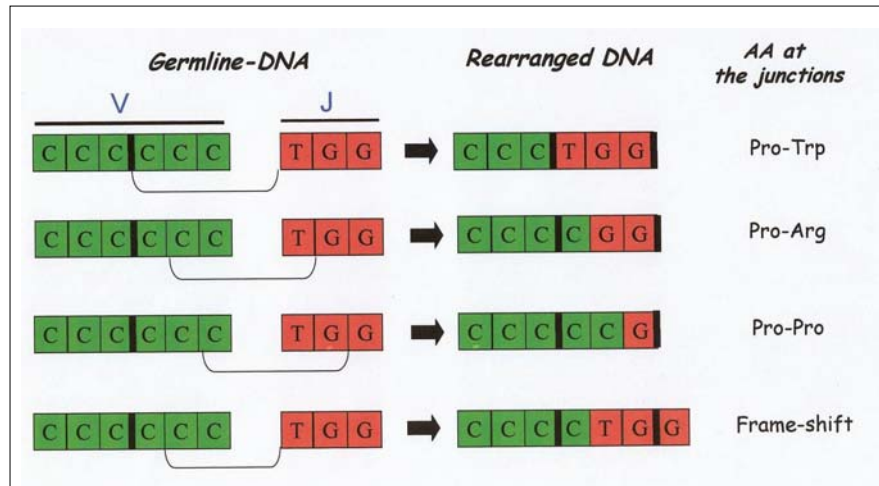


Abb. 9: Junktionale Diversität

Dargestellt ist die ungenaue Rekombination von Ig-Gensegmenten. Die grün markierten V-Basen-Segmente rekombinieren mit den rot dargestellten J-Basen-Segmente zur umgelagerten DNA, die dann die Aminosäuresequenz (AA) ergibt.

Neben der somatischen Hypermutation, bei der Punktmutationen in den Kodons auftreten, die zu einer Veränderung der Aminosäuresequenz führen, kann es auch zu Insertionen kommen. Selbst die D-Segmente können in allen 3 Leserastern benutzt werden. Auch Genkonversionen, d.h. nicht-reziproke homologe Rekombinationen sind möglich.

Allerdings entstehen auch bei dieser Vielzahl von Rekombinationsmöglichkeiten Nonsense-Codons und damit unbrauchbare Antikörper, d.h. ihre Antigenerkennung ist fehlerhaft. Um diese Immunglobuline zu eliminieren, werden die Antikörper selektioniert.

1.4.3 Antigenerkennung

Als **Antigen** wird alles bezeichnet, was eine adaptive Immunreaktion auslösen kann. Die chemische Zusammensetzung ist dabei von geringer Bedeutung. Antikörper erkennen **größere, dreidimensionale Oberflächenstrukturen der Antigene**. Oft hat ein Makromolekül mehrere solcher Strukturen, so genannte **antigene Determinanten oder Epitope**, die unabhängig voneinander die Bildung von Antikörpern auslösen. Umgekehrt können zwei in der Primärstruktur ganz unterschiedliche Moleküle trotzdem vom selben Antikörper erkannt werden, wenn ihre Oberflächenstruktur zufällig sehr ähnlich ist. Man spricht dann von einer Kreuzreaktion. T-Lymphozyten sind nicht auf dreidimensionale Epitope, sondern auf lineare Peptide von 8 bis 20 Aminosäuren Länge spezialisiert.

1.4.4 Antikörperselektion

Zellklone, die Antikörper produzieren, die ubiquitäre körpereigene Antigene erkennen, werden in einem frühen Entwicklungsstadium abgetötet (klonale Deletion) oder in ein Stadium gebracht, in dem sie zwar bestehen bleiben, aber nicht mehr reagieren können (klonale Anergie). Da diese Schutzmechanismen nicht perfekt sind, ist die **klonale Selektion** notwendig. Dabei sucht sich das Antigen die B-Zelle mit dem passenden Antikörper auf der Oberflä-

che aus und regt diese zur Bildung von Tochterzellen an, die alle denselben "nützlichen" Antikörper bilden. Die Proliferation startet aber erst dann, wenn sie durch ein unspezifisches Signal, das von einer T-Zelle (oder über eine Antigenpräsentierende Zelle) abgegeben wird, freigegeben wurde. Mit dieser T-Zell-Hilfe ist gewährleistet, dass keine Autoimmunreaktionen im intakten Körper stattfinden.

1.5. Major histocompatibility complex (MHC)

Die nicht adaptive Immunabwehr gegen exogene Pathogene ist oftmals erfolglos, weil diese in Vesikeln oder im Zytosol persistieren können. TH1-Zellen verstärken dabei die Fähigkeit der Makrophagen, vesikuläre Pathogene abzutöten, während TH2-Zellen B-Zellen zur Antikörperproduktion stimulieren. Damit aber CD4-TH1-Zellen und –TH2-Zellen exogene Antigene erkennen und auf diese antworten können, benutzen sie ihren T-Zellrezeptor, um die an den Klasse II MHC-Proteine gebundenen antigenischen Peptide zu sehen, die ihr von den Antigen-präsentierenden Zellen (APC, siehe 1.6) präsentiert werden. Entsprechend den TH-Zellen können zytotoxische T-Zellen nur durch endogene Antigenpeptide aktiviert werden, die gebunden an Klasse I MHC-Proteine der infekten Zellen präsentiert werden.

Die MHC Antigen-präsentierenden Zellen sind mit „human leucocyte antigen“ (HLA)- System beim Menschen identisch und werden in den Haupt-Histokompatibilitätsgenen (MHC-Gene) kodiert. Klasse 1- MHC-Proteine werden von fast allen kernhaltigen Zellen besonders von Leukozyten exprimiert. MHC-Klasse-II-Moleküle findet man immer auf B-Lymphozyten, dendritischen Zellen und Endothelzellen und können in menschlichen T-Lymphozyten und Makrophagen induziert werden.

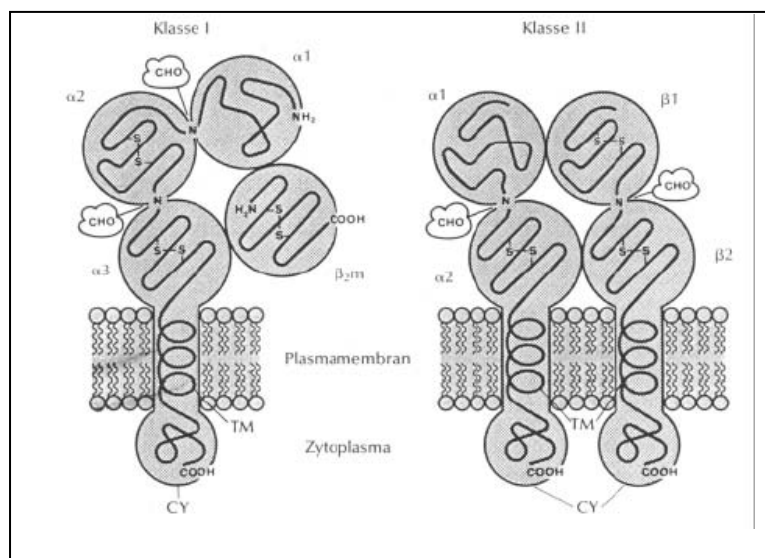


Abb. 10: MHC-Proteine der Klassen 1 und 2

Die MHC-Klasse 1 Proteine sind als β_2 -Mikroglobulin-Komplex, die MHC-Klasse 2- Proteine als nicht kovalent gebundener Strukturkomplex gezeigt.

Klasse 1-MHC ist ein Heterodimer mit einer membran-gebundenen α -Kette und einem nicht kovalent assoziierten β_2 -Mikroglobulin. Sowohl die α -Kette als auch das β_2 -Mikroglobulin gehören der Ig-Superfamilie an. Die α -Kette besteht aus 3 Domänen, wie Abbildung 10 zeigt, wobei das Antigen an die $\alpha1$ - und die $\alpha2$ -Domäne bindet und die $\alpha3$ -Kette als Membrananker sowie als Bindungsstelle für CD8 fungiert. Über die $\alpha1$ - und die $\alpha2$ -Domäne erfolgt auch die Bindung an den TZR. Der TZR unterscheidet unterschiedliche MHC-Klasse I-gebundene Peptide durch ihre Konformation und durch die Klasse I -Konformation, die durch ihre Bindung induziert wird.

MHC Moleküle sind in Abwesenheit eines gebundenen Peptids instabil und werden dann erst um die Peptide gefaltet, bevor sie an die Plasmamembran transportiert werden.

Klasse II-MHC ist ein nicht kovalent gebundener Heterodimer, bestehend aus α - und β -Ketten. Die Peptidantigene mit einer Länge von 13-18 Resten binden an die α 1- und β 1-Domänen (vgl. Abb. 10), an die auch die TZR binden. Die membrangebundenen α 2- und β 2-Domänen sind invariant. CD4 bindet an die α 2-Domäne und verstärken so den Kontakt zwischen der T-Zelle und den APCs. Entsprechendes gilt auch für CD8.

Allerdings verlieren CD4-T-Zellen ihre Reaktionsbereitschaft, wenn sie Peptidantigene erkennen, die von anderen T-Zellen, die MHC II-Moleküle exprimieren, präsentiert werden.

Die Gruppe der MHC-Moleküle, die es den T-Zellen ermöglicht, zwischen körpereigenen und körperfremden Antigenen zu unterscheiden, enthält noch eine weitere Klasse, die HLA Klasse III, auch HLA III genannt. Diese Klasse kodiert für verschiedene Komplementkomponenten (vgl. 1.2.1), Interleukine und andere Moleküle, die für Peptidpräsentationen wichtig sind und werden deshalb nicht zu den MHC dazugezählt.

