Computación y Sistemas Vol. 7 Núm. 4 pp. 249-259
© 2004, CIC-IPN, ISSN 1405-5546, Impreso en México

On Modelling an Immune System

Sobre un Modelo Computacional del Sistema Inmune

Raúl Monroy[1], Rosa Saab[2] and Fernando Godínez[3]
[1]Departamento de Ciencias Computacionales, Campus Estado de México del Tecnológico de Monterrey
Carretera al lago de Guadalupe Km 3.5, Col. Margarita Maza de Juárez, Atizapán de Zaragoza, 52926, México, Mexico
raulm@itesm.mx
[2]Instituto Tecnológico y de Estudios Superiores de Coacalco, 16 de Septiembre 54, Col. Cabecera Municipal, Coacalco de Berriozábal, Estado de México, 55717, Mexico
rosa.saab@itesm.mx
[3]Centro de Sistemas Inteligentes, Campus Monterrey del Tecnológico de Monterrey, CETEC Torre Sur 5o Piso, Eugenio Garza Sada 2501, Monterrey, Nuevo León, 64849, Mexico
fgodinez@itesm.mx

Article received on March 27, 2003, accepted on June 08, 2004

Abstract

Immune systems of live forms have been an abundant source of inspiration to contemporary computer scientists. Problem solving strategies, stemming from known immune system phenomena, have been successfully applied to challenging problems of modern computing. However, research in artificial immune systems has overlooked establishing a coherent model of known immune system behaviour. This paper aims reports on an preliminary computer model of an immune system, where each immune system component is specified in terms of its observable behaviour using a suitable process algebra. Our model is not only suitable to simulation but also and more importantly to formal analyzes of immune system behaviour.
Keywords: Multiagent systems and Distributed AI, Immunology, Immune Based Computer Systems.

Resumen

El sistema inmune de los seres vivientes ha sido una fuente rica de inspiración para científicos en computación contemporáneos. Estrategias de solución de problemas, cuyos orígenes se encuentran en fenómenos inmunológicos, han sido exitosamente aplicadas en problemas desafiantes de la computación moderna. Sin embargo, el trabajo de investigación en sistemas inmunológicos artificiales ha ignorado el establecer un modelo coherente que incorpore comportamientos conocidos y aceptados del sistema inmune. Este artículo reporta un modelo computacional preliminar del sistema inmune, en el cual cada componente del mismo es especificado en términos de su comportamiento observable, usando una álgebra de procesos adecuada. Nuestro modelo no sólo es adecuado para la simulación sino también y aún más importante para el análisis formal del comportamiento de un sistema inmune.
Palabras Clave: Sistemas Multiagentes y AI Distribuida, Inmunología, Sistemas de Cómputo basado en Inmunidad.

1. Introduction

Underlying natural evolution, the immune system of live forms is a robust, effective and highly reliable defense mechanism. An immune system is made out of a number of simple components, each of which carries out a small, specific task. Yet the immune system is able to achieve complex goals. Its overall behaviour is the product of the number of interactions that happen amongst its individuals. Thus, the immune system is a large, complex, highly coordinated *communicating system*.

The immune system has recently captured increasing interest in computing. Biology and Computing have merged together, yielding two central disciplines. First, *computational Biology*, which aims to duplicate within a machine outstanding biological phenomena. Second, *artificial immune systems,* which aims to discover problem solving strategies using the immune system as source of inspiration. Yet, research on artificial immune systems seems to proceed without a formal model of the natural system from which it has developed.

This paper reports on an experiment towards the computer modelling of the immune system. The model provided is intended to be used both by biologists, as a controlled laboratory, and by computer scientists, as a reference for developing and hopefully discovering new problem solving strategies.

The model comprehends only aspects of behaviour. Hence, it is written in a process algebra. Process algebra is a suitable means for modelling, analyzing and building large communicating systems from simpler ones. These communicating systems have their own identity circumscribed by their entire capabilities of interaction. Instead of using a known process algebra, such as PEPA [6] or WSCCS [27], we use a combination thereof. This is because no existing process algebra can singly model an system as complex as the immune one. We shall argue that a model of immune system behaviour should encompass issues about activity duration, probabilistic quantification for non-determinism a multiple communication.

This paper also hints how to formally specify some properties associated with our model of the immune system. Example properties include effectiveness, self-stabilization and robustness. The model, unfortunately, is not suitable for formal analysis yet. We outline directions to indicate what sort of toolbox is required to formally analyze it.

1.1. Paper Overview

The rest of the paper is organized as follows: We first overview outstanding behavioral aspects of an immune system, motivating the sort of formalism required to model them (§2). Then we argue that these informal requirements can be captured using a process algebra (§3). Then, using this algebra, we specify the behaviour of some immune system components (§4). We next outline how to conduct formal model analysis, as well as providing guidelines towards mechanical analysis (§5). Finally, related work is contrasted (§6) and conclusions drawn from our experiments presented (§7).

