Multiformalism modeling and simulation of immune system mechanisms

Elvio Gilberto Amparore†, Marco Beccuti†, Paolo Castagno†,
Giuliana Franceschinis*, Marzio Pennisi*, and Simone Pernice†,
* *Computer Science Institute, DiSIT, University of Eastern Piedmont, Alessandria, Italy*
† *Department of Computer Science, University of Torino, Torino, Italy*

***Abstract*—The immune system (IS) represents a complex network of cells and molecules devoted to the protection of individuals from external pathogens, and in terms of complexity, it is only second to the central nervous system. As our knowledge of the IS mechanisms has become more exhaustive, interest has grown in applying modeling and simulation techniques in this context. In particular, among these techniques, the Agent Based Models (ABMs) have been increasingly applied for the IS simulation. One of the major drawbacks of ABMs is represented by the lack of well-defined semantics, which may lead to inconsistent results in comparison to other stochastic approaches. In this paper, we make use of the well-defined semantics and the simulation algorithm for ABMs that we proposed in [1] to implement a few models of the Cancer-Immune System. Comparing ABMs and Gillespie's Stochastic Simulation Algorithm results we show that our methodology brings coherence among the results of ABMs and SSA.**

***Index Terms*—Extended Stochastic Symmetric Nets, Agent Based Models, Stochastic Simulation Algorithm, Immune system modeling.**

1. Introduction

The immune system (IS) represents a complex network of cells and molecules devoted to the protection of individuals from external pathogens. It is highly heterogeneous, multi-scale, self-organized, highly distributed, capable of recognizing, learning, and adapting to novel viruses and bacteria. Furthermore, it is able of showing important emergent behaviors such as discrimination between self and non-self, memory, and tolerance. In terms of complexity, the IS is probably only second to the central nervous system. As our knowledge of the IS mechanisms has become more and more exhaustive, a growing interest in applying in this context the modeling and simulation techniques already used in other research fields such as social sciences, engineering, and physics has caught on. IS modeling and simulation represent an important challenge, as thanks to the so-called "in silico" models it is possible to answer some important questions such as: what are the possible causes of diseases, and how do they proceed? what are the best treatments to be used? what is the best administration strategy for such treatments? will a novel treatment be effective against the disease? and so on. Moving forward, the use of "in silico" models and approaches has recently been advocated also by some regulatory agencies such as EMA and FDA.

There are essentially two main strategies that can be applied for the simulation of IS, i.e. deterministic and stochastic simulation approaches. The former are used to analyze the system in terms of mean behavior, while the latter account for certain levels of unpredictability or randomness leaving open the opportunity of considering the environmental variability that deterministic models cannot catch up, e.g. for optimal drug dosage for personalized medicine.

Among the various stochastic approaches, Agent Based Models (ABMs) have been increasingly applied for the immune system simulation thanks to their ability of describing the involved immunological phenomena in a natural and detailed way, their clarity and comprehensibility for the domain experts (i.e., immunologists, biologists, medical doctors and pharmacologists), and their capability of describing the phenomena at the cellular level bringing global results at the organ/tissue level through their intrinsic aptitude in unveiling emergent behaviors.

However, one of the major drawbacks of ABMs is represented by the lack of well-defined semantics, which may lead to the scenario where the same conceptual model and the same data bring to inconsistent results in comparison to other stochastic approaches, or even when using ABMs implemented with different frameworks. More specifically, we can identify two possible sources of bias that can affect the results: one can be introduced at the programming time by the freedom left to modeler in defining the way and the order in which agents' actions occur. The second source of bias has instead to deal with the underlying simulation platform used to implement the model (i.e., Netlogo, Flame, Anylogic, and so on [2]–[5]). Indeed, these platforms do not always follow the same policies for determining the order in which agents will be selected to carry on their actions. While the first source of bias can be somewhat smoothed by good modeling and programming skills, the second source is harder to manage, as the modeler usually has little to no control over those aspects. Such lack of clearness in the definition of unique semantics makes the results dependent on the modeling platform used and may lead to the belief that ABMs may show different results with respect to other stochastic approaches, as presented for example in the work

978-1-6654-0126-5/21/$31.00 ©2021 IEEE

by Figueredo et al [6].

To clarify if ABMs can really bring to different results or not, we will make use in the present paper of the well-defined semantics and the simulation algorithm for ABMs that we proposed in [1] to re-implement the Cancer-Immune System models presented in [6]: the obtained results allow us to conclude that ABMs produce results that are coherent with other stochastic approaches, at the same time providing a basis for observing the interactions and their effects at a micro-level in addition to the emergent trends at the macro-level of the whole population.

Such a coherence among the results may become particularly relevant in various situations, for example when we plan to move towards different platforms that exploit different hardware (e.g. from CPU to GPU simulation), when we want to combine different approaches for different scales (i.e., intracellular pathways modeled with Petri Nets, and cellular behavior modeled with agents), or when we want to interchangeably use different stochastic approaches to observe the system from different points of view.

The paper is organized as follows: in Section 2 we will briefly introduce the models presented in [6] first, the Petri Net formalism, and the GreatMod framework then. Section 3 drafts the automatic translation algorithm from Petri Nets to ABMs, while section 4 will compared the obtained results. Finally, in Section 5 final considerations will be drawn.

2. Background

In this section, we first present the three systems studied in our experiments. Then, we introduce the PN formalism and the GreatMod framework used to model and analyze such systems.