1.6 Antigenpräsentierende Zelle

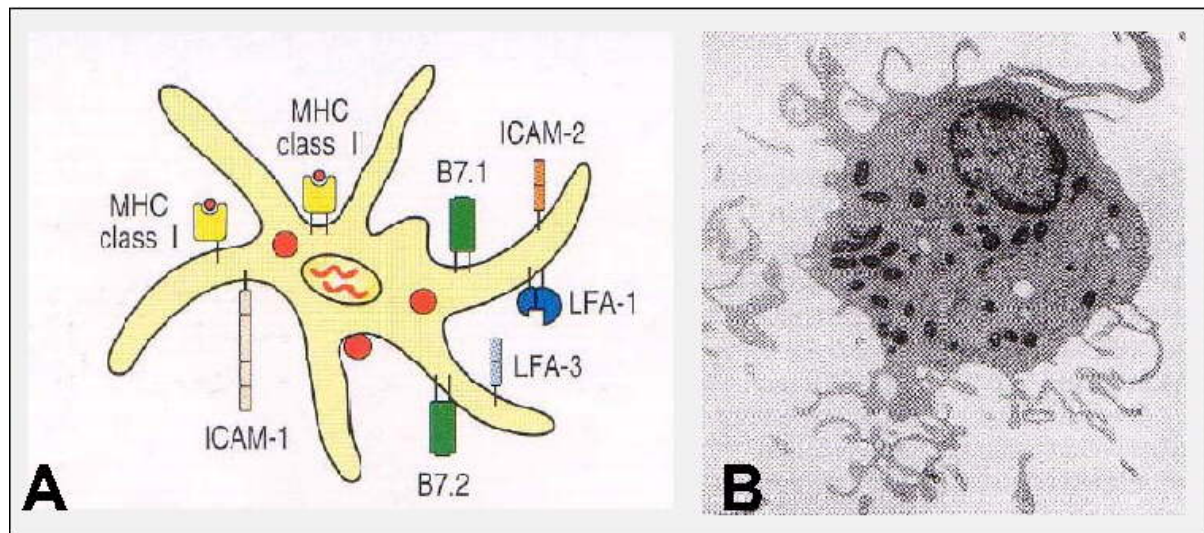


Abb. 11: Dendritische Zelle mit APC-Funktionen

Gezeigt ist in A der schematische Aufbau der dendritischen Zelle mit den MHC-Komplexen. In B ist eine Aufnahme einer dendritischen Zelle dargestellt.

B-Zellen und einige T-Zellen, insbesondere T-Helferzellen, sind nicht in der Lage, freies Antigen zu erkennen. Um erkannt zu werden, muss es zusammen mit MHC-Produkten der Klasse 2 präsentiert werden. Diese Funktion wird von Zellen übernommen, die antigenpräsentierende Zellen (APC) genannt werden.

Einige Zellen der Monozyten-Makrophagenreihe exprimieren Glykopeptide (MHC) der Klasse 2 und können als APC für die T- oder B-Zellen fungieren. Andere Zelltypen z.B. Zellen des Gefäßendothels, können ebenfalls APC-Funktionen übernehmen. Dendritische Zellen, die sich im Blut, Lymphe und anderen Geweben befinden, sind nicht phagozytierende, Fc-Rezeptor-negative Zellen, die in einem grossen Ausmass MHC-Komplexe der Klasse 2 exprimieren. Sie sind wie Abbildung 11 zeigt besonders reich an unspezifischen Rezeptoren zur Antigenaufnahme sowie reich an Rezeptoren/Kostimulatoren zur Präsentation bakterieller und viraler Antigene für die Stimulation von B-Zellen (über B-Zellrezeptor) und Aktivierung/Stimulation zytotoxischer T-Zellen bzw. TH2-Zellen (siehe Abbildung 12, 2).

Neben den B-Zellen sind auch die Langerhanszellen in der Haut und interdigitierende Follikelzellen (Thymus) Zellen, die APC-Funktionen übernehmen können.

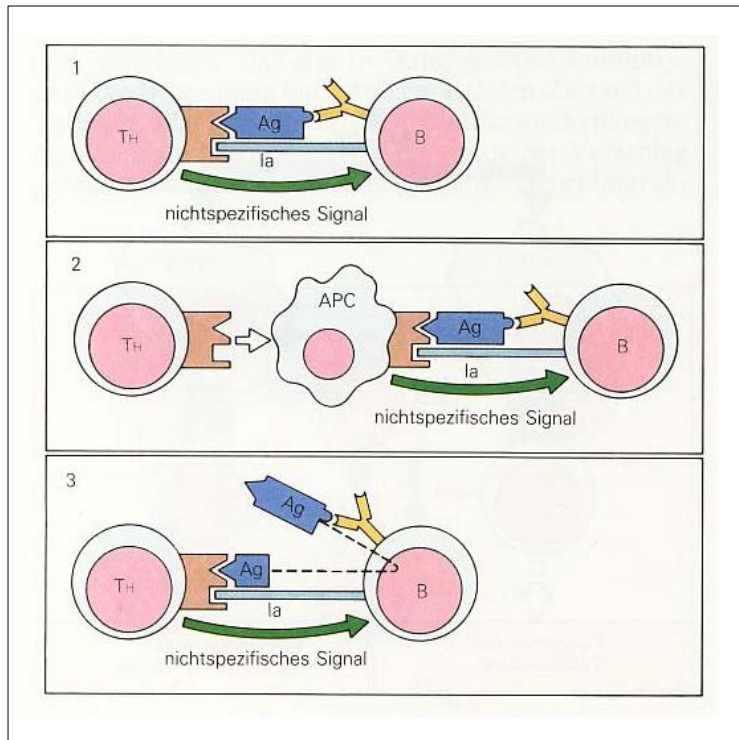


Abb.12: Drei mögliche Mechanismen für die Kooperation zwischen T- und B-Zellen
 Abgebildet ist je eine T-Helfer-Zelle, die auf das Antigen in Verbindung mit einem MHC-Molekül (1a) auf einer antigen-präsentierenden Zelle sensibilisiert ist. Eine TH-Zelle hat verschiedene Möglichkeiten zur B-Zell-Hilfe, indem sie direkt das Antigen als Brücke benutzt (1) oder ihre Rezeptoren als „Helferfaktoren“ freisetzt, die sich an die APC anlagern und eine Brücke zur B-Zelle bilden (2). Eine dritte Möglichkeit ist die, dass die B-Zelle sich das Antigen einverleibt und einen Teil davon zusammen mit dem Ia-Molekül wieder exprimiert (3). In jedem Fall müsste die B-Zelle ein nichtspezifisches Signal empfangen, welches sie auch an andere B-Zellen weitergibt.

1.7 Natürliche Killer-Zellen (NK-Zellen)

Ca. 15 % der peripheren Blut-Lymphozyten sind weder T noch B-Zellen sondern zytotoxische Lymphozyten, die natural killer cells oder NK. Charakteristisch ist für sie, daß sie tumoröse, virusinfizierte und manchmal auch normale Zellen ohne vorherige Kompromitierung ("sensitization") zerstören können.

Sie entstehen aus den gleichen Vorläufern wie T-Zellen im Knochenmark, werden aber nicht im Thymus weiterentwickelt. Hauptsächlich befinden sie sich in den sekundären Lymphorganen, Blut und Milz, nie in der Lymphe der thorakalen Organe oder im Thymus.

Da NKs keine T- und B-Zell-Rezeptoren besitzen, brauchen sie andere Erkennungsmerkmale, um an Zellen oder Erreger anzudocken. Sie werden durch Interferon γ (INF γ) stimuliert.

1.8 Zielsetzung der Simulation

Das Auftreten von „Pathogenen“ wie Viren, Würmer usw. sind heute nicht nur auf den medizinisch-biologischen Bereich beschränkt, sondern stellen auch die Informatik vor immer grösser werdende Probleme. Aufgrund der „schematischen“ Ähnlichkeit der Wirkungsweisen der Pathogene bestand nun die Idee, Immunsysteme als informatische Sicherheitssystem zu implementieren, um damit der Infektion der Software eines Computersystems entgegenzuwirken.

Einen ersten Schritt auf dem Weg zur Generierung eines solchen System wollen S.H. Kleinstein und P.E. Seiden tun, indem sie sich das Ziel gesetzt haben, den modularen Aufbau der körpereigenen Immunabwehr ohne Einsatz von Differentialgleichungen am Beispiel des adaptiven Immunsystem per Computer zu simulieren.

2. Das Computersimulationsmodell IMMSIM

Das Computermodell IMMSIM, IMMSIM steht für *immune simulation*, orientiert sich an dem schematischen Aufbau des adaptiven Immunsystems. Im Simulationsmodell werden folgende Bestandteile des adaptiven Immunsystems und deren Wirkungsweise berücksichtigt: **B-Zellen, T-Zellen, Antigenpräsentierende Zellen (APC), Antikörper, Antigene und Antigen-Antikörper-Komplexe**. Allerdings erfolgen die Interaktionen, ausgeübt von den Oberflächenrezeptoren und MHC-Komplexen, zwischen den einzelnen Teilen nicht über dreidimensionale Strukturen, sondern über Bitstrings, bestehend aus Nullen und Einsen. Dies bedeutet, dass beispielsweise zelluläre Oberflächenrezeptoren oder MHC-Komplexe durch n -Bitstrings in der Simulation dargestellt werden und damit jeweils 2^n spezifische Oberflächen-Bindungsmoleküle (z.B. Rezeptoren oder MHC usw.) generiert werden können.

2.1. Aufbau und Eigenschaften der zellulären Bestandteile des Modell

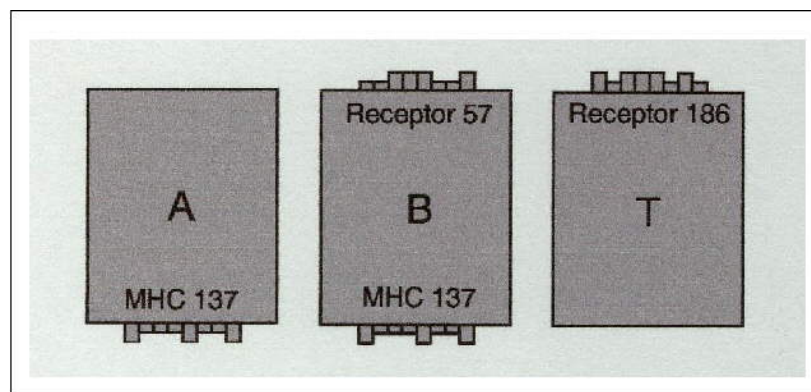


Abb. 13: APC-, B- und T-Zellmodelle

Gezeigt sind die Schemata einer Antigen-präsentierenden Zelle (A) mit einem MHC-Komplex, einer B-Zelle (B) mit einem MHC-Komplex und einem 8-Bit-Rezeptor und sowie einer T-Zelle (T) mit einem 8-Bit-Rezeptor. Die ganzzahligen Namen der 8-Bit-Segmente stellen die dezimalen Werte des binären Strings dar. MHC steht für den Haupthistokompatibilitätskomplex und ist hier 8-Bit lang.

Jede **B-Zelle** besitzt einen exponierten Antigen-spezifischen Rezeptor als n -Bit String. Damit können 2^n spezifische B-Zellen, die Antikörper herstellen und prozessieren, in der Simulation implementiert werden. Ferner verfügt jede B-Zelle über einen MHC-Komplex, simuliert durch einen n -Bit-String. **T-Zellen** besitzen einen T-Zellrezeptor, der wie die B-Zellen durch einen n -Bit String repräsentiert wird. Dieser bindet an die binären MHC-Komplexe der B-Zellen und **APCs**, die ihrerseits über einen eigenen aus einem n -Bit-String aufgebauten MHC-Komplex verfügen (siehe Abbildung 13).