2. Modelling an Immune System

This section briefly accounts for current knowledge of immune system phenomena. It aims to motivate what takes to formally modelling the behaviour of the immune system.

2.1. Immune Systems: General Features

An immune system consists of a number of components, for example cells, antigens, etc. Each element performs a small, specific task. Components are independent but they coordinate one another. Overall immune system behaviour is a product of the number and kinds of interactions played by its individual components. The immune system is thus a highly *distributed communicating system*.

An immune system is *diverse*: cells of the same kind vary from one another. This involves a cell's ability to non-self detection. So what is distinguishable to one cell may go unnoticed to others [26]. Due to this cell diversity, an immune system is highly robust.

To increase their possibilities of successful antigen detection, lymphocytes, as well as other components, complement their recognition abilities by traveling the body through the blood stream. Mobility is hence crucial to immune system performance.

Also an immune system is highly dynamic: the number of components of the same kind, as well as the structure of the entire system, changes with time and according to circumstances. Non-cell components, such as antigens, will be part of the inhabited body as long as it last. By contrast, some living cells may last a few days; other will last much longer. Living cells therefore obey to a universal clock; they are programmed to die—apoptosis [15].

To compensate for short-lived cells, the body is continuously producing cells and so are some cells of the immune system itself. Yet the population does not grow boundlessly, it self-stabilizes. A cell population increase amounts to the detection of an invading antigen. It is followed by the corresponding decrease when the disease is eliminated [22].

Learning and memory are key to body protection. When facing an new, unknown antigen, not only will the immune system battle the invader, but it will also learn the invader structure, called unfolding [22]. As a result of antigen unfolding, the immune system will evolve a collection of lymphocytes, specially designed and designated to detect and protect the body against the invader.

While traveling through the blood stream, a group of immune system's components may coincide and form a chain, clustering. The cluster is able to provide a more sophisticated defense, stronger than that given by any cell in isolation—emergent behaviour.

Thus, immune systems of life forms are mobile, highly distributed and coordinated communicating systems. However, the behaviour of each element is simple and so in principle suitable to modeling and verification. This paper argues that a process algebra provides a suitable means to express each individual cell and then use a collection of them to build the entire system and analyze its behaviour.

2.2 Process Calculi: General Features

Basic process calculi, such as CCS [16], CSP [12] or ACP [2], provide a means suitable for modeling and analyzing communicating systems. They have been largely and successfully applied to formal modeling and analysis, evolving into an international standard, LOTOS [14].

In a process algebra, communicating systems are modelled as a collection of autonomous agents. An agent is defined in terms of the discrete actions it may perform. In a basic process algebra, an agent's capabilities of interaction do not change with time, actions are atomic, they have no structure and are assumed to be executed instantaneously.

Extended process algebras, such as Message-Passing Process Algebra [10], PEPA [11,6], WSCCS [27], the π -calculus [17,18] and fusion [21], provide a more expressible language, together with a more powerful verification method, that relaxes one or more of the aforementioned limitations. Extended process algebras are capable of expressing and reasoning about, for example, the communication of messages of any kind, value-passing process algebra, action duration and probabilistic choices, PEPA and WSCCS, and mobility, π –calculus or fusion. These aspects, as discussed in the previous section, are required to modelling immune system behaviour.

Process algebras are either synchronous or asynchronous, but not both. Asynchrony is the assumption that concurrent agents run at different speeds, while synchrony is the assumption that they run in lockstep [16].

In what follows, we assume synchronous, message-passing process algebra, using the so-called early semantics for modelling communication [19], with probabilistic choices. Like PEPA, the calculus allows for multiple inter-process communications, using a CSP like composition operator. However, like WSCCS, the calculus is synchronous, choices are quantified with weights. Complementary actions, forming the basis for communication, do not yield $\angle$, the unobservable action. The process algebra does not include mobility either and so here we constrain ourselves to a modest, limited but intuitive model, which involves only a single site.

Given that we consider a singled-site system, our model of the immune system does not comprehend clustering either. Clustering requires specifying coordination and competence protocols, which would necessarily increase the complexity of the model.

This paper extends [23]. A refined version of the intended process algebra is presented in [20] at a more elaborate level of detail.

3. A Synchronous, Value-Passing CCS, an Extended Version

We assume value expressions, e, built from value variables x, y, ..., value constants v1, v2, ... and any other operators, e.g., +, ×, ...We also assume Boolean expressions, b, with similar properties except that they are closed under the logical connectives. The set of actions, Act, contains the set of input actions, the set of output actions and the unobservable action $\angle$, which denotes an internal, unknown activity.

Input an output actions are respectively of the form $a?x$ an a $!e$, where a$\in L$, the set of channels. We use l, l',..., to range over L, and let, K, L, ..., denote subsets of L. Actions, whether input or output, may take no parameters at all. In that case, we simply write $a?$, $a!$, ...