2.1. The models

In this section, we briefly introduce the three different immunological systems presented in [6]. The former simply represents the interactions between Tumor Cells (Tcells) and Generic Effector Cells (ECells). In particular, it is modeled (i) the control action of the ECells by killing the Tcells, (ii) the Ecell death by apoptosis and their proliferation, which increases proportionally with the number of Tcells, (iii) the proliferation of Tcells, and (iv) a possible tumor treatment defined by an injection of ECells. We refer to this model as *TCells v.s. ECells*, and the mathematical details of these events are summarized in Table 1.

The second system considered extends the previous one explicitly modeling the cytokine IL2 (interleukin-2), an immunomodulating molecule released for a self-stimulation of ECells to duplicate and propagate. Specifically, in this system, the IL2 are modeled as molecules mediating the immune response towards tumor cells. The mathematical details regarding the events modeled in this case are summarized in Table 2, and we refer to this model as *TCells v.s. ECells with IL2*.

Eventually, the third system is derived from the second one by specifically introducing the *transforming growth factor beta* (TGF-β), i.e. a multifunctional cytokine that stimulates tumor growth and suppresses the immune system by inhibiting the activation of ECells and reducing tumor antigen expression. The mathematical details regarding the events modeled in this case are summarized in Table 3, and we refer to this model as *TCells v.s. ECells with IL2 and TGF-β*.

2.2. Petri Net formalism

PNs and their extensions are effective formalisms to model biological systems thanks to their capability of representing simply and clear way the system features and providing efficient techniques to derive system qualitative and quantitative properties.

Among the PN formalisms, to study the system evolution characterized by complex rate functions and to obtain a more parametric and readable representation of the system, in this paper we focus on Extended Stochastic Symmetric Nets (ESSNs) [7]. In details, an ESSN is a bipartite graph whose nodes are *places*, and *transitions*. Places, graphically represented as circles, denote a system local state while transitions graphically represented as boxes, encode the system events. Places and transitions are connected by directed and annotated *arcs*, which express the relation between states and event occurrences.

For instance the ESSN model of the first system *TCells v.s. ECells* showed in Fig. 1 has two places, i.e. *AliveT* and *AliveE*, representing Tcells and Ecells respectively. Furthermore, in this model seven events are explicitly represented through the transitions *birthT*, *deathT_e birthE*, *deathE*, *deathT_e* and *deathE_t* . In detail the first four transitions model proliferation and cellular decay of the Tcells and Ecell respectively, while the last two transitions model the killing of a cancer cell by Effector cells and the damage underwent by Ecells when interacting with Tcells.

Places can contain tokens, and the global number of tokens in each place defines the state of a PN, namely a *marking*. Furthermore, it is possible to associate each token with different characteristics by having tokens with colors. Specifically, a place p can hold tokens belonging to the place color domain $cd(p)$ where color domains are defined by the Cartesian product of elementary types called *color classes*, $\mathcal{C} = \{C_1, \ldots, C_n\}$, which are finite and disjoint sets, and might be further partitioned into (static) subclasses. For instance, in the model in Fig. 1, to distinguish the tokens in the same place two color classes are defined: *Ecells* for the place *aliveE* ($cd(aliveE) = Ecells$), and *Tcells* for the place *aliveT* ($cd(aliveT) = Tcells$).

Similarly, a color domain is associated with each transition and it is defined as a set of typed variables where the variables are those appearing in the functions labeling the transition arcs and their types are the color classes. Thus, var assigns to each transition $t \in T$ a set of variables, each taking values in a given element C_i of $\mathcal{C}$ (the variable's type); fixed order on the set of variables, the color domain of t, $cd(t)$, is defined as the Cartesian product of its variables' types. Specifically, considering the transition

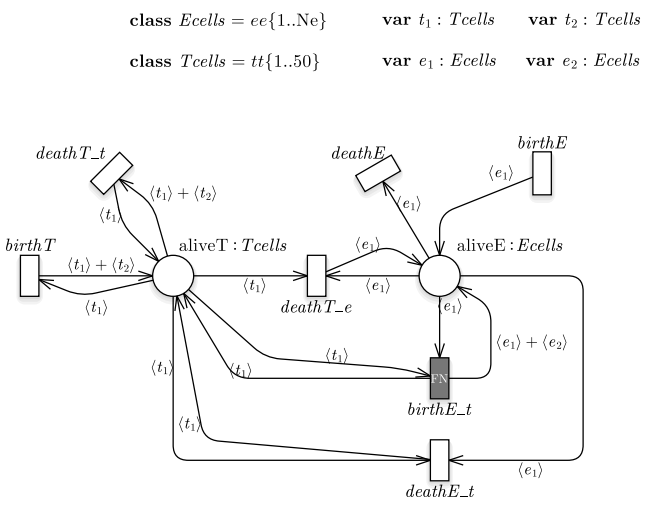

Figure 1. ESSN modeling the *TCells v.s. Ecells* system characterized by the interaction of the Tumor Cells (place *aliveT*) and Effector Cells (place *aliveE*).

deathT_e modeling the TCells killed by the ECells, its color domain is defined as $cd(deathT\_e) = Tcells \times Ecells$, with variables named t_1 and e_1 representing one Tcell killed by one Ecell, which returns in the origin place *aliveE* after its control action against the Tcell.