Was die Verhaltensweisen der einzelnen Zelltypen anbelangt, so werden sie im Modell nicht verändert. Eine B-Zelle, die als „naive“ Zelle gerade das Knochenmark verlassen hat, kann ein Antigen binden und dessen antigenische Peptide präsentieren, d.h. die B-Zelle wird durch das Antigen und den auf der Oberfläche positionierten MHC-Peptidkomplex charakterisiert. Auch die anschließende Stimulation durch T-Zellen, an die sich die Zellteilungsphase mit der Differenzierung in Plasmazellen und Gedächtniszellen anschließt, wird durchgeführt. Während Plasmazellen mit der Antikörperproduktion beginnen, sind Gedächtniszellen in der Lage, diesen Vorgang neu zu starten.

Auch die Selektionsprozesse, durch die Thrombozyten zu T-Helfer-Zellen differenziert werden, erfolgen durch thymische Interaktionen während der Modellthymuspassage. So werden T-Zellen in der negativen Selektion dem MHC-Eigenpeptid-Komplex u. U. viele Male ausgesetzt und im Verhältnis zur Bindungsstärke eliminiert. Je grösser die Anzahl der Tests ist, umso effizienter ist die Eliminierung der autoreaktiven T-Zellen. In der positiven Selektion

wird dafür gesorgt, dass nur potentiell MHC bindungsfähige T-Zellen den Thymus verlassen, d.h. wenn bestimmte Mindestanforderungen an diese Bindung erfüllt sind. So müssen beispielsweise bei einem 8-Bit-MHC-String 7 matchende Bits vorkommen, damit mindestens 3 Bits für MHC-Peptid-Bindung vorhanden sind. Damit hängt die Zahl der überlebenden T-Zellen von der Zahl und dem Charakter der verfügbaren MHCs und Eigenpeptide ab.

Bezüglich der Lebensdauer verhalten sich die Zellen im Modell wie die im realen Immunsystem. So werden zusätzlich zum klonalen Wachstum neue, undifferenzierte („naive“) Zellen produziert. Allerdings sterben Zellen auch nach einer vorgegebenen Halbwertszeit.

2.2 Repräsentation der Antikörper, Antigene und Antigen-Antikörperkomplexe im Modell

Antigene werden durch eine Serie von hintereinander angeordneten Bitstrings im Modell dargestellt. Einige dieser Bitstrings stellen **Epitope**, andere **Peptide** dar (siehe Abb. 14).

Als **Epitop** wird der Teil des Antigens bezeichnet, der durch einen B-Zellrezeptor erkannt wird. Dagegen ist ein **Peptid** der Antigenteil, der durch ein MHC-Molekül gebunden werden kann und durch eine geeignete T-Zelle erkannt wird. Epitope und Peptide sind spezifisch getrennt, da ein Epitop zu einer vollständig gefalteten dreidimensionalen Struktur korrespondiert und Peptide als Teile eines verdauten Antigens nicht notwendigerweise eine einfache Beziehung zu einem Epitop haben.

Antikörper besitzen den gleichen Rezeptor wie die B-Zelle, von der sie abstammten. Allerdings können sie auch, wenn in der Simulation gewünscht, andere Rezeptoren und Peptide besitzen. Damit erhöht sich u. U. die Anzahl der spezifischen Antikörper, d.h. das Repertoire der verschiedenen Antikörper wird grösser und damit die Zahl der zu verwendeten Bits für Rezeptor, Peptid, Epitop, Antikörper oder MHC-Komplex.

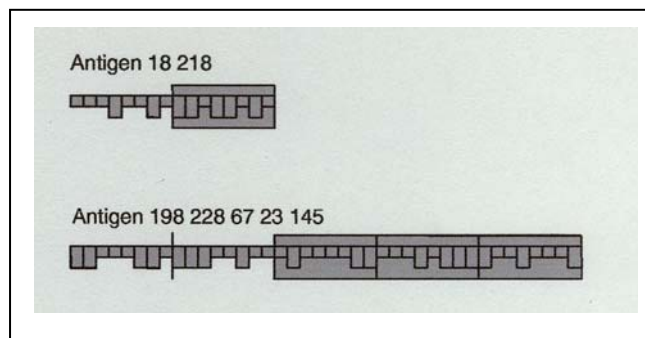


Abb.14: Beispiele für Antigene

Dargestellt sind Beispiele für Antigene in einem 8-Bit-System. Die Epitope sind nicht wie die Peptide eingeschachtelt dargestellt. Die ganzzahligen Namen der 8-Bit-Segmente sind die dezimalen Werte der binären Strings

Antigen-Antikörper-Komplexe werden durch die Bindung des Antikörpers an das Antigen über Epitope dargestellt. Im Modell stellen Antigen-Antikörperkomplexe verklumpte Moleküle dar, die nur durch ihre Epitope und Peptide charakterisiert sind. Sie haben keinen inneren Zustand.

2.3 Interaktionen, Komplementarität und Affinität

Damit eine Bindung zwischen einem Rezeptor, repräsentiert durch einen Bitstring und einem binären Epitop beispielsweise entstehen kann, müssen beide Bitstrings miteinander interagieren. Voraussetzung dafür ist in dieser Simulation, dass beide Bitstrings zueinander komplementär sein, d.h. 0 korrespondiert mit 1 und umgekehrt. Denn es gilt: je grösser die Anzahl der Bits ist, die in beiden Strings komplementär sind, umso stärker werden die beiden

Strings miteinander interagieren. Das Mass für die Stärke der Bindungsaffinität, d.h. der Interaktionen ist durch die Gesamtheit aller p-Bit-Matches gegeben, wobei ein **p-Bit-Match** als Paar definiert wird, für das exakt p Bits in beiden Strings komplementär sind. Zusätzlich wird in diesem Modell ein Minimum für p definiert, unter dem keine Interaktionen mehr möglich sind.

Ansonsten werden in diesem Modell alle möglichen Interaktionen betrachtet. Sollte jedoch der Fall eintreten, dass eine Zelle, Antikörper usw. mehr als eine erfolgreiche Bindung gleichzeitig eingehen kann, dann wird die aktuelle Interaktion zufällig bestimmt. Erst wenn alle Interaktionen entschieden sind, werden neue Zellen produziert.

2.4 Ablauf einer Simulation am Beispiel der B-Zell-Aktivierung

Über eine probabilistische Bindung der B-Zelle über seinen Rezeptor an die komplementären Antigenepitope wird das Antigen an die B-Zelle gebunden. Diese Bindung ist von der Zahl der gematchten Bits abhängig. Denn je grösser die Zahl der matchenden Bits, umso grösser ist die Interaktionswahrscheinlichkeit.

Nach der Endozytose des gebundenen Antigens, wie in Abbildung 15 gezeigt, werden die antigenen Peptide an MHC-Moleküle gebunden und auf der B-Zelloberfläche exponiert, was dem 1.Signal der Aktivierung entspricht.

Die MHC-Peptid-Bindung erfolgt der Art, dass der MHC-Bitstring in der Hälfte geteilt wird. Die linke Hälfte ist der MHC-Teil, der den T-Zellrezeptor bindet, und die rechte Hälfte der, der die Grube repräsentiert, in welcher das Peptid bindet. Wenn es nur einen MHC-Komplex und ein Peptid gibt, kann entweder die linke oder die rechte Hälfte des Peptids binden. Bei der Bindung des MHC-Bitstrings mit einer Hälfte des Peptids wird die andere Peptidhälfte den T-Zellen präsentiert.

Wenn nun der T-Zellrezeptor an den MHC-Peptid-Komplex bindet, wobei diese Bindung den gleichen Regeln unterliegt wie die, mit der ein B-Zellrezeptor mit dem antigenischen Epitop interagiert, dann wird das 2.Signal der Aktivierung auf die B-Zelle übertragen und T- und B-Zelle beginnen mit der Zellteilung und etablieren Klone ihres eigenen Typs.

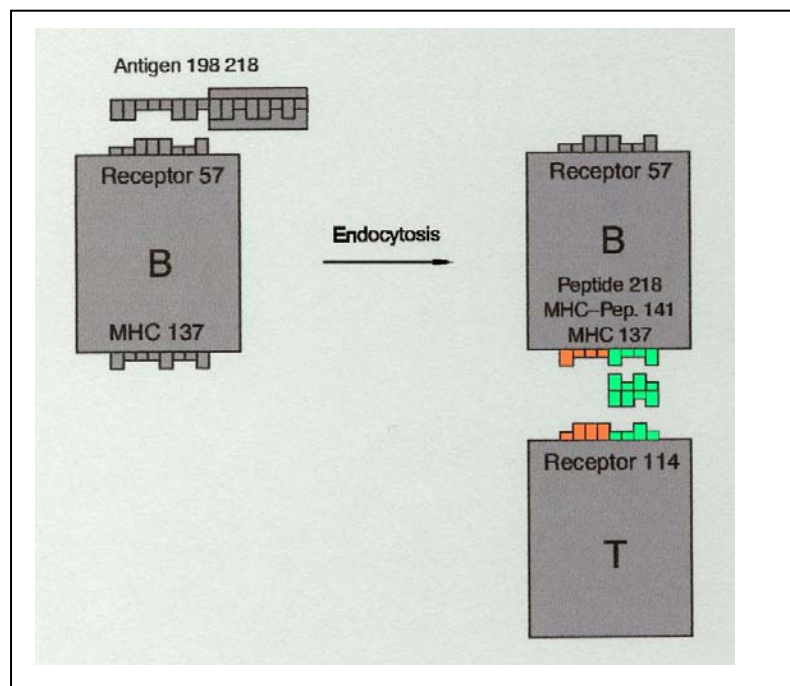


Abb. 15: Schritte in der Erkennung und Prozessierung eines Antigens durch B- und T-Zellen

Das Schema zeigt den Ablauf der B-Zellaktivierung, die die Erkennung und Prozessierung des Antigens einschliesst. Das Peptid ist grün, während der MHC-Komplex orange eingefärbt ist.

Dabei teilen sich T-Zellen dreimal, wobei 2^3 T-Zellen entstehen. B-Zellen teilen sich viermal. Von den insgesamt 16 neuen undifferenzierten („naiven“) B-Zellen differenzieren 8 zu Gedächtniszellen und die restlichen 8 B-Zellen zu Plasmazellen, die die B-Zell-spezifischen Antikörper produzieren. Sich teilende B-Zellen durchlaufen auch Hypermutationen durch zufälligen Einbau von 1-Bit-Austauschen in ihren spezifischen Bitstringrezeptoren.

Antigenpräsentierende Zellen (APC) können bei dieser Aktivierung als Konkurrenten um die T-Zellbindung mit den B-Zellen auftreten, obwohl sie Antigene nur unspezifisch binden, prozessieren und die antigenen Peptide präsentieren. Denn in diesem Modell teilen sich nur T-Zellen, wenn sie eine T-Zellbindung mit dem MHC-Peptid-Komplex, der sich auf der APC befindet, über ihren Bitstringrezeptor herstellen.