The set of agent expressions, $\mathcal{E}$, is defined as the smallest set that contains the agent expressions below:

- $a?x.E$, $a!e.E$, $\angle.E$, *prefixes* (where a is a channel, x an input variable, and e is an output expression);
- $w_1{:}E_1 + w_2{:}E_2$, *choice* (w_1 an w_2 weights);
- $E_1 \mid \{ L \} \mid E_2$, a *composition*($L$ a set of channels);
- $E\backslash L$, a *restriction* ($L \subseteq L$); and
- E/L, a *hiding* (L a set of channels).

Where E, Ei are already in $\mathcal{E}$.[1]

Informally, the meaning of the *combinators* is as follows: prefix, (.), is used to convey the discrete actions an agent may perform. For example, $a.E$ denotes an agent capable of executing the action a, and then evolving into E. $a?x.$ accepts any input value, binding the value variable x, whereas the output Prefix $a!e.$ sends e to the outside world.
Choice, +, disjoins the capabilities of the agents that either side of it; as soon as one performs any action, the others are dismissed. Choices are quantified: w1:E1 + w2:E2 means that we shall see w1 occurrences of E1 for each w2 of E2. We shall use $\Sigma i \in I Ei$ for the summation of all processes Ei, i$\in$I.

Parallel composition, ($\mid \{ L \} \mid$), is used to express synchronous concurrency. Let L be a set of action names, then the composite agent E1 {L} E2 denotes an agent where E1 and E2 may proceed independently but simultaneously, but may also interact one another via some action name in L. $\mid\mid$ is a special case of $\mid \{L\} \mid$, when L is empty. The set L is significant because it determines those activities where the agents are forced to interact. We shall use $\prod i \in I$ Ei for the composition of all processes Ei, i$\in$I.

Restriction, (\), is used to enforce communication as well as binding the scope of actions: E\L behaves like E, except that it cannot execute any action □ that uses a name in L. Finally, hiding, /, is used for internalizing actions: E/L is as E except that any one time it is about to perform an action in L, it will produce $\angle$ instead. We now proceed to describe the semantics of the language.

Processes are given meaning by two transition relations. One specifies the actions an agent may perform, and the other the weight with which the associated state is reached. Weights are associated with choices, representing probabilistic branching. They propagate as choices are performed.

We are now ready to present our model of an immune system. The model abstracts out only details about behaviour. We leave out some components, including cytotoxic T-cells—which are a type of lymphocyte—, antigens, antibodies and infected cells. The complete model is available upon request by sending electronic-mail to the second author.

4. A Behavioural Model for an Immune System

This section presents part of our behavioural model for an immune system. In the model, each component is idealized at least in two senses: first, it cannot be infected; and second, it cannot spontaneously die as time goes by. The first issue we have ignored both for the sake of clarity and manage-ability. The second issue is a bit more complex. To specify the corresponding mortal component, C^r, we must get the idealised agent, C, concurrently work with its own individual clock, Clock. Whenever the individual clock of an agent stops ticking, then so should the real, compound agent. This can be modeled by making C^r execute the tick action infinitely many times. Unfortunately, $\mid \{ L \} \mid$ is not enough to specify cell mortality and accordingly the process algebra must have to be extended with a new parallel composition. As it serves only this purpose, we have decided not to include other composition operator and hence model an idealized immune system.

In what follows, tick denotes the *idle* process, capable of delaying only, given by:

$$\text{tick} \stackrel{\text{def}}{=} \angle.\text{tick}$$

[1] Notice we do not include two WSCCS operators, namely: restriction and process priority decomposition.

4.1. The Natural Killer

A Natural Killer, NK, plays a major role in the protection of the body against a disease. It may get engaged in either of three main activities [26]: i) to recognize the presence of antigens or infected cells; ii) having identified it, to attack the non-self until possibly extinguishing it; and iii) to regulate the immune response. These activities are all kinds of interactions.

On non-self recognition, a natural killer will start attacking immediately. The attack will be either direct, if the non-self is an infected cell, or indirect, if it is an antigen, via an specific antibody. On cell attack, the interaction takes the form of toxin injection from the NK into the non-self and is usually lethal. On response regulation, the interaction amounts to cytosine injection from the NK into the blood stream. This either suppresses or stimulates the development of T-lymphocytes, each of which may inherit different features. Also on response regulation, an NK may proliferate [26].

