Efficient Simulation and Parametrization of Stochastic Petri Nets in SystemC: A Case study from Systems Biology

***Abstract*—Stochastic Petri nets (SPN) are a form of Petri net where the transitions fire after a probabilistic and randomly determined delay. They are adopted in a wide range of applications thanks to their capability of incorporating randomness in the models and taking into account possible fluctuations and environmental noise. In Systems Biology, they are becoming a reference formalism to model metabolic networks, in which the noise due to molecule interactions in the environment plays a crucial role. Some frameworks have been proposed to implement and dynamically simulate SPN. Nevertheless, they do not allow for automatic model parametrization, which is a crucial task to identify the network configurations that lead the model to satisfy temporal properties of the model. This paper presents a framework that synthesizes the SPN models into SystemC code. The framework allows the user to formally define the network properties to be observed and to automatically extrapolate, thorough Assertion-based Verification (ABV), the parameter configurations that lead the network to satisfy such properties. We applied the framework to implement and simulate a complex biological network, i.e., the purine metabolism, with the aim of reproducing the metabolomics data obtained in-vitro from naive lymphocytes and autoreactive T cells implicated in the induction of experimental autoimmune disorders.**

***Index Terms*—Stochastic Petri Net, Metabolic Networks, Electronic Design Automation, T cells, Autoimmunity.**

I. Introduction

In Systems Biology, model development and analysis is recognized as a key requirement for integrating in-vitro and in-vivo experimental data. In-silico simulation of a biological model allows one to test different experimental conditions, helping in the discovery of the dynamics that regulate the system. Although qualitative characterizations of such complex mechanisms are, at least partially, available, a fully-parametrized quantitative description is often missing. Indeed, the large number of independent variables, the lack of quantitative information and the relationships strongly depending on qualitative events, make mathematical models difficult or even prohibitive to obtain and analyse [1].

In this context, Petri nets (PN) are an effective formalism to model biological networks as they provide an intuitive graphical representation, well-founded mathematical properties for qualitative analysis, and extensions to perform dynamic simulation [2]. Some approaches and software applications based on *continuous timed* Petri Nets have been applied to model and simulate biological systems through a semi-quantitative paradigm [3], [4]. Thanks to their mathematical properties, PN have been applied to model *place invariants* and *transition invariants* to validate complex processes such as apoptosis [5] and to understand the processes at the basis of the iron homeostasis [6].

This work has been partially supported by *** ***.

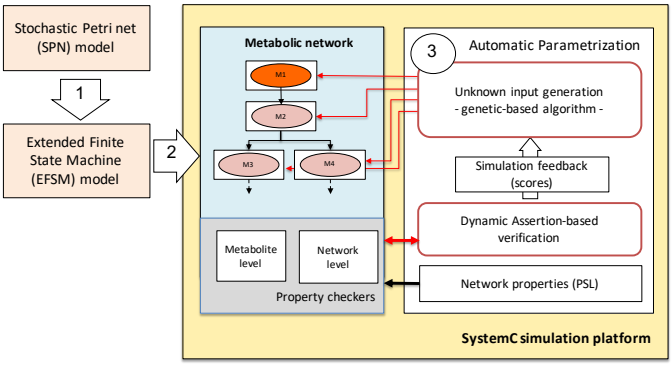

Fig. 1. The framework overview

Stochastic Petri nets (SPN) are a form of PN that allows the model to support randomness, which is fundamental to take into account possible fluctuations and noise due to molecules interacting in the environment [7]. For example, SPN have been succefully applied to obtain new insights in the development of hepatic granuloma throughout the course of infection [8], and to model signal transduction pathways in the process of angiogenesis [9].

Software applications have been developed to simulate SPN models of biological systems. The best known are Snoopy [10] and Monalisa [11]. They allow simulating complex biochemical systems including metabolic pathways, signal transduction pathways, and gene expression networks. Nevertheless, a recurrent issue of these software applications is related to the concept of *parameterization*. Very often, due to the lack of quantitative information, it is necessary to explore the *solution space* of the parameters to identify which network configurations lead the model to satisfy certain biological properties. This process requires the manual tuning of unknown parameters to obtain model behaviours matching the biological knowledge. Since the solution space to explore grows exponentially with the network size, such a manual parametrization task becomes prohibitive when applied to realistically large networks.