Then, we can define an instance, denoted as $\langle t, c\rangle$, of a given transition t as an assignment (or binding) c of the transition variables to a specific color of a proper type. Arcs are annotated by the functions $I[p,t]$, if the arc connects a place p to a transition t, or $O[p,t]$ for the opposite direction. The evaluation of $I[p,t]$ (resp. $O[p,t]$), given a legal binding of t, provides the multiset of colored tokens that will be withdrawn from (input arc) or added to (output arc) the place connected to that arc by the firing of such transition instance. The set of input/output places of transition t is denoted by $^\bullet\mathbf{t}$ / $\mathbf{t}^\bullet$. It is also possible to associate specific *guards* with transitions: a guard is a logical expression defined on the color domain of the transition, which can be used to define constraints on its legal instances. A transition instance $\langle t, c\rangle$ is enabled and can fire in a marking m, if its guard evaluated on c is true, and for each input place p we have that $I[p,t](c) \leq m(p)$, where $\leq$ is the comparison operator between multisets. Thus, the firing of an enabled transition removes a fixed number of tokens from its input places and adds a fixed number of tokens into its output places (according to the cardinality of its input/output arcs). In details, the firing of $\langle t, c\rangle$ in m produces a new marking m' such that, for each place p, we have $m'(p) = m(p) + O[p,t](c) - I[p,t](c)$. The set of all instances of t enabled in marking m is denoted by $E(t,m)$. For instance, the firing of the transition instance $\langle deathT\_e,\ t_1,\ e_1\rangle$ removes the Tcell associated with the color t_1 from the place *aliveT*.

Finally, a velocity is defined representing the parameter of the exponential distribution modeling the random firing time for each transition. Considering the ESSN, the set of transition T is divided into two subsets T_{ma} and T_g depending on the associated velocity. The former subset contains all transitions firing with a velocity defined by the Mass Action (MA) law [8]. The latter includes all transitions whose random firing times have velocities that are defined as general real functions. Hence, we will refer to the transitions belonging to T_{ma} as standard transitions and as general transitions those in T_g. This allows to easily model events that do not follow the MA law but have more complex functions. Let us define $\hat{m}(\nu) = \mathbf{m}(\nu)_{|^\bullet\mathbf{t}}$, where the notation $m_{|P_i}, P_i \subset P$— denotes the projection of the marking m on a subset P_i of places; the rate parameter associated with an enabled transition instance $\langle t, c\rangle$ is given by the function

$$F(\hat{m}(\nu), t, c, \nu) := \begin{cases} \varphi(\hat{m}(\nu), t, c), & t \in T_{ma}, \\ f_{\langle t,c\rangle}(\hat{m}(\nu), \nu), & t \in T_g, \end{cases} \quad (1)$$

In particular, $\varphi(\mathbf{m}(\nu), t, c)$ is the MA law, i.e.

$$\varphi(\mathbf{m}(\nu), t, c) = \omega(t,c) \prod_{\langle p_j, c'\rangle |\ p_j \in {}^\bullet\mathbf{t}\ \wedge\ c' \in cd(p_j)} m[p_j][c'](\nu)^{I[p_j,t](c)[c']} \quad (2)$$

with $\omega(t,c)$ the MA constant rate parameter of the enabled transition instance $\langle t, c\rangle$.

A simple example of general transition is *birthE_t*, whose velocity is described by a more complex law,

$$f_{\langle birthE\_t,\ t_1,\ e_1\rangle}(\hat{m}(\nu), \nu) = \frac{p\ m_{aliveT}(\nu)\ m_{aliveE}(\nu)}{g + m_{aliveT}(\nu)} \quad (3)$$

where p, g are constants, and $m_{p_i}(\nu)$, $p_i \in \{aliveT,\ aliveE\}$ is the number of tokens in place p_i at time ν.

Observe that $\varphi(\hat{m}(\nu), t, c)$ and $f_{\langle t,c\rangle}(\hat{m}(\nu), \nu)$ can depend only on the time ν and the marking of the input places of transition t at time ν. Stochastic firing delays, sampled from a negative exponential distribution, allow one to automatically derive the underlying CTMC that can be studied to quantitatively evaluate the system behaviour [9]. In details, the CTMC state space, $\mathbb{S}$, corresponds to the reachability set of the corresponding ESSN, i.e. all possible markings that can be reached from the initial marking. The Master equations (MEs) for the CTMC are defined as follows:

$$\frac{d\pi(\mathbf{m}_i, \nu)}{d\nu} = \sum_{\mathbf{m}_k} \pi(\mathbf{m}_k, \nu) q_{\mathbf{m}_k, \mathbf{m}_i} \qquad \mathbf{m}_i, \mathbf{m}_k \in \mathbb{S} \quad (4)$$

where $\pi(\mathbf{m}_i, \nu)$ represents the probability to be in marking $\mathbf{m}_i$ at time ν, and $q_{\mathbf{m}_k,\mathbf{m}_i}$ the element of the *infinitesimal generator* (i.e., the velocity to reach the marking $\mathbf{m}_i$ from $\mathbf{m}_k$), which is defined as follows:

$$q_{\mathbf{m}_k,\mathbf{m}_i} = \sum_{\substack{\mathbf{t}\in T \wedge \mathbf{c}' \in cd(\mathbf{t}) \wedge \\ \langle \mathbf{t}, \mathbf{c}'\rangle \in E(\mathbf{t}, \mathbf{m}_k)_{|\mathbf{m}_i}}} F(\mathbf{m}_k, \mathbf{t}, \mathbf{c}', \nu). \quad (5)$$

with $E(\mathbf{t}, \mathbf{m}_k)_{|\mathbf{m}_i}$ is the subset of $E(\mathbf{t}, \mathbf{m}_k)$ whose firing leads to marking $\mathbf{m}_i$.