Our process algebra model of the natural killer is given in Table 1. There, NK(R,A) means an NK with reception capabilities R.[2] and antibody population A (initially empty). As the model states, an NK is initially ready to detect the presence of either an antibody or an infected cell. If an antibody is detected, encrust, the NK will absorb it and then specialize itself as a terminator of the specific antibody's antigen. This process, called affinity maturation, we model simply using memorization, via a set.

$$\mathtt{NK}(R, A) \stackrel{\text{def}}{=} 1{:}\,\mathtt{encrust}?x.\ \mathtt{NK}(R, A \cup \{x\}) + 1{:}\,\mathtt{bind}?y.(\ \textbf{if}\ (y \in R \cup A)\ \textbf{then}\ \mathtt{NK'}(R, A)\ \textbf{else}\ \mathtt{NK}(R, A)) + 1{:}\,\angle.\mathtt{NK}(R, A)$$

$$\mathtt{NK'}(R, A) \stackrel{\text{def}}{=} 1{:}\mathtt{txn}!.\mathtt{NK''}(R, A) + 1{:}\mathtt{cts}!\uparrow.\ \mathtt{NK'''}(R, A)$$

$$\mathtt{NK''}(R, A) \stackrel{\text{def}}{=} w_0{:}\mathtt{die}?.\mathtt{NK}(\mathrm{R,A}) + w_1{:}\mathtt{txn}!.\mathtt{NK''}(R, A) + w_2{:}\,\angle.\mathtt{tick}\quad (w_0 \gg w_1 \gg w_2)$$

$$\mathtt{NK'''}(R, A) \stackrel{\text{def}}{=} \mathtt{cts}?x.\mathtt{NK'''}(R, A, x) \qquad \alpha$$

$$\mathtt{NK'''}(R, A) \stackrel{\text{def}}{=} \textbf{if}\ (x{=}\downarrow)\ \textbf{then}\ \mathtt{NK}(R, A)\ +\ \textbf{if}\ (x{=}\uparrow)\ \textbf{then}\ \mathtt{NK'}(R, A)\ |||\ \prod \mathtt{NK'}(R, \varnothing)$$

Table 1. Model of the Natural Killer, NK

Upon non-self detection, bind, the cell will either attempt to kill it, txn, but may react according to existing immune stimulus, cts. While attacking it, an NK may successfully get rid of the antigen, die, or may get exhausted, $\angle$, dying immediately afterwards. If it injects cytosine into the blood stream, an NK will either extinguish its own response, $\downarrow$, or proliferate, $\uparrow$. In the model, α stands for an oracle function which returns how many cells ought to be left upon proliferation.

4.2 The Monocyte Cell

Monocytes aim to engulf non-self, both active and inert (debris), thereby getting rid of it. They also help other immune system components recognise the presence of invading microorganisms [15]. In particular, monocytes have the capability of functioning as an antigen presenting cell, APC, for which they first require to interact with antibodies. Table 2 displays our model of a monocyte.

$$\mathtt{M}(\mathrm{R,I}) \stackrel{\text{def}}{=} w_0{:}\,\mathtt{bind}?x.\mathtt{M'}(\mathrm{R,I},x) + w_1{:}\,\mathtt{cts}?\mathrm{y}.\textbf{if}\ (y{=}\uparrow)\ \textbf{then}\ 1{:}\mathtt{cts}!\uparrow.\mathtt{M}(\mathrm{R,I}) + 1{:}\,\mathtt{bind}?x.\mathtt{M'}(\mathrm{R,I},x)\ \textbf{else}\ \mathtt{M}\ (\mathrm{R,I}) + w_2{:}\,\mathtt{apc}?.(\textbf{if}\ (\ \mathrm{I} \neq \varnothing\ \textbf{then}\ \mathtt{apc}!(x \in \mathrm{I}).\ \mathtt{M}(\mathrm{R,I}\text{—}\{x\})\ \textbf{else}\ \mathtt{fail}!.\mathtt{M}(\mathrm{R,I})) + w_3{:}\,\angle.\mathtt{Macrophage}\quad (w_o > w_1 > w_2 > w_3)$$

$$\mathtt{M'}(\mathrm{R,I},x) \stackrel{\text{def}}{=} \textbf{if}\ (x \in \mathrm{R})\ \textbf{then}\ \mathtt{Mp}(\mathrm{R,I},x)\ \textbf{else}\ \mathtt{M}(\mathrm{R,I})$$

$$\mathtt{Mp}(\mathrm{R,I},x) \stackrel{\text{def}}{=} w_0{:}\,\mathtt{engulf}!.\mathtt{die}?\ .\mathtt{M'p}(\mathrm{R,I},x) + w_1{:}\,\angle.\mathtt{tick}\quad w_0 = DC \times w_1$$

$$\mathtt{Mp'}(\mathrm{R,I},x) \stackrel{\text{def}}{=} \mathtt{bind}?y.(\ \textbf{if}\ (\ x = y)\ \textbf{then}\ \mathtt{Mp}(\mathrm{R,I},x)\ \textbf{else}\ \mathtt{M}(\mathrm{R,I} \cup \{x\})$$

Table 2. The Model of a Monocyte

[2]The reception capabilities are different from one lymphocyte to other. Moreover, they improve when a *major histocompatibility complex*, *MHC*, molecule enables the lymphocyte to detect an invading microorganism residing inside a cell.