2.5 Modellierung der lymphatischen Organe

Wie in 1.1.1 bereits vorgestellt, findet die Bildung von Lymphozyten und die Immunabwehr in den lymphatischen Organen statt. Um die Organe beispielsweise den Thymus strukturell in das Modell zu integrieren, verwendet die Simulation einen modifizierten zellulären Automaten.

Ein zellulärer Automat ist ein dynamisches System, in dem Raum und Zeit diskret sind. Jede Stelle nimmt eine begrenzte Serie möglicher Werte an. Die Entwicklung der Werte an jeder Stelle erfolgt nach den gleichen deterministischen Regeln, wobei die Regeln für die Entwicklung einer Stelle von einem lokalen Nachbarn abhängen, der diese Stelle umgibt. Im Gegensatz dazu entwickeln sich die Werte im IMMSIM-Modell probabilistisch. Ausserdem wird aus der lokalen Nachbarschaft der ursprünglichen Stelle eine Stelle ausgewählt, die dann die ursprüngliche Stelle selbst annimmt. Dies bedeutet, dass den einzelnen Bestandteilen erlaubt wird, sich von Stelle zu Stelle zu bewegen. Letztlich können alle Bestandteile des Modells in die sie umgebenden Stellen diffundieren, was **einem Zeitschritt** entspricht.

Das in diesem Modell gewählte Gitter ist dreieckig (siehe Abb. 16), das einen kleinen Teil eines Körpers beispielsweise des Thymus darstellen soll. Jede Stelle kann mit einer Zahl verschiedener Bestandteile einiger Typen bevölkert werden. Allerdings wird dabei darauf geachtet, dass beispielsweise naive B- und T-Zellen zufällig mit kompletter Diversität ($2^8 = 256$ Typen für 8 Bits) gewählt werden.

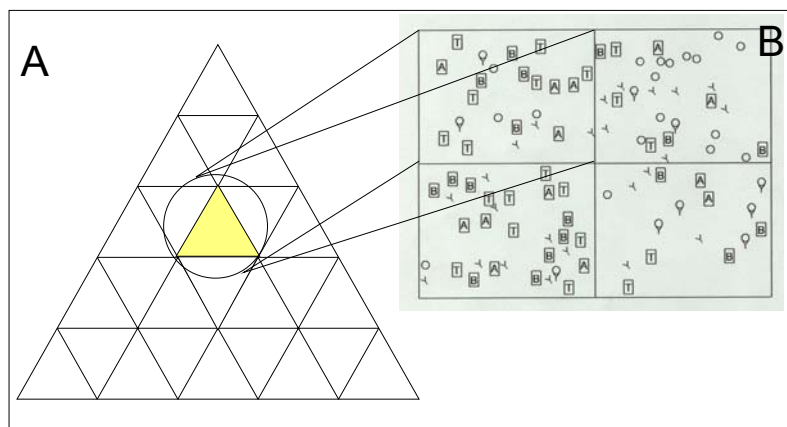


Abb. 16: Darstellung des zellulären Automaten

Zeichnung von 4 Stellen in dem Automaten mit einem dreieckigen Gitter. Die jeweilige korrespondierende Gitterstelle ist farbig hervorgehoben. Gezeigt ist die rechteckige Anordnung der normalerweise dreieckigen Gitterstellen. A, B und T bezeichnen APCs, B-Zellen und T-Zellen. Die Kreise sind Antigene und die Ys Antikörper. Die Kreise, die an Antikörper gebunden sind, stellen Antigen-Antikörper-Komplexe dar. Zu irgendeinem Zeitschritt können nur die Bestandteile in der gleichen Stelle interagieren.

2.6. Die Implementierung

Die Algorithmen und Datenstrukturen, die in dieser Version verwendet werden, sind ähnlich, aber nicht identisch mit der kompletten Version. Aus Effizienzgründen werden in dieser Version nur B-Zellen, Antikörper und Antigene als Datenstrukturen verwendet. Erlaubt ist ausserdem, dass B-Zellen Antigene binden und Antikörper mit Antigenen interagieren.

2.6.1 Die Datenstrukturen

Alle Datenstrukturen sind Arrays mit wenigstens 2 Dimensionen für die Positionsbeschreibung im Gitter.

Das Array für die **Antigen-Speicherung** hat 3 Dimensionen, wobei jedes Arrayelement ein Integer ist und der Antigenkonzentration an dem Gitterpunkt, festgelegt durch 2 der 3 Arrayindizes, entspricht. Die dritte Dimension steht für die Art der Diffusion. Strukturell wird dabei angenommen, dass es nur ein einziges Antigen gibt, dass nicht mutiert. Dies ist für die Strukturausdehnung auf das Modell eines mutierenden Virus (HIV) wichtig.

Das **Antikörper-Array** besitzt neben den 3 Dimensionen, die bereits bei der Antigenspeicherung eingeführt wurden, eine Extra-Dimension im Bereich von 0 bis n, wobei n die Bit-Länge ist. Die Arrayelemente sind Integers, die die Zahl der Antikörper an der korrespondierenden Gitterstelle und die Zahl der matchenden Bits mit Bezug auf das Antigen, das durch den Index des Extraarrays gegeben ist, darstellen. Nur die Zahl der matchenden Bits mit dem Antigen ist notwendig, da sie die Stärke der Interaktion bestimmt. Deshalb kann der Speicherbedarf und die Rechenzeit durch einfaches Zusammenfassen der Antikörper mit der gleichen Zahl an matchenden Bits verringert werden.

Das Array für die **B-Zellen** hat wie die Antigene 3 Dimensionen. Jedoch stellen die Arrayelemente in diesem Fall verbundene Listen dar, die die individuellen B-Zellen enthalten, die sich aktuell an der Gitterstelle befinden. Jede B-Zelle wird individuell gespeichert. Die B-Zell-Struktur definiert die Information über den Bit-String-Antikörperrezeptor, den B-Zelltyp, die B-Zellteilung und die Zahl der matchenden Bits mit dem Antigen, die mit jeder B-Zelle gespeichert wird.

2.6.2 Initialisierung der Simulation

```
for (simTime = 0; simTime < 200; simTime++)
{
    birth();          /* B cell proliferation */
    produceAb();      /* Plasma B cell antibody production */
    iB_Ag();          /* B cell + Antigen interaction */
    iAb_Ag();         /* Antibody + Antigen interaction */
    death();          /* Cell death */
    flow();           /* Flow of naive B cells from bone marrow */
    diffusion();      /* Diffusion within the body grid */

    ...               /* Check for injections of antigen */
    ...               /* Output data */
}
```

Abb. 17: Die Hauptschleife des IMMSIM tutorial-Programms

Die Schleife beinhaltet die Hauptmethoden der Simulation IMMSIM und umfasst die B-Zell-Proliferation, die Antikörperproduktion, des Zelltods, der Diffusion, die Nachproduktion von B_Zellen, die Interaktionen zwischen B-Zelle und Antigen bzw. Antigen und Antikörper, die Diffusion, die Antigeninjektionen sowie die Ausgabedaten.

Nach der zufälligen Bevölkering der Stellen mit der gewünschten Zahl von B-Zellen im Rahmen der Initialisierung startet die Hauptsimulationsfunktion, die eine Schleife enthält (siehe Abb. 17) und die Dynamiken bei jedem Schritt der Simulation aktualisiert. Dabei erfolgt auch die Überprüfung der festgelegten Antigeninjektionen in der Simulation. Die Dynamiken eines jeden simulierten Prozesses hängen von der Anzahl der Parameter, die im Code durch einen vorangestellten Unterstrich gekennzeichnet sind und sich am Anfang des Codes befinden, ab. Sie werden durch den Benutzer gesetzt. Zur Erleichterung liegen sie im Programmcode bereits als Konstanten vordefiniert vor.

2.6.3 Die Methoden der Hauptschleife

2.6.3.1 Die Methode `birth()`

Die Funktion `birth()` bestimmt die Dynamik der B-Zellteilung. Eine B-Zelle, stimuliert durch eine Antigenbindung, teilt sich in geringen Umfang. Bei der Produktion der neuen B-Zelle (Methode `createB()`) wird dieser der Oberflächenbitstring-Rezeptor und der Antigen-Bit-String für das Matchen in der Methode `match()` übergeben.

Die genaue Zahl der Teilungen wird durch den Simulationsparameter `_divideB` bestimmt. Während eines Zeitschritts darf sich eine B-Zelle nur einmal teilen, was die Zeitskala der Simulation definiert. Ferner sind B-Zellen über die Funktion `countB()`, Antigene über `countAg()` und Antikörper über `countAg()`, initialisiert in der `main`-Methode versehen, mit einem spezifischen Zähler verknüpft, um deren jeweilige Konzentration zu ermitteln. Während der Zellteilung gehen B-Zellen keine anderen Antigeninteraktionen ein. Nach Ende der Teilungsphase entscheidet die B-Zelle, ob sie eine Plasma- oder eine Gedächtniszelle wird, d.h. vorerst im Zustand einer naiven B-Zelle verbleibt. Die Wahl wird durch den Parameter `_pmRatio` kontrolliert, der die Wahrscheinlichkeit angibt, mit der eine naive B-Zelle eine Plasmazelle wird.

2.6.3.2 Die Methode `produceAb()`

Plasma B-Zellen produzieren kontinuierlich Antikörper. Für jede existierende Plasma-Zelle fügt die Methode `produceAb()` `_numAb` Antikörper zu den Antikörpern an der gleichen Gitterstelle hinzu, sobald diese von den Plasmazellen produziert wurden.

2.6.3.3 Die Methode `iB_Ag()`

B-Zellen werden durch die Bindung an Antigene stimuliert. Jede B-Zelle durchläuft eine Interaktion mit einem Antigen während eines jeden Schritts. Die Methode `iB_Ag()` betrachtet alle Paare von B-Zellen, mittels der Methode `randomizeB()` randomisiert, und Antigenen (Ag), die sich an der gleichen Gitterstelle befinden und gibt jedem eine Chance zu interagieren, wenn die B-Zelle keine Plasmazelle ist, sich nicht teilt und zwischen dem Antigen und der B-Zelle eine genügend grosse Interaktionswahrscheinlichkeit aufbaut. Die Interaktionswahrscheinlichkeit zwischen einer B-Zelle und einem Antigen mit einem p -Bit-Match, wobei ein Match zwischen 2 Bitsträngen die Hamming-Distanz ist, ist durch `IS[p]` gegeben und entspricht der biologisch gemessenen Affinität.

Wenn sich eine Interaktion ereignet, wird das Antigen aus der Simulation entfernt und die B-Zelle gegen weitere Interaktionen in diesem Zeitschritt geschützt.