In complex systems, the equations (4) are often computationally intractable and several techniques can be exploited to study the system taking into account stochasticity. The Stochastic Simulation Algorithm (SSA) [10] is an exact stochastic method used to simulate systems, whose behaviour can be described by the MEs.

2.3. Framework

GreatMod[1] is a novel computational framework for studying complex systems [11]. It encompasses both modeling tools and several computational tools for computing solutions. The approach underlying the GreatMod workflow builds upon a high-level graphical formalism called Petri Net (PN). Representing a model through a PN allows an automatic check of some structural features, for instance, the conservation of the population in a closed system. Furthermore, the PN formalism enables GreatMod to exploit its semantics to translate the same model into several solution approaches.
GreatMod encompasses both equation-based solution methods and simulation ones. The first class of solvers allows to devise the system of Ordinary/Stochastic Differential Equations (ODE/SDE) representing the system's dynamics through time and to solve it. The latter approach allows us to compute the stochastic evolution of the system in time through two different simulation approaches: Gillespie's Stochastic Simulation Algorithm (SSA) and Agent-Based Model simulations (ABMs). GreatMod encompasses both SSA and ABMs given their complementary characteristics: SSA allows one to approach the model at large—for instance, the diffusion process at a population level–ABM allows investigating the system dynamic in much greater detail— modeling the interactions among different agents.
GreatMod includes also specific features to handle experiment reproducibility. Provided that the GreatMod version and the model are the same used in previous experiments, the framework allows controlling the experiment's outcomes setting a specific seed used to rule the random numbers generation, thus leading to the same final results.

3. Automatic generation of the ABM simulation model

In this section, we briefly describe the translation steps from the ESSN model to an ABM model. The automatic translation of the PN model into an Agent-Based Model produces a code that can be executed by NetLogo [2].

NetLogo is one of the most easy-to-learn programmable modeling environments for the development of Agent-Based Models. In NetLogo, we have two types of agents, the "turtles" and the "patches". Turtles best suit the common definition of agents. Patches are special agents that cannot move, as they represent the positions of the environment on which turtles act and interact. Both can have internal variables and can execute even complex behaviors defined by the programmer using the command "ask". Commonly, two procedures are present in any NetLogo code, the "setup" and "go" procedures. While the former is used to set up the initial conditions, the latter continuously iterates the commands defined within it until certain conditions set by the user are reached. NetLogo uses a FITA (Fixed Increment Time Advance) with equally spaced time-steps managed through a "tick" counter, but the code produced by the translation algorithm will simulate a NETA (Next Event Time Advance) approach on top of NetLogo. Making a long story short, the produced code asks all the turtles to calculate their cumulative rates according to the actions they can carry, calculates the total rate as the sum of agents' rates, estimates the timing of the next event according to an exponential distribution and the total rate, and selects with a roulette-wheel method the next agent that will act. The chosen agent will randomly choose, always following a roulette-wheel method, the next action it will carry on. We remind here that for actions that require 2 or more agents, only one agent will be selected as "leading agent" and will be in charge of managing the interaction. Further details about the ABM semantics and the translation algorithm can be found in [1].

The translation is performed automatically using a module of the GreatMod framework. Net elements are translated, following these high-level principles:

- Individual tokens in the net become agents.
- Agents move between places, and the domain of each place identifies the agent attributes.
- Agent types are identified by the first color class in the color domain of the place where they are.
- Transitions encode the agent's behaviors.

To this purpose, the ESSN model must be annotated to identify which agent types are modeled (through color classes), which are their possible states (subset of places and their color domain), which transitions represent internal agents' evolution, and which instead represent the interaction among multiple agents. A primary set of elementary color classes form the set of *agent classes* $\mathcal{A} \subseteq \mathcal{C}$. All other color classes form the set of *agent attributes*, $\mathcal{B}$. Allowed place color domains have form $A[\times B_1 \dots \times B_n]$, where $A \subseteq \mathcal{A}$ is the agent type of that place, and $B_1 \dots B_n$ is the set of agent attributes describing its complete state when it stays in that place. The set of attributes may change when an agent moves to a new place, but all agents in the same place have the same attributes (possibly with different values).

The translation starts with declaring the agent classes as Netlogo agents and all the possible associated attributes. New agent classes are declared in NetLogo with the command "breed", while "<breed>-own" defines the attributes of all the agents belonging to the same class. For instance, on the model in Figure 1, the translation is:

```
;; declare main agent classes
breed [Tcells a_Tcells]
breed [Ecells a_Ecells]
...
```

1. Available at https://qbioturin.github.io/epimod/

```
;; Agent attributes
Tcells-own [place myrate totrate]
Ecells-own [place myrate totrate]
...
```

Every agent has a *place* attribute, which encodes the ESSN place the agent is currently residing in; although in the three models presented in this paper there are no additional attributes, in general these may exist, and are represented through additional colors associated with the tokens in each place: in this case one attribute would be declared in "<breed>-own" for each additional color-attribute. Instead *myrate* and *totrate* attributes are used to implement a NETA simulation algorithm ensuring that the ABM has the same semantics as the originating ESSN. After this first step, each place is associated with a unique identifier that can be assigned to the *place* attributes, and a set of initial agents (corresponding to $\mathbf{m}_0$) is generated:

```
;; place identifiers
set aliveT 1000
set aliveE 1001
...
;; setup initial marking
create-Tcells 1 [ set place aliveT ]
...
```