As indicated by the model,[3] monocytes require an stimulus, namely: cytosine, in order to bind and then absorb a non-self. Upon detection, monocytes absorb as much non-self as possible but they may die if they exceed their digestion capability. Sometimes, a monocyte moves from the blood stream into the muscular tissue and, hence, differentiates into another cell, called *macrophage* [26]. Macrophages are just as monocytes except that they are bigger and may swell an infected area.

4.3 The B Lymphocyte

B cells play a chief role at yielding the acquired immune response [26]. They are concerned with 4 main activities: i) getting rid of the non-self; ii) learning from past attacking activities; iii) providing a repository for documenting experienced attacks; and iv) tuning the cell reception capabilities, via either antibody encrustation or through incoming cytosine. In Table 3, B(R,A, I) represents a B cell with reception capabilities R, population of antibody encrustation A and repository of previous attacks I.

$\mathtt{B}(R,A,I) \triangleq w_0\colon \mathtt{cts}?y.\mathtt{B}(R \cup \{y\}, A, I)$
$\quad + w_0\colon \mathtt{encrust}?x.\mathtt{B}(R, A, \cup \{x\}, I)$
$\quad + w_1\colon \mathtt{endocyte}?z.\mathtt{B}(R,A,\sigma(I))$
$\quad + w_1\colon$ **if** $(I \neq \varnothing)$ **then** $\mathtt{query}!\ (u \in I).\mathtt{B}(R,A,I))$
$\quad + w_0\colon \mathtt{bind}?v.\mathtt{B}(R,A,I,v)$
$\quad + 1\colon \angle.\ \mathtt{B}(R,A,I)$

$\mathtt{B}(R,A,I,v) \triangleq$ **if** $(v \in R \cup A)$ **then** $\mathtt{engulf}!.\mathtt{die}?\mathtt{P}(R,A,I,v)$
$\quad \alpha$
$\quad$ **else** $\mathtt{B}(R,A,I) \mid \prod_{i=1}^{\alpha_0} \mathtt{Ab}(\varepsilon)$

$\mathtt{P}(R,A,I,x) \triangleq \tau.\mathtt{Bm}(R,A,I,y) \mid \{\mathtt{cts},\mathtt{encrust}\} \mid \prod_{i=0}^{\alpha_1} \mathtt{Ab}(y)$

$\mathtt{Bm}(R,A,I,y) \triangleq w_0\colon \mathtt{cts}?.\ (\prod_{i=0} \mathtt{P}(R,A,I,y) + \mathtt{Bm}'(R,A,I,y))$
$\quad + w_1\colon \mathtt{endocyte}?z.\mathtt{Bm}\ (R,A,I,\sigma(I))$
$+$ **if** $(I \neq \varnothing)$ **then** $w_1\colon \mathtt{query}!\ (u \in I).\mathtt{Bm}\ (R,A,I)$
$+ 1\colon \angle.\ \mathtt{B}(R,A,I,y)$

$\mathtt{Bm}\textcircled{i}\ (R,A,I,z) \triangleq 1\colon \mathtt{cts}!\uparrow.\mathtt{Bm}\textcircled{i}\ (R,A,I,z) + 1\colon \mathtt{cts}!\downarrow.\mathtt{Bm}\ (R,A,I,z)$
$\quad (w_0 \gg w_1 \gg 1)$

Table 3. Definition of a B cell

The model expresses that once the antibody-antigen complex has been formed, a B cell may engulf it and then endocyte it, endocyte, hence, modifying the repository of known attacks, $\sigma(I)$. The information in the repository, I, can be consulted any other time, query. The size of the repository increases monotonically, upon endocytosis.

When its receptors bind to an specific antigen, bind, a B cell differentiates into a plasma cell, P(R,A, I, x). Then, the plasm cell will immediately yield a fixed but arbitrary number of antibodies, specific to the detected antigen, ending up as a B memory cell.

The behaviour of a B memory cell is similar to that of a B cell. The main difference being that the B memory cell is involved in the regulation of the immune response. After the reception of cytosines, a B memory cell will, in turn, inject cytosines into the blood stream. This will result in a number of interactions whereby the cell may either proliferate or suppress the immune activity. Notice that once differentiated, a B memory cell will never go back to its original, B cell state.

[3] *DC* amounts to the digest capability of an standard monocyte. So if it goes beyond *DC*, the monocyte may die.

4.4 The Th Lymphocyte

The T helper lymphocyte, Th, aims at speeding up the immune response [26]. It has a special, unique receiver, which is different to other lymphocytes'. With its unique receiver, the Th cell is able to recognize an antigen, provided that the antigen has formed an MHC-antigen complex already.