`IS[]` ist ein Array der Länge `_bits+1`. Jede Komponente des Arrays gibt die Interaktionswahrscheinlichkeit für einen p -bit-Match an, wobei p in der Bandbreite von 0 bis `_bits` liegt. Für $p < \text{_minMatch}$ ist `IS[p] = 0`, d.h. `_minMatch` steht für die minimale Zahl an

matchenden Bits, die für die Bindung erforderlich sind. Die minimale Affinität ist durch $IS[_minMatch] = _affLevel$ gegeben. Die Zunahme der Affinität, die bei der Vergrößerung des Matches um ein Bit entsteht, ist in Abbildung 18 definiert.

$$\frac{IS[p+1]}{IS[p]} = _affEnhance \times \frac{\binom{_bits}{p}}{\binom{_bits}{p+1}} \quad \text{mit} \quad \binom{a}{b} = \frac{a!}{b!(a-b)!}$$

Abb. 18: Definition der Zunahme der Affinitätsstärke

Die Gleichung definiert die Veränderung der Affinitätsstärke in Korrelation zu der Veränderung eines Matches um ein Bit als Produkt des Affinitätsfaktors für ein zusätzliches matchendes Bit und dem Quotienten der Binomialkoeffizienten, die die Inverse der Typenrate darstellen.

Die Verteilung der B-Zellrezeptorenanzahl mit einer gegebenen Affinität für ein bestimmtes Antigen folgt der binomischen Verteilung. Es gibt perfekte Matches, 1-Bit-Mismatches usw. Die Definition der Gesamtzahl der B-Zellrezeptortypen, die ein bestimmtes Antigen stimulieren kann, ist in Abbildung 19 gezeigt.

$$\sum_{b=_minmatch}^{_bits} \binom{_bits}{b}$$

Abb. 19: Definition der Gesamtzahl der B-Zellrezeptortypen

Die Summe definiert die Gesamtzahl der binomisch verteilten B-Zellrezeptortypen, wobei $_bits$ die Länge des Oberflächenrezeptorbits angibt und $_minmatch$ für die minimale Anzahl der matchenden Bits, die für eine Bindung notwendig sind, steht.

Wegen der binomischen Verteilung sollte aber $_minMatch$ immer grösser als $_bits/2$ gewählt werden, da $IS[p]$ im Allgemeinen, im Gegensatz zur biologischen Vorgabe, sich abschwächt, weil die binomische Verteilung an ihren Enden über Minima verfügt. Eine zwischenzeitliche Erholung ist dabei nicht ausgeschlossen, da die binomische Verteilung nicht monoton ist.

2.6.3.4 Die Methode `iAb_Ag()`

In dieser Methode werden die Antikörper-Antigen-Interaktionen betrachtet. Nachdem eine Affinität für einen p-Match zwischen dem Antikörper und dem Antigen festgestellt worden ist, werden die beiden Bindungspartner aus der Simulation entfernt.

2.6.3.5 Die Methode `death()`

Jede Zelle, Antikörper oder Antigen im System hat eine bestimmte Halbwertszeit. Diese bestimmt die Wahrscheinlichkeit, mit der eine Zelle, Antikörper oder Antigen während eines Zeitschritts stirbt, d.h. aus der Simulation entfernt wird. Die spezifische Wahrscheinlichkeit der Halbwertszeit während eines Zeitschritts wird durch $\exp(-\ln(2)/\tau)$ definiert, wobei τ die Halb-

wertszeit der Zelle, des Antikörpers oder Antigens darstellt. Die Methode `death()` umfasst neben der Entfernung naiver B-Zellen, Plasma- und Gedächtniszellen aus der Simulation, auch den durch die vorgegebene Halbwertszeit induzierten Abbau der Antikörper und Antigene. Denn für jeden Bestandteil gibt es einen, vor dem Simulationsbeginn festgelegten Parameter, der die Halbwertszeit des betrachteten Modellbestandteils definiert. So steht `_tauAb` für die Halbwertszeit des Antikörpers, `_tauAg` für die des Antigens, `_tauB` für die Halbwertszeit der naiven B-Zelle, `_tauMB` für die der Gedächtniszelle und `_tauPB` für die der Plasma-B-Zelle.

2.6.3.6 Die Methode `flow()`

Um die Anzahl der B-Zellen ohne Infektionseinfluss konstant zu halten, müssen B-Zellen bei Verlust vom Knochenmark nachproduziert werden. Wenn die Anfangszahl an B-Zellen `_initB` ist und die Halbwertszeit einer naiven, undifferenzierten B-Zelle über `_tauB` vorgegeben ist, werden $\ln(2) \times (_initB / _tauB)$ B-Zellen bei jedem Zeitschritt produziert. Diese Zellen werden dem Gitter an zufällig ausgewählten Positionen durch die Methode `flow()` eingefügt.

2.6.3.7 Die Methode `diffusion()`

```

void diffusion() {
...
/* Swap the lattices */
...
/* Antibody diffusion -- move from temporary to current lattice */
for ...
    switch (newLoc) {
        case 0: addAb(bt,x,y); break;
        case 1: addAb(bt,x-1,i+y); break;
        case 2: addAb(bt,x-1,i+y+1); break;
        case 3: addAb(bt,x,y-1); break;
        case 4: addAb(bt,x,y+1); break;
        case 5: addAb(bt,x+1,i+y); break;
        case 6: addAb(bt,x+1,i+y+1); break;
    }
...
/* Antigen diffusion -- move from temporary to current lattice */
...
/* B cell diffusion -- move from temporary to current lattice */
...
}
        
```

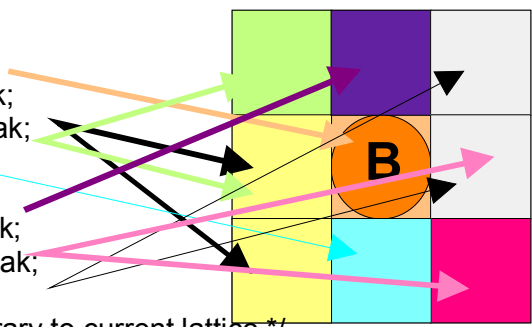


Abb. 20: Programmcode zur Beschreibung der Diffusionsbewegung

Gezeigt ist ein Ausschnitt aus der Methode `diffusion()` für Antikörperdiffusion, in der der angesprochene Antikörper entsprechend der zufälligen Auswahl die ihm zugewiesene nächste Stelle des Nachbarn annimmt. Die Funktion `addAb()` fügt dann den Antikörper in die dann aktuelle Gitterstelle unter Berücksichtigung der toroidalen Geometrie des Gitters ein (siehe Methode `translate()` im Anhang). Für die Antigene und B-Zellen sind in der Methode `diffusion()` entsprechende Algorithmen vorgesehen. Zur Vereinfachung ist die Lage der Nachbarstellen rechteckig angegeben.

In dieser Simulation ist es den Zellen, Antigenen und Antikörpern erlaubt zu ihren nächsten Nachbarstellen zu diffundieren. In einem Dreiecksgitter gibt es 6 nächste Nachbarn, die in Verbindung mit dem Ursprungsstelle stehen (siehe Abb.20). In der Methode `diffusion()` ist es jedem Bestandteil erlaubt, zwischen der Bewegung zu einer der nächsten Nachbarstellen oder dem Verbleib auf seiner aktuellen Position zufällig zu wählen.

Das Problem, das sich durch die Stellen ergibt, die an den Kanten des Gitters gelegen sind, wird durch die Anwendung einer toroidalen Geometrie für das Gitter gelöst. Dies bedeutet, dass die an den Gitterrändern gelegenen Stellen auf der gegenüberliegenden Gitterseite, wie in Abbildung 21 gezeigt, ihren nächsten Nachbarn haben, wenn die Bewegung über die Gittergrenze hinausgeht. Damit ist gewährleistet, dass periodische Grenzbedingungen entstehen und die Systembestandteile nicht aus dem Gitter herausfallen. Dies entspricht auch den biologischen Organen, die auch keine Grenzen besitzen.

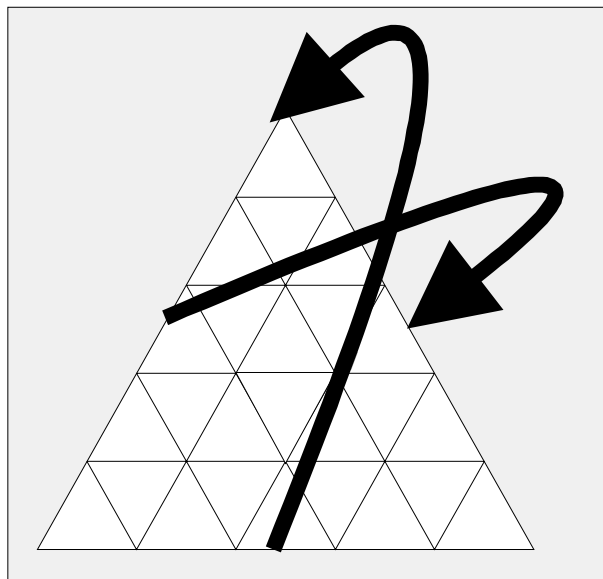


Abb. 21: Darstellung der toroidalen Gittergeometrie

In dieser Zeichnung ist in das toroidale Dreiecksgitter dargestellt. Die Bewegungsrichtung für eine B-Zelle, für einen Antikörper oder ein Antigen sind mit Pfeilen ausgedrückt.

2.7 Simulationsergebnisse

Um zu verifizieren, dass dieses System, sich wie ein reales Immunsystem verhält, wurde ein Simulationsversuch durchgeführt. Dazu wurde zu den Zeitschritten $t=0$ und $t=100$ je eine Antigeninjektion definiert. Die Ergebnisse dieser Simulation sind Abbildung 22 dargestellt.

Bei der ersten Injektion ist die Immunantwort gering. Denn es wird Zeit dafür benötigt, damit sich die Klone der antwortenden Zellen ausgehend von ihrer geringen undifferenzierten Konzentration entwickeln können. Es erfolgt nur die Produktion einer geringen Antikörperkonzentration, so dass einige Zeit zur Antigenentfernung aus dem System notwendig ist. Jedoch ist die Konzentration an T-Zellen bei diesem Injektionsschritt höher als die der B-Zellen. Dies ist vermutlich darauf zurückzuführen, dass in dieser Simulation auch Antigenpräsentierende Zellen (APC) vorhanden sind, die in Konkurrenz zu den B-Zellen auftreten. Deren Populationsverhalten ist in der Ergebnispräsentation in Abbildung 22 nicht aufgeführt.

Dagegen werden bei der zweiten Antigeninjektion in das System sehr schnell geeignete B- und T-Zellen zur Verfügung gestellt, so dass die Antwort sehr schnell erfolgt. Es werden sehr viel mehr Antikörper produziert, wodurch die Antigene ziemlich schnell eliminiert werden können. Hier ist auch festzustellen, dass die T-Zell-Konzentration doppelt so groß ist, wie die der B-Zellen, was für die Existenz von APCs in diesem Modellversuch spricht. Leider sind auch in diesem Fall die Konzentrationen der APCs nicht angegeben.