In this paper we propose a framework that applies languages, techniques, and tools well established in the field of

electronic design automation (EDA) to model and simulate metabolic networks. Since biological systems and electronics systems share several characteristics like concurrency, reactivity, abstraction levels as well as issues like reverse engineering and design space exploration [12], we show how the proposed EDA-based framework can introduce automation and flexibility to model, to simulate, and to help the analysis of metabolic networks. Figure 1 shows an overview of the framework. Starting from the SPN model of a metabolic network, the framework generates SystemC code that implements the network dynamics. The user defines network properties through a formal specification language (i.e., PSL), which are synthesized and integrated into the network code. The framework then applies Assertion-based Verification (ABV) combined to an automatic parameter generation based on a genetic algorithm to extrapolate the parameter configurations that satisfy the defined network properties.

We applied the framework to model and simulate the purine metabolism and to reproduce the metabolomics data obtained from naive lymphocytes and autoreactive T cells implicated in the induction of experimental autoimmune disorders. We show that the framework automatically extrapolates system parametrizations that reproduce the experimental results and that allow simulating the model under different conditions.

The paper is organized as follows. Section II presents the background on SPN. Section III presents the framework, while Section IV presents the experimental results. Section V is devoted to the concluding remarks.

II. Background on Stochastic Petri Nets

SPN is a class of Petri nets in which at every transition k of the network state is associated a delay τ_k, which is determined by a random variable. Formally, a SPN is a five-tuple $SPN = \{P, T, F, M_0, \Lambda\}$ where P is the set of the places, T the set of transitions, $F \subset (P \times T) \cup (T \times P)$ is the set of relations between places and transitions, M_0 is the initial configuration of the net (marking), and Λ is the set of exponentially distributed firing rates λ_k associated with the transitions. The firing rates are defined trough *propensity functions* (hazard) a_k that have the pre-places of the transition k as domain, and it gives the probability that a reaction will occur in the next infinitesimal time interval.

To simulate biochemical systems, specific types of propensity functions are used, such as *mass-action propensity functions* and *Michelis-Menten propensity functions*. A well-established method to perform a simulation of a SPN is the Gillespie's algorithm, called Stochastic Simulation Algorithm (SSA). This method is a Monte Carlo procedure that calculates a possible trajectory of the system simulating one chemical reaction at each step, and that chooses the firing time τ. Gillespie proposed two equivalent variants of his algorithm: The Direct Method (DM) [13], and the First Reaction Method (FRM) [14]. Several algorithms have been proposed to improve the efficiency of these algorithms [15]–[17]. In general, DM is more efficient in terms of computational time and space while FRM is well suited for a concurrent and parallel implementation [18]. Our methodology is based on the FRM variant.

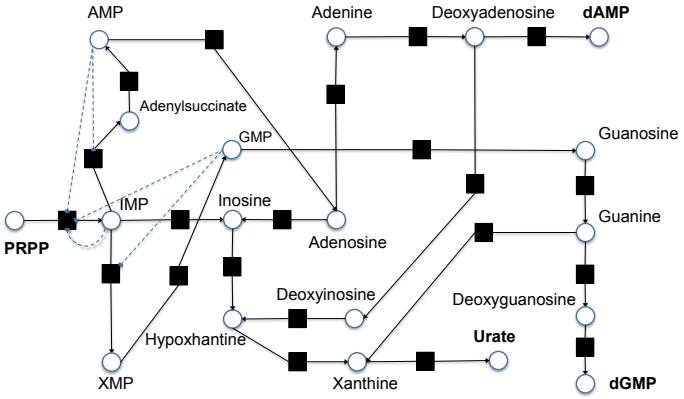

Fig. 2. Example of PN model: the purine pathway case study.

III. Methodology

Fig. 1 shows an overview of the simulation and parametrization framework. It relies on three main phases:

1) The modeling of SPN through extended finite state machines (EFSM) as explained in Section III-A.
2) The synthesis of EFSM into a SystemC with the support for stochastic simulation, as explained in Section III-B.
3) The automatic parametrization of the network through a genetic-based input generation guided by dynamic assertion-based verification, as explained in Section III-C.