Detection of an MHC-antigen complex is not enough for a Th to help the immune system to increase the response. It further requires the assistance of an *antigen presenting cell*, *APC*. Unless this assistance arrives, the Th will go unnoticed. In the normal case of successful activation, the Th cell is able to stimulate other cells' response, both by injecting cytosine in the blood stream and, provided the right feedback is detected, by proliferation. Upon reception of negative feedback, the Th will rest going back to its initial state. Our model of a Th cell is given in Table 4.

$$\mathtt{Th} \triangleq \mathtt{mhc\text{-}ag}?x.\mathtt{Th}(x)$$

$$\mathtt{Th}(x) \triangleq \mathtt{apc}!x.\mathtt{apc}?y.(\textbf{if } (y = \text{fail}) \textbf{ then } \mathtt{Th}(x) \textbf{ else } \mathtt{Th'})$$

$$\mathtt{Th'} \triangleq \mathtt{cts}!\uparrow.\mathtt{Th'} \;\; \underset{\alpha}{+} \; \mathtt{cts}?x.(\textbf{if } x = \uparrow) \;|||\; \prod_{i=0} \mathtt{Th'} \textbf{ else } \mathtt{Th}$$

Table 4. Model of a Lymphocyte type Th

4.5 The Ts Lymphocyte

The T suppressor lymphocyte, *Ts*, aims to slow down the immune response, injecting cytosine with negative stimulus into the blood stream. Given that it has no receptors, the Ts cannot detect the presence of invading microorganisms and it therefore does not need toxin injection capabilities. To become active, a Ts needs to get cytosine with positive stimulus from the blood stream. Then, it further emphasizes deactivation of immune response, injecting itself negative cytosine and then proliferating. The high concentration of negative cytosine will eventually give rise to a immune response decay. Our model of the Ts lymphocyte appears in Table 5.

$$\mathtt{Ts} \triangleq \mathtt{cts}?x.\mathtt{Ts}(x) \; + 1: \angle.\, \mathtt{Ts}$$

$$\mathtt{Ts}(x) \triangleq (\textbf{if } x \neq \uparrow) \textbf{ then } \mathtt{Ts} \textbf{ else } (\mathtt{cts}!\downarrow.\; |||\; \prod_{i=1}^{\beta} \mathtt{Ts})$$

Table 5. Ts Definition

4.6 The Blood Stream

Apart from other substances, such as micro- and macro-nutrients, the blood stream carries the immune system components and provides a means of component intercommunication. Thus, the blood stream is vital to the immune system. In this paper, we model the blood stream as an agent which may input or output information, via cytosine—see Table 7—. Cytosine leaves from and arrives to the blood stream in no specific order, so BloodStream is simply modelled as a computer bag with infinite capacity (see Table 6)

$$\begin{aligned}\mathtt{BloodStream}(S) \triangleq\; & 1: \mathtt{cts}?x.\mathtt{BloodStream}(S \cup \{x\}) \\ & + 1: (\textbf{if } (S \neq \varnothing) \textbf{ then } \mathtt{cts}!(e \in S).\mathtt{BloodStream}\ (S - \{e\})) \\ & + 1: \angle.\mathtt{BloodStream}\ (S)\end{aligned}$$

Table 6. Blood Sream Definition

To complete our model for the immune system, we need only to compose the elements and then properly link them. To approach composition, we first look into main components:

NS | {bind, txn, engulf, die} |ISC|{cts}|BloodStream

Where NS stands for non-self components, antigens and infected cells, and ISC for immune system components, lymphocytes, monocytes and antibodies, properly linked.

The initial population of each agent type will be different and so should be set according to standard literature. We specify an immune system at an idle state by omitting antigens and infected cells. Clearly, agent population and links, called the *configuration* of the system, can be set according to the type of analysis to be conducted.

With this, we complete the description of our model of the immune system. Now we now give attention to formally specifying its associated properties.

5. Immune System Analysis

An immune system aims to protect the body against invading microorganisms. It portrays a number of properties one would like to have in a contemporary computer system application, for instance effectiveness, reliability, robustness, adaptability and self-stabilization—to mention a few—. This section discusses how to provide a formal specification about some of these properties. It also outlines the sorts of mechanisms that would be required to prove whether or not the proposed model enjoys any one of these properties.

Effectiveness is achieved when the system gets rid of both all antigens and all infected cells. Consider that, when dying, any one invader issues the action die (see §4). Then effectiveness can be succinctly specified using the μ-calculus [25], as shown below:

$$IS \models Ev\ (die)$$

where IS and |= respectively denote the immune system model and a satisfaction relation, and where Ev(die) is given by:

$$Ev(die) = \mu X.\ \langle\rangle tt \wedge [-die]X$$

Roughly, this means that IS can do no action sequence where die does not appear.

The above formula guarantees only that one invader will die. To handle an arbitrary number of invaders, we ought to use nested applications of Ev. Since this is a bit awkward, we appeal to an alternative specification method.