After these initial declarations, the ESSN model semantics is translated in NetLogo as well. Every ESSN transition t is associated with a *leading agent class* $L_t \subseteq \mathcal{A}$, chosen among the agents residing in the input places of t. In the main loop (the *to go* section), each transition t is translated into a code in which a leading agent asks all other agents connected to the transition to count all possible interactions (the count is required to correctly evaluate the transition rate in the specific marking). A typical translated transition has form[2]:

```
;; transition birthT
set A1 Tcells with [place = aliveT]
if any? A1 [
  ask A1 [
    let countInstances 1
    ;; summing up all rates
    set myrate replace-item 0 myrate
              (countInstances * (a))
  ]
]
```

where all Tcells agents $A1$ in place *aliveT* that satisfy the *birthT* action conditions participate updating an instance counter *countInstances* and a rate counter *myrate*. Actually *myrate* contains a list of rates, one for each transition which represents an internal state change or an interaction where the considered agent is leader: in the above example item 0 of list *myrate* refers to transition *birthT*, which involves only one alive agent of type *Tcells* (so that *countInstances* is 1). If selected in the subsequent step, this transition will generate a new agent of the same type. In general, when other alive agents are involved in the same transition, the number of possibly interacting partner agents is reflected in the value of *countInstances* and has an impact on the rate.

2. The current implementation is in beta state and does not translate general laws for rates' calculations yet; these must be added manually.

To have the same stochastic semantics in the NetLogo model and the originating ESSN model, the system enumerates all rates of the possible events and then chooses the next event randomly (weighted by the event rates). To perform this step, a second loop step takes place, where each agent sets its *totrate* local variable as the sum of all elements in its *myrate* list, then a leading agent is randomly selected:

```
let allAgents (turtle-set Tcells Ecells ...)
ask allAgents [set totrate sum myrate]
;; select the next agent doing an action
let chosenAgent rnd:weighted-one-of
                    allAgents [totrate]
```

according to the due proportion of its rate w.r.t. the total rates of all agents (the `weighted-one-of` function of the NetLogo `Rnd` extension here is applied to the set of all agents, and randomly selects one of them with a probability proportional to the value of the *totrate* local variable of each agent in the set). In the next step an inner loop chooses the participating transition t within the selected agent, and performs the proper updates according to the agent variables labeling the transition arcs: (a) An agent variable that appears both on an input and an output arc is updated (the place is changed, its attributes are modified according to the arc function); (b) An agent variable that only appears on an output arc is newly created. (c) An agent variable that appears in input but does not appear in output is destroyed.

Considering the *birthT* transition, it does not change the state of the leading agent ($t1$ was already in place *aliveT*) but generates a new agent ($t2$) of the same type, starting in the same state:

```
;; chosenAgent is leader of birthT
;; agent t1 is modified
ask turtle t1 [
      set place aliveT  ]
;; agent t2 is new
    hatch-Tcells 1 [
      set place aliveT
      set myrate 0
      set totrate 0  ]
```

Once an event is completed, the loop restarts asking all agents their rates and selecting the next event.

In addition to the model translation, the instructions needed to compute the measures of interest are generated: in the examples discussed in the next section these are just the number of alive agents of each type, but it can be enriched with agent specific measures or observations, thus facilitating studies that require a micro-perspective.

4. Results

In this section, we aim at comparing the SSA and ABM solutions methodologies applied to the models presented in Section 2.1. From the ESSN model the GreatMod framework can generate both an SSA and an ABM, the latter exploiting the translation presented in Section 3. The two simulation models are therefore stochastically equivalent, as

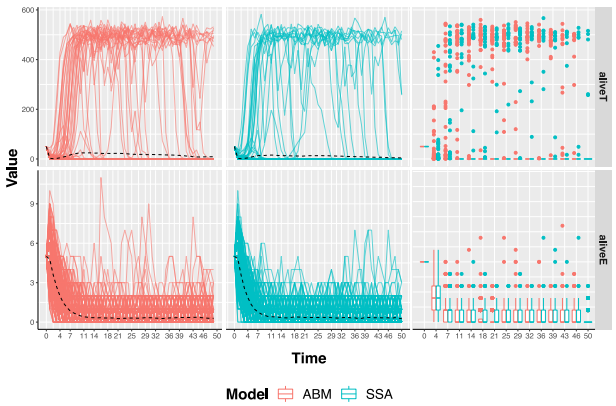

Figure 2. 500 runs considering the *TCells v.s. ECells* model, with initial condition of 50 TCells, and 5 ECells.

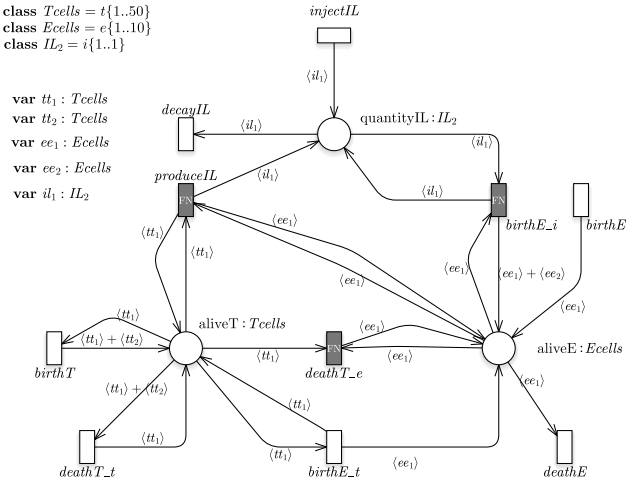

Figure 3. ESSN model modeling the *TCells v.s. ECells with IL2* system.

the results presented in this section show.