Damit ist trotzdem gezeigt, dass das simulierte Verhalten dem einer realen adaptiven, aber fragmentalen Immunantwort sehr ähnlich bzw. identisch ist.

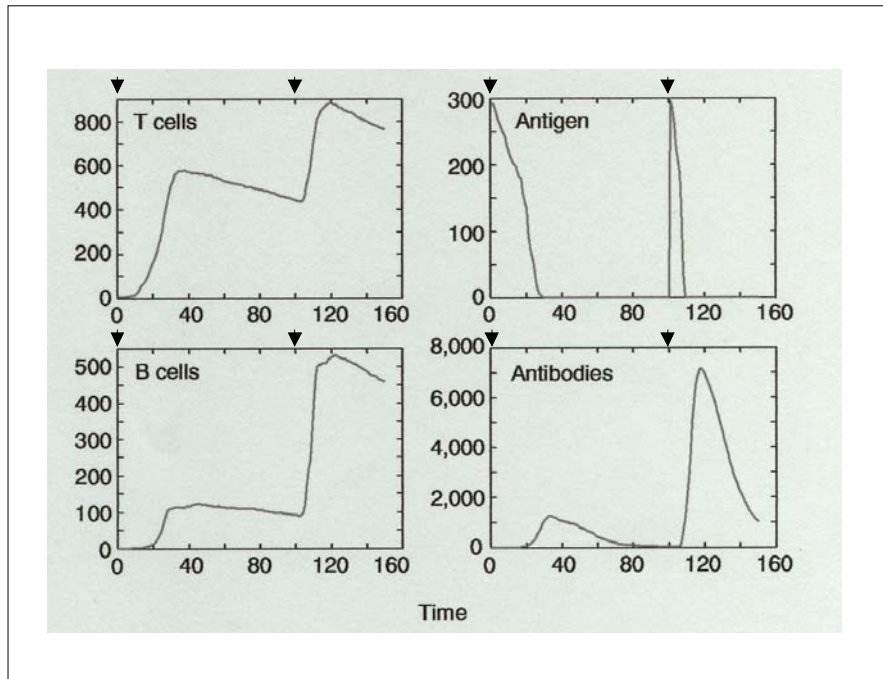


Abb. 22: Ergebnisse des IMMSIM-Simulationsversuchs

Gezeigt ist die Entwicklung der B- Zelle, T-Zelle, Antigen- und Antikörperpopulation eines Systems, das mit Antigen zu den Zeitschritten $t=0$ und $t=100$ injiziert wurde (siehe Pfeile).

3. Diskussion und Zusammenfassung

Mit IMMSIM konnte das adaptive Immunsystem funktionell modelliert und simuliert werden (siehe Abbildung 23).

Allerdings war dies nur dadurch möglich, dass die funktionellen Einheiten der Zelltypen wie MHC-Komplexe und Rezeptoren sowie die Antikörper und Antigene, Peptide und Epitope eingeschlossen, über Bitstrings definiert wurden. Diese Strings interagieren komplementär und binden aneinander, wenn eine ausreichende Anzahl an Matches erreicht oder überschritten wird. Da diese Interaktionen auf bestimmte Zielzellen (B-Zelle) oder Zielmoleküle (z.B. Antikörper) in der Simulation beschränkt sind, sind damit die biologischen Interaktionen modellmässig nachgezeichnet. Selbst die biologischen Eigenschaften der Zelle, wie Lebensdauer, Einstellung in den entsprechenden Organen sind im Modell berücksichtigt. Auch die Induktion der Antikörperproduktion nach vorheriger Differenzierung der undifferenzierten B-Zelle zu einer Plasmazelle und einer Gedächtniszelle ist in der Simulation enthalten. Selbst der Ort der Immunabwehrreaktionen ist mit einem modifizierten zellulären Automaten in der Simulation berücksichtigt. Die Bewegung der Zellen (wie B-Zellen) und Moleküle (z.B. Antikörper) des Immunsystems sind im Modell vorgesehen.

Der Vorteil dieser Simulation gegenüber dem biologischen System besteht damit in dem modularen Aufbau des Simulationsmodells IMMSIM. Damit lassen sich verschiedene immunologische Prozesse, wie die Gewinnung und Prozessierung von Antigenen und Beeinflussung verschiedener Zellpopulationen beobachten. Damit können Aspekte der klonalen Selektion (Mutation, Affinitätsreifung), des Thymus und der Toleranz, aber auch der Vakzinierung betrachtet werden.

Zu berücksichtigen ist allerdings, dass eine völlige 1 zu 1 Übersetzung des biologischen Immunsystems in ein künstliches Immunsystem wegen der biologischen 3D-Erkennungsmechanismen nicht möglich ist. Vielmehr ist es eine „Transformation“ aus dem biologischen „3D-Raum“ in den komplementären Bitstringraum unter Beibehaltung der intermolekularen und interzellulären Konzepte durchführbar.

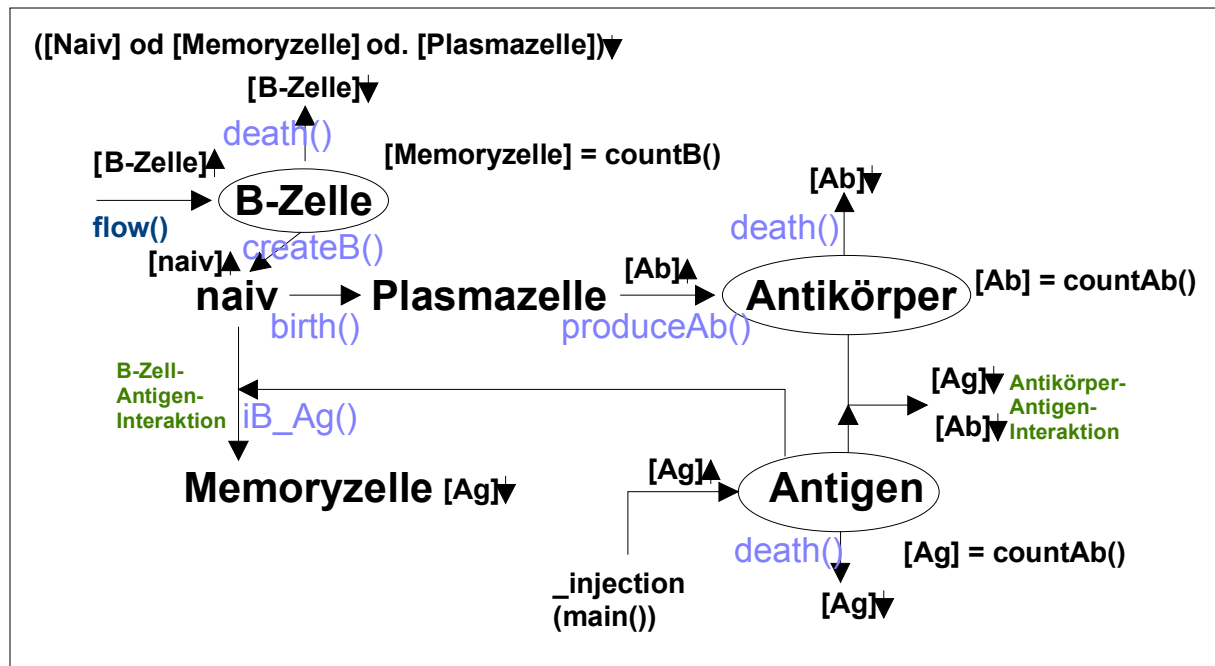


Abb. 23: Darstellung der tutorial Version von IMMSIM

Die zusammenfassende Darstellung zeigt die Simulation IMMSIM mit den Bestandteilen B-Zelle (undifferenzierte (naive) B-Zelle, Plasmazelle und Memoryzelle), Antikörper und Antigen. Die verändernden Methoden, wie `produceAb()` sind blau hervorgehoben. Feststellende Funktionen, wie `countAb()` sind schwarz gekennzeichnet. Die Konzentrationen sind in eckigen Klammern angegeben. Die Veränderungen der Konzentrationen sind mit senkrechten Pfeilen angegeben. Dabei steht \uparrow für steigende Konzentration, \downarrow für fallende Konzentrationen. Die Interaktionen zwischen den Antikörpern und den Antigenen bzw. zwischen Antikörper und B-Zelle sind grün hervorgehoben. Die Pfeile zwischen den B-Zelltypen, Antikörpern und Antigen symbolisieren die entsprechenden Beziehungen, mit der die Struktur in die andere überführt wird. Die Kennzeichnungen an den Pfeilen beschreiben die Methoden durch welche der Übergang vollzogen wird bzw. die Auswirkung auf die Konzentration.

Alternativ hätte die Simulation mittels üblicher Differentialgleichungen (ODE) kombiniert mit numerischer Integration zur Modellierung eingesetzt werden können. Aber ODE-Modelle benötigen genügend grosse Populationen, um die Wahrscheinlichkeit bestimmen zu können. Jedoch hat jede Zelle ihre einzigartige Lebensgeschichte. Ferner geben ODEs nur das Durchschnittsverhalten eines Systems wieder. Auch sind Modelle eines Immunsystems oft nicht linear, was die numerische Integration oder analytische Lösung eines ODE-Modells besonders schwierig macht. Damit scheidet diese Art von Modellen für eine Simulation eines spezifischen komplex strukturierten adaptiven Immunsystems aus.

Stattdessen ist es mit einer modular aufgebauten, nach dem Komplementaritätsprinzip agierenden Simulation, aus einem komplexen System wie dem biologischen Immunsystem, möglich, deren grundlegenden Gesetzmässigkeiten zu extrahieren. Dieser Vorgang ist in einem finiten System wie dem Computersystem mit realistischer Grösse ohne Artefakte als Ergebnisse durchführbar. Dies bedeutet, dass die Simulation des adaptiven Immunsystems nur einen Anfang darstellt und sich auf das gesamte Immunsystem ausdehnen lässt.

Andererseits ist mit der Transformation und der Simulation im Komplementärraum basierend auf Bitstrings ein weites informatisches Forschungsgebiet eröffnet worden, in dem neue informatische Konzepte und Implementierungen basierend auf biologischen Schemas entwickelt werden können, die in der Lage sind, selbstständig eine „virozide“ Zukunft für Computersysteme zu bewirken.

4. Literaturverzeichnis

Celada F, Seiden PE (1992)

A computer model of cellular interactions in the immune system.

Immunol Today 13:56-62. Review.