Recall that the configuration of a system is given by the size of each component's population, together with the link structure (see §4.6). Upon action execution, the configuration changes. We take action execution to be a language generation process: The output language being the set that contains all possible configurations. Then, effectiveness can be modelled as follows: Given an initial configuration with a non-empty population of invaders, the system will eventually become idle, getting rid of the non-self. Effectiveness can be formally verified using methods borrowed from probabilistic process calculi: A system is effective if the expected value of non-self population tends to zero.

Using the notion of configuration, we can model other properties self stabilization and robustness. Self-stabilization is achieved if, after activity, the immune system goes back to a configuration similar to the initial one. Put differently, an idle system does not evolve. Thus, the expected size of the configuration of an infected immune system should tend to the size of that configuration after eliminating the non-self.

Robustness is achieved if, no matter the population of a particular component, the system does not loose effectiveness. Thus, the expected value of non-self population should tend to zero, even though we omit part of the immune system population from the initial configuration.

There are of course properties, such as a adaptability, that cannot be even specified within our model. That would take giving a full account of learning.

To formally analyze our system, we only need existing proof methods. However, these methods ought to be re-implemented so that they can be used to analyze partial system behaviour. This is because, prior to verification, existing methods first compute the entire system behaviour. Yet this is not always possible, especially when the system space is infinite. The space associated with our immune system is essentially infinite and so is the language it generates. This is because, the configuration evolves dynamically. Accordingly, we need to do our observations along a predefined number of transitions, ticks of a universal clock.

6. Related Work

Hofmeyr and Forrest have invented ARTIS, an architecture for immune computer systems [13]. While ARTIS involves the basic ingredients of an immune system, the authors o not develop the underlying model. Rather they include only operational ideas associated with immunology—attack, detection, learning and rejection—.

In [5], the authors suggest a methodology for immune system architecture design. Roughly, the design of an architecture should be based on the informal requirements associated with the target immune system. These requirements must be first matches against a natural immune system, identifying responsible mechanisms. Then, given the isolated mechanisms, the authors suggest feasible implementations.

Artificial Immune System, AIS [1], portrays a collection of mechanisms that combine characteristics found in some elements of a natural immune system (cells and proteins). The mechanisms are used to address intrusion detection, so the underlying model does not consider the entire variety of system elements. A similar argument applies to both [4]an [3,pages 242-261]. Other applications of immune system phenomena can be found in [24].

More related to ours is the work of Hatcher, Tofts an Dunn [9, 8, 7]. They have applied WSCCS to computer modelling issues of biological systems, such as sex ratio, sex determination and parasite transmission rate. Focusing on a very specific problem, they have been able to formally analyze or simulate the behaviour of biological systems. Their papers have been a source of inspiration for the work presented here. This paper extends [23].

7. Conclusions

We have presented a model for an immune system based in process algebra. Our model involves only aspects of behaviour, ignoring other important subtleties, such as learning a detection. Our model can be used to analyze immune system's expected population, with which we can address properties like effectiveness, robustness and self-stabilization. To conduct formal analysis of our system, however, we need to extend existing techniques so as to incorporate the analysis of partially developed in infinite-state systems. While this development is time consuming, the associated payoffs are worthwhile. We reckon the resulting tool could be used as a controlled laboratory by biologists.

Acknowledgements

This research was partially supported by CONACYT grants CONACYT-BMBF J200.1442/2002 and 33337-A.

References

[1] R. Belew and S. Forrest. Learning classifier systems, from foundations to applications. In, P.L. Lanzi, W. Stolzmann, and W.Wilson S, editors, Proceedings of IWLCS '99, volume 1813 of Lecture Notes in Computer Science. Springer, 2000.

[2] J. A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Compute Science*, 37(1):77—121, 1985.

[3] D. Dasgupta. *Artificial Immune System and Their Applications*. Springer, U.S.A.,1998.

[4] P. De Haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: Algorithms analysis an implications. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pages 110—119. IEEE Computer Society Press, 1996.

[5] S.Forrest, S.A. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the Association for Computing Machinery*, 40(10):88—96, 1997.

[6] S. Gilmore, J. Hillston, and M. Ribaudo. An efficient algorithm for aggregating PEPA models. *IEEE Transactions on Software Engineering*, 27(5):449—464, May 2001.

[7] M.J. Hatcher, A.M. Dunn, and C. Tofts. The effect of the embryonic bottleneck on vertically transmitted parasites. In *Proceedings of 1st International Conference on Information Processing in Cells and Tissues*, page In Press. University of Liverpool, 1996.

[8] M.J. Hatcher and C. Tofts. The effect of point of expression on ESS sex ratios. *Journal of Theoretical Biology*, 175:263—266, 1995.

[9] M.J. Hatcher and C.Tofts. The evolution of polygenic sex determination with potential for environmental manipulation. Technical Report Series, Department of Computer Science, UMCS-95-4-2, University of Manchester, 1995.