TCells v.s. ECells This simple model describes the interaction between the two populations of cells. Specifically, it models the control action role that ECells play on the spread of TCells, by killing them. In the ESSN of Figure 1, the transition *DeathT_e* models such control action. Furthermore, the general transition *birthE_t* express the increased rate of production of ECells due to the presence of TCells in the organism. Figure 2 shows the system dynamics arising from the interaction of TCells and ECells. Specifically, the first row shows the trend of the number of alive TCells in time for the ABM and SSA solutions, respectively in red and light blue. In both such plots, it is possible to pinpoint that most of the simulations fall into one equilibrium, where the number of TCells drops to zero. The remaining traces, just a few tens, reach a different equilibrium, namely around five-hundred alive TCells. The third plot—which compares the distribution of the solutions at each data point through a boxplot—clearly shows two aspects of this experiment; first of all, the equilibrium showing a number of TCells alive greater than zero can be considered as composed by all outlier solutions. Last but not least, it clearly shows that the empirical distributions obtained by the two solution approaches are almost the same. The second row shows the dynamic of the ECells in time. It can be noted that, with respect to the previous result, there is no bi-stability in the system, and the number of alive ECells drops rapidly towards zero. As for the TCells, from the boxplot it can be noted that the stochastic evolution of the system computed through the two solution methodologies does not show significant differences.

TCells v.s. ECells with IL2 The ESSN representation of the model is presented in Figure 3, and it is possible to identify ten transitions and three places. With respect to the ESSN model of the*TCell v.s. ECell*, the additional transitions and place allow to model the interaction with the cytokine IL2 molecule, refer to Table 2 for the list of all places and transitions. Specifically, the amount of cytokine IL2 molecule in the system is dependent on both the number

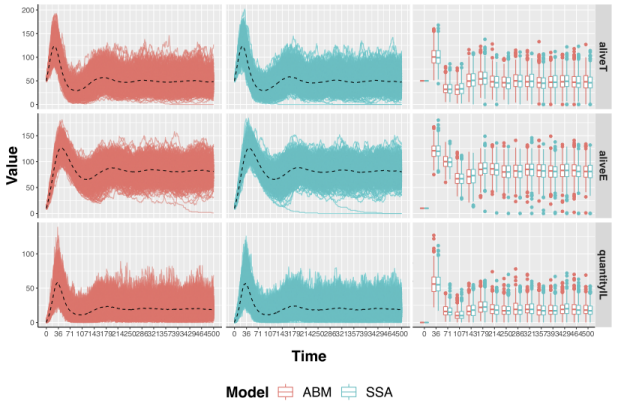

Figure 4. 500 runs considering the *TCells v.s. ECells with IL2* model, with initial condition of 50 TCells, 10 ECells, and 0 IL2.

of alive TCells and ECells: this fact is modeled through the arcs connecting the places *aliveTCells* and *aliveEcells* to the general transition *produceIL*. Furthermore, the quantity of cytokine IL2 in the system impacts the ECells' birth rate: indeed, the place *quantityIL* is one of the prerequisites to the birth of new ECells, being connected to transition *birthE_t* by an input arc. The rest of the transitions and places have the same meaning as the ones in the *TCells v.s. ECells* ESSN of Figure 1. The dynamics arising from this system are quite different from the one observed for *TCells v.s. ECells with IL2*. First of all, none of the variables taken into account show a bi-stable behavior, nor does their value drop to zero. Both the number of TCells and ECells, after an initial peak due to the initial condition, converge to an equilibrium. Similarly, the quantity of cytokine IL2—third row in Figure 4—after an initial spike, settles to a steady state. Considering the boxplots in the third column of Figure 4, it can be noted that the empirical distribution of the solutions of the two methodologies shows a very good

Event	Transition	Transition Law	Constant values
TCell birth	*birthT* (standard transition)	$a\ m_{aliveT}$	$a = 1.636$
TCell death	*deathT_t* (standard transition)	$a\ b\ m_{aliveT}^2$	$a = 1.636,\ b = 0.002$
TCell death by effector cells	*deathT_e* (standard transition)	$n\ m_{aliveT}\ m_{aliveE}$	$n = 1$
Effector death	*deathE* (standard transition)	$d\ m_{aliveE}$	$d = 0.3743$
Effector death by fighting TCells	*deathE_t* (standard transition)	$c\ m_{aliveE}\ m_{aliveT}$	$c = 0.00311$
Effector supply	*birthE* (standard transition)	s	$s = 0.1181$
Effector proliferation	*birthE_t* (general transition)	$\frac{p\ m_{aliveT}\ m_{aliveE}}{g + m_{aliveT}}$	$p = 1.131,\ g = 20.19$

TABLE 1. SUMMARY OF THE MATHEMATICAL DETAILS REGARDING THE EVENTS MODELED IN THE *TCells v.s. ECells*. THE NOTATION $m_{p_i},\ p_i \in \{aliveT,\ aliveE\}$ IS THE NUMBER OF TOKENS IN PLACE p_i.