- Celada F, Seiden PE (1996)
Affinity maturation and hypermutation in a simulation of the humoral immune response.
Eur J Immunol. 26:1350-8.
- Janeway CA, Travers P (2002)
Immunologie
Spektrum, Heidelberg
- Janeway CA, Travers P (1996)
Immunology. The Immune System in Health and Disease
Garland, New York
- Klein J (1999)
Immunologie
VCH, Weinheim
- Kleinstein SH, Seiden PE (2000)
Simulating the Immune System
Computing in Science & Engineering 2(4): 69-77.
- Kohler B, Puzone R, Seiden PE, Celada F (2000)
A systematic approach to vaccine complexity using an automaton model of the cellular and humoral immune system. I. Viral characteristics and polarized responses.
Vaccine 19:862-76.
- Morpurgo D, Serentha R, Seiden PE, Celada F (1995)
Modelling thymic functions in a cellular automaton.
Int Immunol. 7(4):505-16.
- Perelson AS, Weisbuch AS (1997)
Immunology for Physicists
Review of Modern Physics 69: 1219-1268
- Prikrylova D, Jilek M, Waniewski J (1992)
Mathematical Modelling of the Immune Response
CRC Press, Boca Raton
- Roitt IM, Brostoff J, Male DK (1995)
Kurzes Lehrbuch der Immunologie
Georg Thieme Verlag, Stuttgart
- Seiden PE, Celada F (1992)
A model for simulating cognate recognition and response in the immune system.
J Theor Biol. 158:329-57.
- Stewart JJ, Agosto H, Litwin S, Welsh JD, Shlomchik M, Weigert M, Seiden PE (1997)
A solution to the rheumatoid factor paradox: pathologic rheumatoid factors can be tolerated by competition with natural rheumatoid factors.
J Immunol. 159:1728-38
- Wolfram S (1984)
Preface to Cellular Automata
Proc. Interdisciplinary Workshop, North-Holland Amsterdam, vii-xii
www.stephenwolfram.com/publications/articles/ca/84-preface/index.html

5. Anhang

In Abbildung 24 a-c ist der gesamte Programmcode der tutorial Version on IMMSIM aufgeführt. Die vollständige Version ist unter www.cs.princeton.edu/immsim/software.html zu beziehen.

<pre> IMMSIM – Tutorial Version * written by Steven Kleinste (stevenk@cs.princeton.edu) Copyright (c) 2000 by Steven Kleinste ("LICENSOR"). All Rights Reserved. This software is provided as is without warranty of any kind. The entire risk as to the results and performance of this software is assumed by the user. This program is provided for the user's personal, non-commercial, experimental use. All title, ownership and rights to this program and any copies remain with LICENSOR. The user is permitted to create derivative works to this program. However, all copies of the program and its derivative works must contain this copyright notice. Full versions of IMMSIM can be downloaded from www.cs.princeton.edu/immsim */ #include <stdlib.h> #include <stdio.h> #include <time.h> #include <math.h> #ifdef WIN32 #include <debug.h> #include <crtdbg.h> #define new DEBUG_NEW #endif static char THIS_FILE[] = __FILE__; #endif /* Constants */ #define LN2 0.6931471806 /* Some useful probability functions WARNING: This program uses the built-in rand() function to generate random numbers. For any real research it is best to use a higher quality generator. */ #define rand01() ((rand()/(RAND_MAX+1.0))) /* Produces a random number in the interval [0,1) */ #define randint(L,H) ((H<L) ? ((int)(rand01()*(H-L)+1)) + (L)) : (L)) /* Produces a random integer in the interval [L,H] */ #define COIN(p) (rand01() < p) /* Returns true with probability p */ #define PDIE(halfLife) (1-exp(-LN2/(halfLife))) /* Returns the probability of death at each time-step, given halfLife */ #define __max(a,b) (((a) > (b)) ? (a) : (b)) #define __min(a,b) (((a) < (b)) ? (a) : (b)) unsigned long combinatorial(unsigned n, unsigned r) { /* Returns (n C r) */ unsigned long i, numerator = 1, denominator = 1; </pre>	<pre> for (i = n; i >= __max(r, n - r) + 1; i--) numerator = numerator * i; for (i = __min(r, n - r); i >= 1; i--) denominator = denominator * i; return (numerator / denominator); } /* B cell structure */ enum bType {NAIVE,PLASMA,MEMORY,ANY}; /* Types of B cells */ struct BCELL { /* NOTE: The number of matching bits with antigen (match) is pre-calculated when the B cell is created and must be changed whenever slg is altered */ struct BCELL * next; /* Linked list pointers */ struct BCELL * prev; enum bType type; /* Naive, Memory, Plasma */ short divide; /* Number of divisions left to undergo */ unsigned long slg; /* Surface immunoglobulin bitstring receptor */ unsigned match; /* Number of matching bits with antigen, based on slg (pre-calculated) */ }; /* Model Parameters (variable names are preceeded with an underscore) */ #define _xDim 20 /* X dimension of lattices (must be even) */ #define _yDim 20 /* Y dimension of lattices (must be even) */ #define _ag 0x0 /* The antigen bitstring */ #define _divideB 2 /* Number of B cell divisions on activation (3) */ #define _initB 200 /* Initial number of B cells (1000) */ #define _pmRatio 0.5 /* Probability that B cell daughter is plasma cell */ #define _tauAb 3 /* Antibody half-life */ #define _tauAg 0 /* Antigen half-life */ #define _tauB 10 /* Naive B cell half-life */ #define _tauMB 40 /* Memory B cell half-life (50) */ #define _tauPB 10 /* Plasma B cell half-life */ #define _numAb 5 /* Number of Ab secreted by plasma B cell per time-step */ #define _injection 3000 /* How many Ag to inject */ #define _injection1 0 /* Time of first injection */ #define _injection2 150 /* Time of second injection */ #define _bits 8 /* Length of slg bitstring */ #define _minMatch 7 /* Minimum number of matching bits required for binding */ #define _affLevel 0.01 /* Minimum affinity */ #define _affEnhance 12.5 /* Factor affinity increase for each additional matching bit */ float IS[_bits+1]; /* Affinity for a p-bit match */ /* Data structures for storing simulation entities */ short c; /* Current lattice (ie, first index in all lattice arrays) -- for diffusion */ unsigned abLattice[2][_bits + 1][_xDim][_yDim]; /* Antibody -- second index gives number of matching bits with antigen */ unsigned agLattice[2][_xDim][_yDim]; /* Antigen */ struct BCELL * bLattice[2][_xDim][_yDim]; /* B cell */ </pre>
<pre> /* Counting functions */ unsigned countB (enum bType t, int match) { unsigned count = 0; int x,y; struct BCELL *temp; for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { temp = bLattice[c][x][y]; while (temp != NULL) { if ((t == ANY temp->type == t) && (match < 0 match == temp->match)) count++; temp = temp->next; } } } return count; } unsigned countAb () { unsigned bt, count = 0; int x,y; for (bt=0; bt<=_bits; bt++) { for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { count += abLattice[c][bt][x][y]; } } } return count; } unsigned countAg () { unsigned count = 0; int x,y; for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { count += agLattice[c][x][y]; } } return count; } /* Helper functions */ void translate (int *x, int *y) { /* Periodic boundary conditions */ if (*x < 0) *x = _xDim + *x; </pre>	<pre> } unsigned match (unsigned long A, unsigned long B) { unsigned long mVector; unsigned long pos = 0x1; unsigned match = 0; int bit; mVector = A^B; for (bit=0; bit<_bits; bit++) { if (mVector & pos) match++; pos <<= 1; } return match; } struct BCELL *createB(struct BCELL *b) { /* Creates a single B cell */ struct BCELL *temp = (struct BCELL *) malloc(sizeof(struct BCELL)); temp->next = NULL; temp->prev = NULL; if (b == NULL) { /* Creating new B cell from bone marrow */ temp->type = NAIVE; temp->divide = 0; temp->slg = randint(0,0x1<<_bits); } else { /* Dividing B cell */ temp->type = b->type; temp->divide = b->divide; temp->slg = b->slg; } /* Pre-calculate the number of matching bits with the antigen */ temp->match = match(temp->slg,_ag); return temp; } void randomizeB() { unsigned i, count, pos = 0; int x,y; struct BCELL *b; for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { //Count number of B cells b = bLattice[c][x][y]; count = 0; while (b != NULL) {count++; b = b->next;} </pre>

Abb. 24a: Programmcode von IMMSIM

Dargestellt ist der erste Teil des Programmcode von IMMSIM beginnend bei I.

Überblick und Simulation des Immunsystems

```

if (count > 1) {
    //Set aside temporary storage
    struct BCELL **temp = (struct BCELL **) malloc(count * sizeof(struct BCELL));
    //Move B cells to array and randomize
    b = bLattice[c][x][y]; count = 0;
    while (b != NULL) {temp[count++] = b; b = b->next;}

    for (i=0; i<count; i++) {
        int md = randInt(i, count-1);
        struct BCELL *t = temp[i];
        temp[i] = temp[md];
        temp[md] = t;
    }

    bLattice[c][x][y] = temp[0];

    for (i=1; i<count-1; i++) {temp[i]->next = temp[i+1]; temp[i]->prev = temp[i-1].}

    temp[0]->next = temp[1]; temp[0]->prev = NULL;
    temp[count-1]->next = NULL; temp[count-1]->prev = temp[count-2];

    free(temp);
}
}
}

/* Functions to add entities to the current lattice incorporating periodic boundary conditions */
void addAb(unsigned bt, int x, int y) {translate(&x,&y); abLattice[c][bt][x][y]++;}
void addAg(int x, int y) {translate(&x,&y); agLattice[c][x][y]++;}
void addB(struct BCELL *b, int x, int y) {
    if (b != NULL) {
        translate(&x,&y);
        b->prev = NULL;

        if (bLattice[c][x][y] == NULL) b->next = NULL;
        else {
            b->next = bLattice[c][x][y];
            bLattice[c][x][y]->prev = b;
        }

        bLattice[c][x][y] = b;
    }
}

/* Functions implementing the various simulation processes */
void death () {
    short die;
    int x,y;