A. Modeling PN through EFSM

The proposed methodology considers a PN model of the system under analysis as a starting point. Figure 2 shows the PN model of the purine pathway, which is the case study from Systems Biology considered as running example in this paper. Figure 3 shows an overview of the different possible reactions in metabolic networks, which represent the basic blocks of the PN models. The figure reports the PN models and the corresponding representations in Systems Biology Graphical Notation (SBGN-PD), which is the standard representation in the Systems Biology community. For the sake of clarity, we refer to the simplest reaction (top of Figure 3) in the follows.

To perform the event-driven simulation of the SPN, we represent each SPN node model through Extended Finite State Machines (EFSMs) [19]. This allows us to modularly synthesized the whole SPN model into executable SystemC code. Figure 4 shows such a representation. Each reaction involves the consumption precess c_i of the reactant M_i, and the production process p_j of the reaction result M_j. We thus represent the production and consumption processes through two concurrent EFSMs, each one based on a two-state machine.

Starting the consumption process from the `Ready` state of the preplace M_i, which represents the possibility to start the reaction, the model moves to the `In consumption` state by

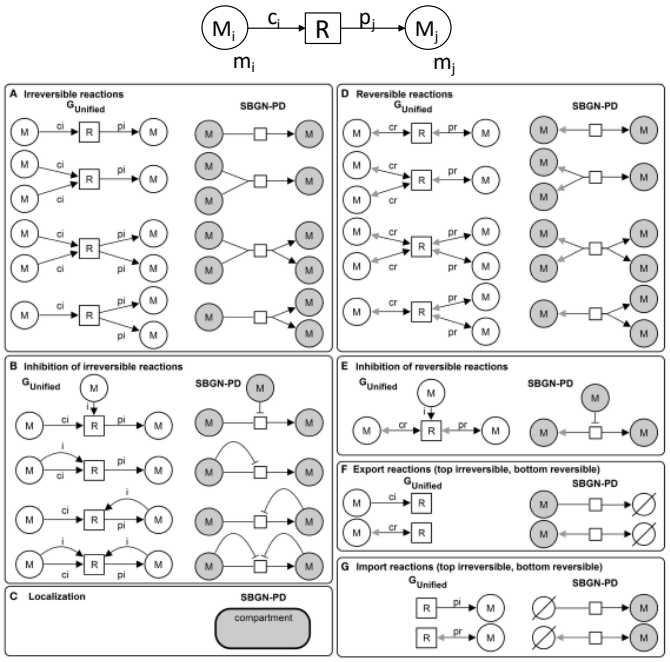

Fig. 3. Basic blocks (reactions) of a metabolic network in PN and the corresponding representation in SBGN-PD.

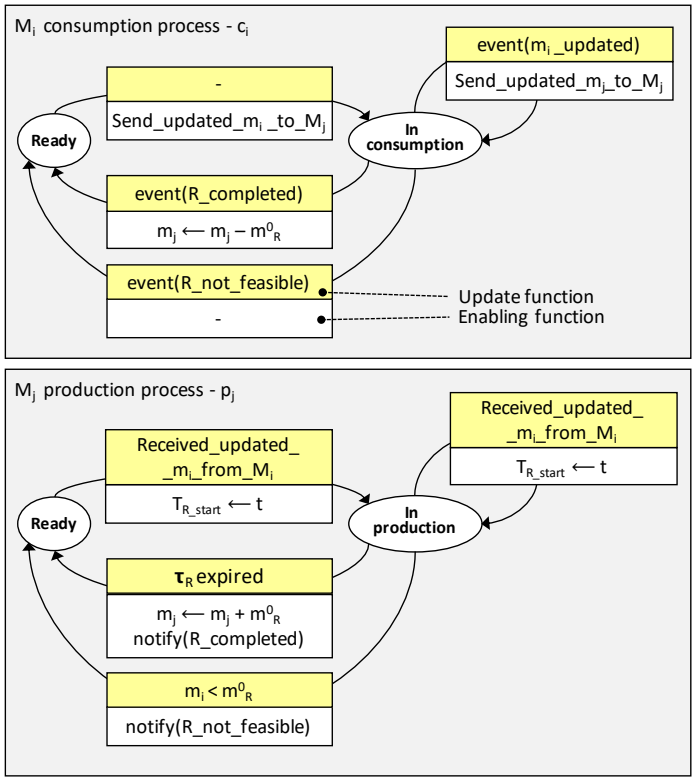

Fig. 4. The EFSM representation of the reaction processes R between M_i and M_j.