Event	Transition	Transition Law	Constant values
TCell birth	*birthT* (standard transition)	$a\ m_{aliveT}$	$a = 0.08$
TCell death	*deathT_t* (standard transition)	$a\ b\ (m_{aliveT})^2$	$a = 0.08,\ b = 0.000000001$
TCell death by effector cells	*deathT_e* (general transition)	$\frac{\alpha m_{aliveT}\ m_{aliveE}}{g_2 + m_{aliveT}}$	$\alpha = 1,\ g_2 = 1000$
Effector death	*deathE* (standard transition)	$\mu_2\ m_{aliveE}$	$\mu_2 = 0.03$
Effector recruitment	*birthE_t* (standard transition)	$c\ m_{aliveE}\ m_{aliveT}$	$c = 0.00311$
Effector supply	*birthE* (standard transition)	s_1	$s_1 = 0$
Effector proliferation	*birthE_i* (general transition)	$\frac{p_1\ m_{quantityIL}\ m_{aliveE}}{g_1 + m_{quantityIL}}$	$p_1 = 0.1245,\ g_1 = 20000000$
IL2 supply	*injectIL* (standard transition)	s_2	$s_2 = 0$
IL2 decay	*decayIL* (standard transition)	$\mu_3\ m_{quantityIL}$	$\mu_3 = 1$
IL2 production	*produceIL* (general transition)	$\frac{p_2 m_{aliveT} m_{aliveE}}{g_3 + m_{aliveT}}$	$p_2 = 5,\ g_3 = 1000$

TABLE 2. SUMMARY OF THE TRANSITION RATE LAW OF THE *TCells v.s. ECells with IL2*. THE NOTATION $m_{p_i},\ p_i \in \{aliveT,\ aliveE,\ quantityIL\}$ IS THE NUMBER OF TOKENS IN PLACE p_i.

match.

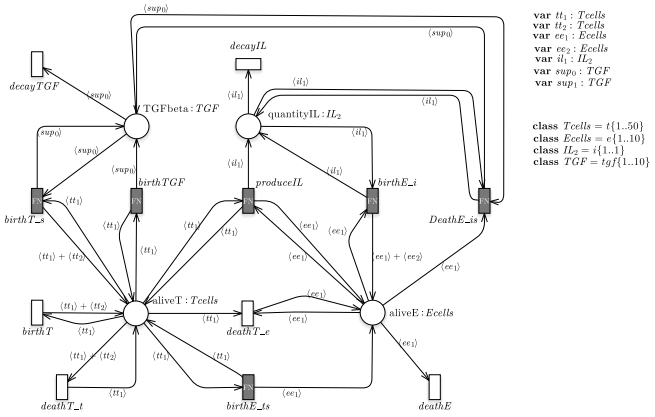

Figure 5. ESSN model modeling the *TCells v.s. ECells with IL2 and TGF-β* system.

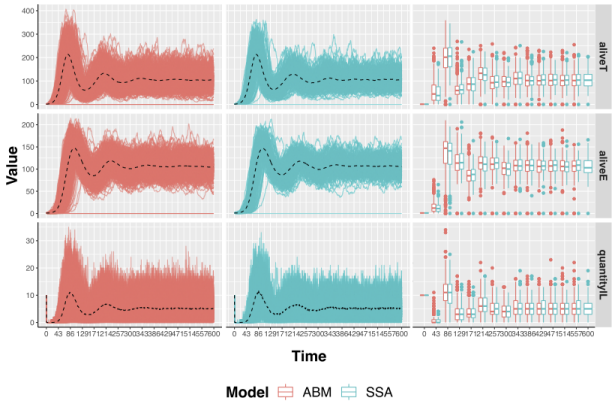

Figure 6. 500 runs considering the *TCells v.s. ECells with IL2 and TGF-β* model, with initial condition of 1 TCell, 1 ECell, 10 IL2, and 0 TGF-β.

TCells v.s. ECells with IL2 and TGF-β The introduction of the TGF-β in this model allows us to study its coupled effect of inhibiting the activation of ECells and stimulating tumor growth. Such effect is modeled in the ESSN of Figure 5 by introducing several elements with respect to the *TCells v.s. ECells with IL2* representation. Place *TGFbeta* represents the number of such cells in the system. This number has a positive impact on the number of TCells in the system. and indeed place *TGFbeta* is an input place for transition *birthT_s*, which injects new TCells into the system. On the other hand, the number of TGF-*beta* cells slows down the production of new ECells: the place *TGFbeta* is one of the input places of the transition removing ECells by the system, namely *deathE_is*. The introduction of the TGF-*beta* slows the initial growth in the cell population, which eventually reaches its steady state. Furthermore, the introduction of TGF-β causes an initial drop of the IL2 quantity in the system, then it increases and eventually reaches its steady state. Figure 6 shows that both the SSA and ABM model