```

V

```

    unsigned bt,ab,ag;
    struct BCELL *b, *temp;

    /* Antibody death */
    for (bt=0; bt<= _bits; bt++)
        for (x=0; x< _xDim; x++)
            for (y=0; y< _yDim; y++)
                for (ab=abLattice[c][bt][x][y]; ab>0; ab--)
                    if (_tauAb && COIN(PDIE(_tauAb))) abLattice[c][bt][x][y]--;

    /* Antigen death */
    for (x=0; x< _xDim; x++)
        for (y=0; y< _yDim; y++)
            for (ag=agLattice[c][x][y]; ag>0; ag--)
                if (_tauAg && COIN(PDIE(_tauAg))) agLattice[c][x][y]--;

    /* B cell death */
    for (x=0; x< _xDim; x++) {
        for (y=0; y< _yDim; y++) {
            b = bLattice[c][x][y];
            while (b != NULL) {
                temp = b->next;
                die = 0;

                switch (b->type) {
                    case NAIVE: if (_tauB && COIN(PDIE(_tauB))) die = 1; break;
                    case PLASMA: if (_tauPB && COIN(PDIE(_tauPB))) die = 1; break;
                    case MEMORY: if (_tauMB && COIN(PDIE(_tauMB))) die = 1; break;
                }

                if (die) { /* Remove the cell from the list */
                    if (b->prev) b->prev->next = b->next;
                    else bLattice[c][x][y] = b->next;

                    if (b->next) b->next->prev = b->prev;

                    free(b);
                }
                b = temp;
            }
        }
    }

    void diffusion() {
        int x,y;
        unsigned bt,ab,ag;
        struct BCELL *b, *temp;

        /* Swap the lattices */
        short newLoc, l, old = c;

```

VI

```

c = (c + 1) % 2;

/* Antibody diffusion -- move from temporary to current lattice */
for (bt=0; bt<= _bits; bt++)
    for (x=0; x< _xDim; x++) {
        i = (x%2 ? 0 : -1);
        for (y=0; y< _yDim; y++) {
            for (ab=abLattice[old][bt][x][y]; ab>0; ab--) {
                short newLoc = randInt(0,6);
                switch (newLoc) {
                    case 0: addAb(bt,x,y); break;
                    case 1: addAb(bt,x-1,y); break;
                    case 2: addAb(bt,x-1,y+1); break;
                    case 3: addAb(bt,x,y-1); break;
                    case 4: addAb(bt,x,y+1); break;
                    case 5: addAb(bt,x+1,y); break;
                    case 6: addAb(bt,x+1,y+1); break;
                }
            }
            abLattice[old][bt][x][y] = 0;
        }
    }

/* Antigen diffusion -- move from temporary to current lattice */
for (x=0; x< _xDim; x++) {
    short i = (x%2 ? 0 : -1);
    for (y=0; y< _yDim; y++) {
        for (ag=agLattice[old][x][y]; ag>0; ag--) {
            newLoc = randInt(0,6);
            switch (newLoc) {
                case 0: addAg(x,y); break;
                case 1: addAg(x-1,y); break;
                case 2: addAg(x-1,y+1); break;
                case 3: addAg(x,y-1); break;
                case 4: addAg(x,y+1); break;
                case 5: addAg(x+1,y); break;
                case 6: addAg(x+1,y+1); break;
            }
        }
        agLattice[old][x][y] = 0;
    }
}

/* B cell diffusion -- move from temporary to current lattice */
for (x=0; x< _xDim; x++) {
    short i = (x%2 ? 0 : -1);
    for (y=0; y< _yDim; y++) {
        b = bLattice[old][x][y];
        while (b != NULL) {
            temp = b->next;
            newLoc = randInt(0,6);
            switch (newLoc) {
                case 0: addB(b,x,y); break;
                case 1: addB(b,x-1,y); break;
                case 2: addB(b,x-1,y+1); break;
                case 3: addB(b,x,y-1); break;
                case 4: addB(b,x,y+1); break;
                case 5: addB(b,x+1,y); break;
                case 6: addB(b,x+1,y+1); break;
            }
            b = temp;
            bLattice[old][x][y] = NULL;
        }
    }

    void flow() {
        /* Add new B cells from bone marrow to balance expected death */
        unsigned bt, num = (unsigned) (0.5 + LN2 * _initB / _tauB);

        for (bt=0; bt<num; bt++) addB(createB(NULL),randInt(0,_xDim-1),randInt(0,_yDim-1));
    }

    void birth () {
        int x,y;
        struct BCELL *b, *temp;

        /* B cell birth */
        for (x=0; x< _xDim; x++) {
            for (y=0; y< _yDim; y++) {
                b = bLattice[c][x][y];
                while (b != NULL) {
                    if (b->divide > 0) {
                        b->divide--;

                        temp = createB(b);

                        if (b->divide == 0) {
                            if (COIN(_pmRatio)) (b->type = PLASMA; b->divide = 0;
                            if (COIN(_pmRatio)) (temp->type = PLASMA; temp->divide = 0;
                        }

                        addB(temp,x,y);
                    }
                    b = b->next;
                }
            }
        }

        void produceAb () {

```

VII

```

c = (c + 1) % 2;

/* Antibody diffusion -- move from temporary to current lattice */
for (bt=0; bt<= _bits; bt++)
    for (x=0; x< _xDim; x++) {
        i = (x%2 ? 0 : -1);
        for (y=0; y< _yDim; y++) {
            for (ab=abLattice[old][bt][x][y]; ab>0; ab--) {
                short newLoc = randInt(0,6);
                switch (newLoc) {
                    case 0: addAb(bt,x,y); break;
                    case 1: addAb(bt,x-1,y); break;
                    case 2: addAb(bt,x-1,y+1); break;
                    case 3: addAb(bt,x,y-1); break;
                    case 4: addAb(bt,x,y+1); break;
                    case 5: addAb(bt,x+1,y); break;
                    case 6: addAb(bt,x+1,y+1); break;
                }
            }
            abLattice[old][bt][x][y] = 0;
        }
    }

/* Antigen diffusion -- move from temporary to current lattice */
for (x=0; x< _xDim; x++) {
    short i = (x%2 ? 0 : -1);
    for (y=0; y< _yDim; y++) {
        for (ag=agLattice[old][x][y]; ag>0; ag--) {
            newLoc = randInt(0,6);
            switch (newLoc) {
                case 0: addAg(x,y); break;
                case 1: addAg(x-1,y); break;
                case 2: addAg(x-1,y+1); break;
                case 3: addAg(x,y-1); break;
                case 4: addAg(x,y+1); break;
                case 5: addAg(x+1,y); break;
                case 6: addAg(x+1,y+1); break;
            }
        }
        agLattice[old][x][y] = 0;
    }
}

/* B cell diffusion -- move from temporary to current lattice */
for (x=0; x< _xDim; x++) {
    short i = (x%2 ? 0 : -1);
    for (y=0; y< _yDim; y++) {
        b = bLattice[old][x][y];
        while (b != NULL) {
            temp = b->next;
            newLoc = randInt(0,6);
            switch (newLoc) {
                case 0: addB(b,x,y); break;
                case 1: addB(b,x-1,y); break;
                case 2: addB(b,x-1,y+1); break;
                case 3: addB(b,x,y-1); break;
                case 4: addB(b,x,y+1); break;
                case 5: addB(b,x+1,y); break;
                case 6: addB(b,x+1,y+1); break;
            }
            b = temp;
            bLattice[old][x][y] = NULL;
        }
    }

    void flow() {
        /* Add new B cells from bone marrow to balance expected death */
        unsigned bt, num = (unsigned) (0.5 + LN2 * _initB / _tauB);

        for (bt=0; bt<num; bt++) addB(createB(NULL),randInt(0,_xDim-1),randInt(0,_yDim-1));
    }

    void birth () {
        int x,y;
        struct BCELL *b, *temp;

        /* B cell birth */
        for (x=0; x< _xDim; x++) {
            for (y=0; y< _yDim; y++) {
                b = bLattice[c][x][y];
                while (b != NULL) {
                    if (b->divide > 0) {
                        b->divide--;

                        temp = createB(b);

                        if (b->divide == 0) {
                            if (COIN(_pmRatio)) (b->type = PLASMA; b->divide = 0;
                            if (COIN(_pmRatio)) (temp->type = PLASMA; temp->divide = 0;
                        }

                        addB(temp,x,y);
                    }
                    b = b->next;
                }
            }
        }

    void produceAb () {

```

VIII

Abb. 24b: Programmcode von IMMSIM

Dargestellt ist der zweite Teil des Programmcode von IMMSIM beginnend bei V.

<pre> int x,y; struct BCELL 'b; /* Plasma B cell production of antibodies */ for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { b = bLattice[c][x][y]; while (b != NULL) { if (b->type == PLASMA) abLattice[c][b->match][x][y] += _numAb; b = b->next; } } } void iB_Ag () { /* B cell + Antigen interaction */ int x,y; unsigned ag; struct BCELL 'b; /* First, randomize the B cell lists. Why is this necessary? */ randomizeB(); /* Next, do the interaction */ for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { b = bLattice[c][x][y]; while (b != NULL) { for (ag=agLattice[c][x][y]; ag>0; ag--) { if (b->type != PLASMA && b->divide == 0 && COIN(!IS[b->match])) { /* Bound Ag -- stimulate the B cell, remove the antigen */ b->type = MEMORY; b->divide = _divideB; agLattice[c][x][y]--; break; /* Move onto the next B cell */ } } b = b->next; } } } </pre>	<pre> void iAb_Ag () { /* Antibody + Antigen interaction */ int x,y; unsigned bt, ab, ag; for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { for (bt=0; bt<=_bits; bt++) { float pBind = IS[bt]; if (pBind > 0.0) { for (ab=abLattice[c][bt][x][y]; ab>0; ab--) { for (ag=agLattice[c][x][y]; ag>0; ag--) { if (COIN(pBind)) { abLattice[c][bt][x][y]--; agLattice[c][x][y]--; break; /* Move onto the next antibody */ } } } } } } } /* MAIN SIMULATION FUNCTION */ int main () { int x,y; unsigned m, bt, simTime, ag; struct BCELL 'b, 'temp; char buffer[500]; FILE *outFile; /* Initialize the (naive) random number generator */ srand((unsigned) time(NULL)); /* Initialize the affinity vector */ for(m=0; m<=_bits; m++) IS[m] = 0.0; IS[_minMatch]=_affLevel; </pre>
<pre> for(m=_minMatch+1; m<=_bits; m++) { IS[m] = _affEnhance * combinatorial(_bits,m-1) / combinatorial(_bits,m) * IS[m-1]; if (!IS[m] > 1.0) IS[m] = 1.0; } /* Open file for data output */ if((outFile = fopen("out.txt","wt")) == NULL) { printf("ERROR: Unable to open/create output file"); exit(-1); } /* Set the current lattice */ c = 0; /* Add the initial B cells to the lattice */ for (bt=0; bt<_initB; bt++) addB(createB(NULL),randInt(0,_xDim-1),randInt(0,_yDim-1)); /* Main simulation loop */ /* Cleanup */ fclose(outFile); for (x=0; x<_xDim; x++) { for (y=0; y<_yDim; y++) { b = bLattice[c][x][y]; while (b != NULL) { temp = b->next; free(b); b = temp; } } } #ifdef WIN32 #ifdef _DEBUG _CrtDumpMemoryLeaks(); /* Check for memory leaks */ #endif #endif return 1; } </pre>	

Abb. 24c: Programmcode von IMMSIM

Dargestellt ist der letzte Teil des Programmcodes von IMMSIM beginnend bei IX.