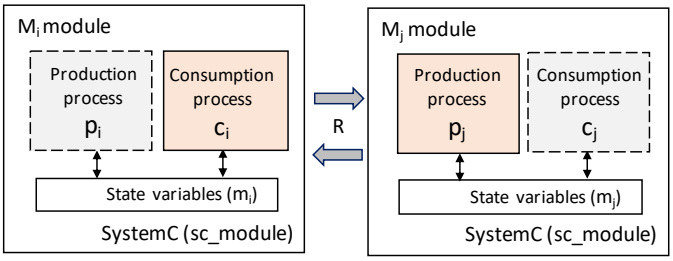

Fig. 5. Implementation of the concurrent consumption and production processes of a reaction R in SystemC modules.

updating the place M_j with the current reactant concentration. This is necessary to start the production activity of the concurrent process p_j. The machine waits in the `In consumption` state until the reaction delay time expires and decreases the reactant concentration m_j of a given reaction constant m_R^0. If the reactant concentration modifies during such a waiting time ($event(m_i\_updated)$ triggers), the new concentration is sent to p_i in order to re-evaluate the reaction delay time. This allows us to handle the multiple and concurrent evolutions of the SPN nodes and the corresponding effect on the rest of the network. As an example, if the reactant concentration decreases under the a minimum threshold during simulation due, for instance, to a parallel reaction process of the reactant that completes sooner (e.g., see the fourth block from the irreversible reactions of Fig. 3), the current reaction suspends (i.e., the machine goes back to the `Ready` state) without decreasing the M_i concentration.

The production process p_j moves from the `Ready` state to the `In production` state as soon as it receives an updated reactant concentration, and it saves the current simulation time t (which is updated by the SystemC simulation kernel at each simulation step) as the starting reaction time (T_{R_start}). When the reaction delay time (τ_R) expires, the reaction is completed and the M_j concentration is increased by the reaction constant m_R^0. In the `In production` state, the model allows the delay counting to be updated, either for a change of the reactant concentration and for the stochastic nature of the implemented simulation model (see Section III-B). The production process may become infeasible if the reactant concentration decreases under a threshold. This leads p_j to stop the process, to not increase the reaction result (i.e., M_j concentration) and to stop any M_i consumption process.

B. Stochastic Discrete Event-based Simulation of Petri Nets through SystemC

In a stochastic simulation of a metabolic network, each reaction R between a metabolite (M_j) and the reactant (M_i) fires:

1) if the reactant (M_i) satisfies a concentration constraint ($m_i > m_i^0$), where m_i is the reactant concentration (i.e., number of tokens of the PN node) and m_i^0 is a reaction coefficient (i.e., a constant), and
2) after a specific time delay τ_R, which is defined as follows:

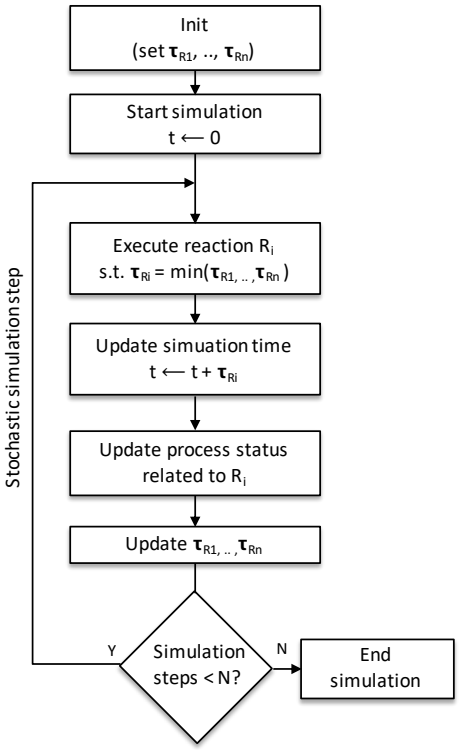

Fig. 6. Overview of the stochastic discrete simulation.

$$\tau_R = \left\lfloor \left(\frac{1}{a_R(m_i)} \right) \cdot ln \left(\frac{1}{rand_k} \right) \right\rfloor + 1$$

where $rand_k \in (rand_1, \ldots, rand_n)$ is a random number from the uniform distribution $U(0,1)$. $a_R(m_i) = c_i m_i$ is the mass-action propensity function where c_i is the reaction rate and m_i is the number of tokens of the reactant M_i. We consider a lower-bound value of each reaction delay (i.e., 1), which is associated to the minimum delay in the discrete simulation.