Event	Transition	Transition Law	Constant values
TCell birth	*birthT* (standard transition)	$a\ m_{aliveT}$	$a = 0.1$
TCell death	*deathT_t* (standard transition)	$\frac{a}{K}(m_{aliveT})^2$	$a = 0.1,\ K = 5000$
TCell death by effector cells	*deathT_e* (general transition)	$\frac{\alpha m_{aliveT}\ m_{aliveE}}{g_2 + m_{aliveT}}$	$\alpha = 1,\ g_2 = 1000$
TCells growth due to TGF-β	*birthT_s* (general transition)	$\frac{p_2\ m_{aliveT}\ m_{TGFbeta}}{g_3 + m_{TGFbeta}}$	$p_2 = 0.27,\ g_3 = 20000$
ECells death	*deathE* (standard transition)	$\mu_1\ m_{aliveE}$	$\mu_1 = 0.03$
ECells suppression due to TGF-β	*deathE_is* (general transition)	$\frac{p_1\ q_1\ m_{TGFbeta}\ m_{quantityIL}\ m_{aliveE}}{(q_2 + m_{TGFbeta})(g_1 + m_{quantityIL})}$	$p_1 = 0.1245,\ g_1 = 2000,\ q_1 = 0.1121,\ q_2 = 200000$
ECells proliferation	*birthE_i* (standard transition)	$\frac{p_1\ m_{quantityIL}\ m_{aliveE}}{(g_1 + m_{quantityIL})}$	$p_1 = 0.1245,\ g_1 = 2000$
ECells recruitment	*birthE_ts* (general transition)	$\frac{c\ m_{aliveT}}{1 + \gamma\ m_{TGFbeta}}$	$c = 0.03,\ \gamma = 10$
IL2 decay	*decayIL* (standard transition)	$\mu_2\ m_{quantityIL}$	$\mu_2 = 10$
IL2 production	*produceIL* (general transition)	$\frac{p_3 m_{aliveT} m_{aliveE}}{(g_4 + m_{aliveT})(1 + \epsilon m_{TGFbeta})}$	$p_3 = 5,\ g_4 = 1000,\ \epsilon = 0.001$
TGF-β production	*birthTGF* (general transition)	$\frac{p_4\ (m_{aliveT})^2}{\theta^2 + (m_{aliveT})^2}$	$p_4 = 2.84,\ \theta = 100000$
TGF-β decay	*decayTGF* (standard transition)	$\mu_3 m_{TGFbeta}$	$\mu_3 = 10$

TABLE 3. SUMMARY OF THE TRANSITION RATE LAW OF THE THIRD SYSTEM.
THE NOTATION m_{p_i}, $p_i \in \{aliveT,\ aliveE,\ quantityIL,\ TGFbeta\}$ IS THE NUMBER OF TOKENS IN PLACE p_i.

computes the same solution, and the boxplot in the last column confirms that the empirical distributions agree.

5. Conclusions

In this paper we presented a multi formalism approach to the study of immune system mechanics, considering three variants of a case study proposed in [6]: each variant is first modeled by an ESSN model which is then automatically translated into both an SSA and an ABM model. The whole process is implemented within the GreatMod framework. The macro-level results obtained with the two simulators show a good match, confirming the coherence of the two models obtained through automatic translation: the ABM model can be naturally enriched to explore micro-level measures or agent-centered observations. The ESSN to ABM translator will be further developed to improve the efficiency of the ABM simulation model, exploiting the symmetries that can be automatically derived from the ESSN model. The integration of this approach into the GreatMod framework is a step towards the goal of offering a user-friendly toolbox implementing several analysis techniques that apply to different but coherent models derived from a unique graphical ESSN representation and offering the possibility of exploring the same problem from different points of view.

References

[1] M. Beccuti, P. Castagno, G. Franceschinis, M. Pennisi, and S. Pernice, "A Petri net formalism to study systems at different scales exploiting agent-based and stochastic simulations," in *Proc. of 17th European Workshop, EPEW 2021, and 26th International Conference, ASMTA, Virtual Event, December 2021, Revised Selected Papers*, ser. LNCS, vol. 13104. Springer, 2022, in press.

[2] U. Wilensky, "Netlogo," Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, http://ccl.northwestern.edu/netlogo/, 1999. [Online]. Available: http://ccl.northwestern.edu/netlogo/

[3] A. Borshchev, "XJ technologies: Anylogic 6," in *Proceedings of the 37th Winter Simulation Conference, Orlando, FL, USA, December 4-7, 2005*. IEEE Computer Society, 2005, p. 82. [Online]. Available: http://dl.acm.org/citation.cfm?id=1163276

[4] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. J. Bragen, and P. Sydelko, "Complex adaptive systems modeling with Repast Simphony," *Complex Adapt. Syst. Model.*, vol. 1, p. 3, 2013. [Online]. Available: https://doi.org/10.1186/2194-3206-1-3

[5] M. Kiran, P. Richmond, M. Holcombe, L. S. Chin, D. Worth, and C. Greenough, "FLAME: simulating large populations of agents on parallel hardware architectures," in *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, Eds. IFAAMAS, 2010, pp. 1633–1636. [Online]. Available: https://dl.acm.org/citation.cfm?id=1838517

[6] G. P. Figueredo, P.-O. Siebers, M. R. Owen, J. Reps, and U. Aickelin, "Comparing stochastic differential equations and agent-based modelling and simulation for early-stage cancer," *PloS one*, vol. 9, no. 4, p. e95150, 2014.

[7] S. Pernice and et al., "A computational approach based on the colored Petri net formalism for studying multiple sclerosis." *BMC bioinformatics*, vol. 20, no. 6, pp. 1–17, 2019.

[8] E. Voit and et al., "150 years of the mass action law," *PLoS computational biology*, vol. 11, no. 1, p. e1004012, 2015.

[9] M. Ajmone Marsan and et al., *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons Ltd, 1995.

[10] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The journal of physical chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[11] P. Castagno, S. Pernice, G. Ghetti, M. Povero, L. Pradelli, D. Paolotti, G. Balbo, M. Sereno, and M. Beccuti, "A computational framework for modeling and studying pertussis epidemiology and vaccination," *BMC bioinformatics*, vol. 21, p. 344, 2020.