[10] M. Hennessy and H. Lin. Proof systems for message passing process algebras. *Formal Aspects of Computing*, 8(4):379—407, 1996. Also available from Sussex as Computing Science Technical Report 3/93.

[11] Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[12] C.A.R. Hoare. Communicating sequential processes. *Communications of the Association for Computing Machinery*, 21(8):666—77, 1978.

[13] S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation Journal*, 8(4):443—473, 2000.

[14] ISO. *Information processing systems—Open systems interconnection*. LOTOS—*A formal description technique based on the temporal ordering of observational behaviour*. ISO 8807, 1989.

[15] R. López, F. Díaz, and S. Arias. *Biología Celular*. Editorial Iberoamérica, México, 1991.

[16] R. Milner. *Communication and concurrency*. Prentice Hall, London, 1989.

[17] R. Milner, J. Parrow, and D. Walker. A calculus for mobile processes, part 1. *Information and Computation*, 100(1):1—40,1992. Also available from Edinburgh, as Research Report ECS-LFCS-89-95.

[18] R. Milner, J. Parrow, and D. Walker. A calculus for mobile processes, part 2. *Information and Computation*, 100(1):41—77, 1992. Also available from Edinburgh, as Research Report ECS-LFCS-89-96.

[19] R. Milner, J. Parrow, and D. Walker. Mobile logics for mobile processes. *Theoretical Compute Science*, 114:149—71, 1993. Also available from Edinburgh, as Research Report ECS-LFCS-91-136.

[20] Raúl Monroy. A process algebra model of the immune system. In M.G. Negoita, editor, *Proceedings of the 8th Knowledge-Based Intelligent Information & Engineering Systems, KES`04*. Lecture Notes in Artificial Intelligence, Vol. (to appear) , pages (to appear), Wellington, New Zealand, September 2004. Springer-Verlag.

[21] J. Parrow and V. Björn. The fusion calculus: Expressiveness an symmetry in mobile processes. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Compute Science*, pages 176—185, 1998.

[22] S. Robbins and R. Cotran. *Pathologic basis of disease*. W B Saunders Co, U.S.A.,1984.

[23] R. Saab, R. Monroy, and F. Godínez. Towards a model for an immune system. In, C.A. Coello-Coello, A. De Albornoz, L.E. Sucar, and O. Cairó, editors, *2nd Mexican International Conference on Artificial Intelligence*, *MICAI '02*, pages 401—410. Springer-Verlag, 2002.

[24] L.A. Segel and I.R. Cohen. *Design principles for the immune system and other distributed autonomous systems*. Oxford University Press, New York, U.S.A., 2000.

[25] C. Stirling. An Introduction to modal an temporal logics for CCS. In *1989 UK/Japan workshop on concurrency*, Lecture Notes in Conputer Science, v.491, pages 2—20. Springer-Verlag, 1990.

[26] D. Stites and T. Abba. *Medical immunology*. McGraw-Hill, 1997.

[27] C. Tofts. Processes with probabilities, priority and time. *Formal Aspect s of Computing*, 6(5):536—564, 1994.

Raúl Monroy. *He obtained a PhD in Artificial Intelligence in 1998 from Edinburgh University, under the supervision of Prof. Alan Bundy. He has been in Computing at Tecnológico de Monterrey (ITESM), Campus Estado de México, since 1985. In 1992 he was promoted to Assistant Professor and in 2000 he was promoted to Associate Professor. Since 1998 he is a member of the CONACYT-SNI National Research System. Dr. Monroy's research focuses on automating the use of theorem proving to formal methods of system development. Currently, his research concerns: the discovery and application of proof plans to automate the verification of security protocols; the discovery an application of general search control strategies for uncovering and correcting errors in either a system or its specification; the discovery of novel methods for anomaly detection in computer security; and the computer modelling of immune system behaviour. Dr. Monroy has been sole or joint holder of 5 CONACYT, NSF, BMBF grants and is the sole or joint author of over 30 published papers. He was programme co-chairman for MICAI-2004 and has been Secretary Treasurer to the Mexican Society for Artificial Intelligence.*

Rosa Saab. *She obtained an MSc in Computer Science from ITESM campus Estado de Mexico in 2002. She is currently at Tecnológico de Estudios Superiores de Coacalco (TESCo), where she conducts both research and administration work. As head of department, she runs projects related to quality in university education and manages third party certifications for TESCo graduate and undergraduate programmes. She has published several papers, one of them obtained the best paper MICAI 2002 award.*

Fernando Godínez. *He is a research assistant at ITESM-CEM and member of the Networking and Security research group. He is currently enrolled as a Ph.D. student at ITESM-MTY and expects to complete his studies in Fall 2004. In his Ph.D. dissertation, Mr. Godínez proposes a methodology for intrusion detection based on session analysis. Mr. Godínez obtained a B.Sc. in electronics and computer systems at ITESM-CEM (1999). His research interests include probabilistic pattern recognition methods, multi-agent systems, process algebras and data mining.*