According to the FRM model (see Section II), the stochastic simulation of the system evolves along discrete steps, where each next time step corresponds to the expiration time of the immediately next reaction. In particular, considering t_i as the current simulation time of the system:

$$t_{i+1} = t_i + \tau_{R_i}, \text{ s.t. } \tau_{R_i} = min(\tau_{R_1}, .., \tau_{R_k})$$

At each simulation step, all the reaction delays are updated, and the updating considers the elapsed time and a random component. Figure 6 shows a summarizing overview of such a stochastic simulation paradigm. It is important to note that the updating phase at each simulation step of all the reaction delays $\tau_{R_i} \forall i$, which include a random component ($rand_k$), is the necessary condition to perform the stochastic simulation of the PN model. The simulation ends after a given number (N) of simulation steps.

Figure 5 shows how the two concurrent processes implementing each reaction are organized in the SystemC implementation. Each node of the SPN (i.e., place) consists of two processes, i.e., the production process from the upstream reaction and the consumption process for the downstream reaction.

C. Parameter estimation through dynamic Assertion-based verification (ABV)

Functional verification based on assertions represents one of the main applied and investigated techniques that combines simulation-based (i.e., dynamic) and formal (i.e., static) verification [21]–[23]. Assertions are formal descriptions that allow system designers to detect functional errors in the model and in the model evolution over time. They are also combined to techniques of automatic input pattern generation [24] that extrapolate *system configurations* to prove the satisfiability or unsatisfiability of the system properties. The proposed methodology applies simulation-based ABV, by which assertions are defined in a formal language (i.e., PSL [25]), they are automatically synthesized into *checkers*[1], and plugged to the SystemC model representing the network [27]. In our context, checkers aim at monitoring the concentration of the metabolites and to give a *score* (i.e., fitness) to an input generation module, which implements a genetic algorithm to generate good configurations of the kinetic parameters.

The module generates a configuration of parameters and runs a dynamic simulation of the network for such a set of input values for a given simulation time. Then, the module generates a new different configuration for a new simulation. A proper fitness function evaluates the goodness of each potential solution estimated through simulation. The run ends when the module finds the parameter configuration that allows the system properties to be satisfied. The definition of the fitness function depends on the property to be checked and it is formulated as the distance between *state vectors* representing the *simulation trend* $s_i(t)$ in comparison to a defined *reference trend* $r_i(t)$ (i.e., the target behaviour of the system).

The ABV checks the simulation trend $s_i(t)$ and, eventually, it stops the simulation to provide a score compared to $r_i(t)$. Given the state vectors $S = [s_1, s_2, \ldots, s_n]$ and $R = [r_1, r_2, \ldots, r_n]$ representing, respectively, the simulation and the reference trend the score is defined as follows:

$$score_c(t) = d(S, R)$$

where d is a distance function (e.g., Euclidean distance) bewteen the state vectors. The formulation of state vectors is general and allows us to model biological behaviours such as the stability in concentration and the change of concentrations between time points.

Figure 7 shows some examples of the state dynamics of a metabolite to be observed and for which assertions can be defined. In particular, the first assertion checks whether the concentration stability of a given metabolite over time, by considering a user-defined tolerance ($\pm\sigma$). The second,

[1]The framework relies on the IBM FoCs synthesizer [26] for the automatic synthesis of assertions.

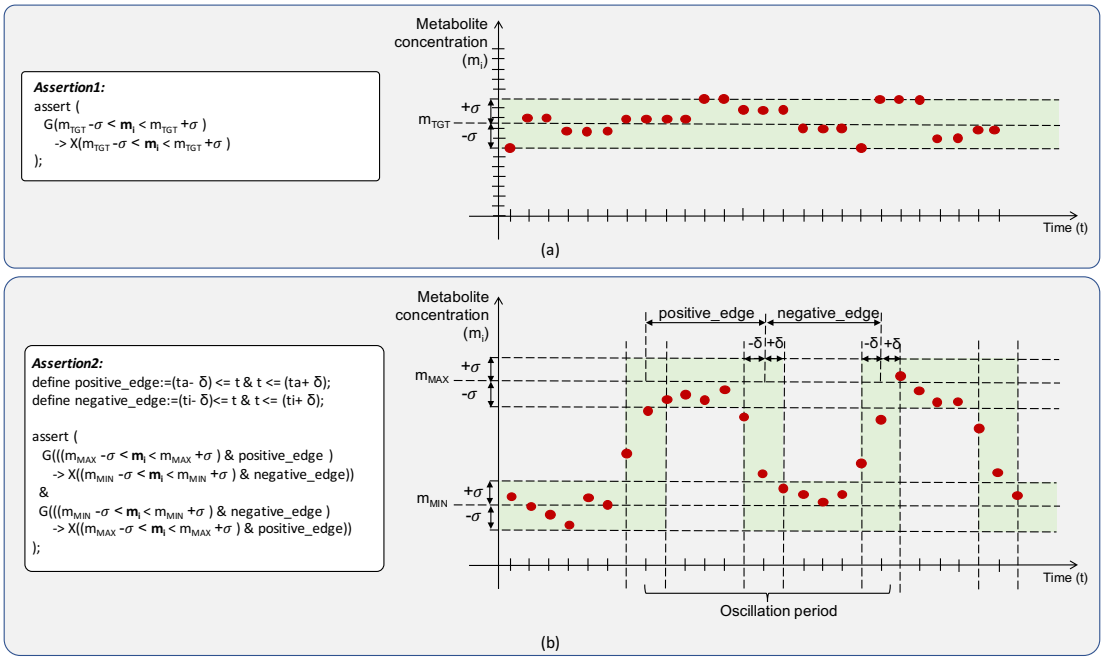

Fig. 7. Examples of assertion templates to verify a metabolite steady state - metabolite concentration stability (a), and the oscillation of a metabolite concentration (b). σ and δ are user-defined tolerance constants. m_{TGT}, m_{MAX}, and m_{MIN} are user-defined concentration values of the observed metabolite. ta and ti are temporal counters initialized at the first oscillation, and that hold the time elapsed from the first state transition ($m_{MIN} \rightarrow m_{MAX}$ and $m_{MAX} \rightarrow m_{MIN}$, respectively). They are used to measure the positive edge and negative edge values. t is the counter, which is set, from the second oscillation on, at each state transition, and it is used to measure the oscillation period.

more complex, assertion checks the periodic oscillations of the metabolite concentration by considering user-defined tolerances ($\pm\sigma$, $\pm\delta$). For both the examples, if the value of the state variable m_i, during simulation, remains in the green zone, the assertion is satisfied. Otherwise, the assertion fails and the system raises an error signal. We defined assertion templates, which have to be filled out by the user by indicating the metabolite to be monitored and the constants (target concentrations and tolerances).

The proposed framework applies ABV for *parameter estimation* phase, which aims at identifying the parameter settings that lead the network to satisfy the properties formalized by the assertions. This verification is delegated to the ABV, which is responsible for property checking during simulation. For each simulation of the model, the input stimuli generator receives a score (See Figure 1) from ABV and it applies this value to perform an evolutionary step of the genetic algorithm used to generate new configurations.

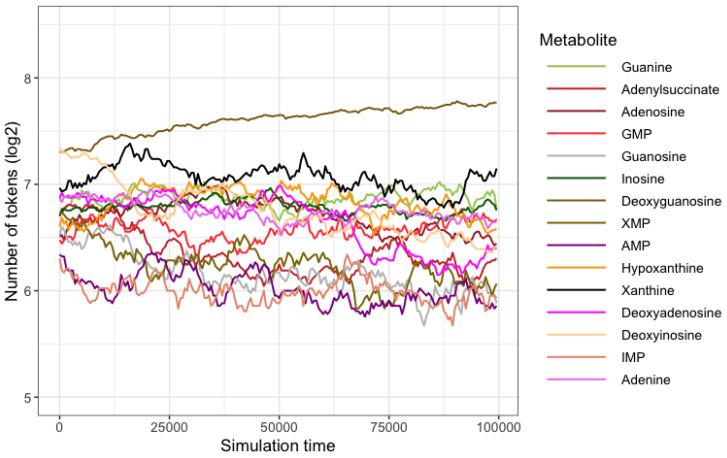

Fig. 8. Example of parameterization of the purine pathway in PLP-specific condition that lead the metabolites to stability within a simulation time of 10^5 clock cycles.

IV. Results and Discussion

We applied the proposed framework to understand how the dynamics of the purine pathway (see Figure 2) changes between normal and autoreactive conditions.

We started from metabolomics data obtained in-vitro and we converted the relative concentrations into number of tokens for each metabolite. With the term *concentration* we refer to the number of tokens of a metabolite. We simulated our model by generating reaction delays in the range $[1 - 1000]$ of all network reactions, which keep stable the concentrations of each metabolite (i.e., steady state) except dAMP, dGMP and urate. We assumed that the pathway is considered at steady state if the concentration of each element does not differ by more than $\pm 50\%$ from the initial concentration and it is maintained stable throughout a simulation time of 10^5 simulation cycles. We formally specified this property through PSL assertions.

In our model, the inhibition mechanisms are represented through the inhibition arcs, which is an extension of the classical Petri nets to represent the inhibition of a molecule when

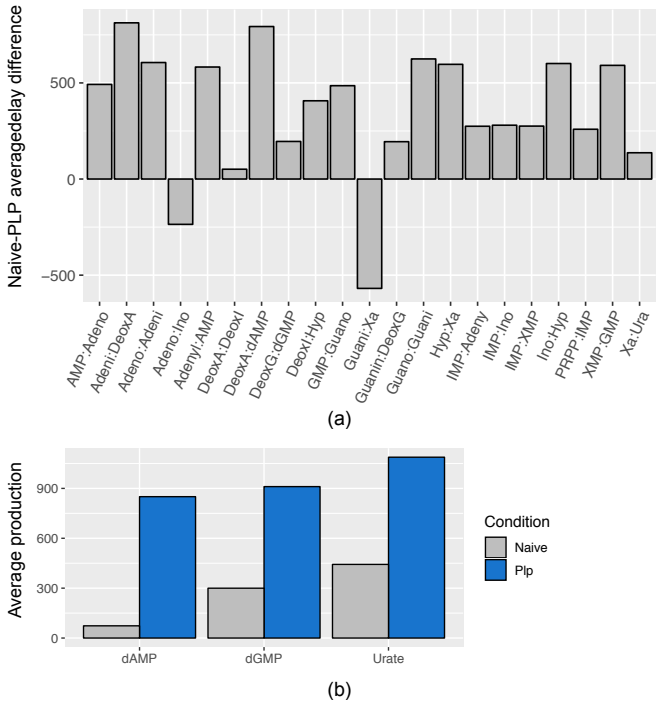

Fig. 9. Results of the analysis of 10 parameterizations of the purine pathway in condition of stability for naive and PLP-specific cells. (a) Average difference of the delays obtained parameterizing the pathway in naive and PLP-specific condition. (b) Difference in the final concentration of the metabolites dAMP, dGMP and urate.

its concentration exceeds a certain threshold. We assumed that a metabolite can inhibit a reaction when it grows up by 30% from its initial concentration. The genetic algorithm used by the automatic input generator has been configured with a population size of 250 individuals, a mutation probability of 0.05 and a crossover probability of 0.1. We defined the reference trend of the system as follows:

$$r_i(t) = m_i, \forall t > 0, i = 1, 2, ..., n$$

where m_i is the starting concentration of a metabolite and t is the simulation time. The state vector of the simulation trend is defined as $S = [c_1, c_2, \ldots, c_n]$, where c_i are the set of angular coefficients of the linear functions $s_i(t)$, linking the starting and the ending concentrations of the metabolites. The state vector of the reference trend was defined as $R = [0, 0, \ldots, 0]$. The fitness function was defined as the inverse of the Euclidean distance between the simulation and the reference trends. The selection method used to pick an individual from the population is *rank-based*, meaning that the reproduction is always done by taking individuals with better fitness.

We obtained 10 parameter configurations of the purine pathway for each condition. Fig. 8 shows the plot of the metabolite concentration obtained with one of such configurations. For each parameter configuration, the network simulation required around 7 seconds to simulate 1 million time instants (Simulation time in Fig. 7. The complete parametrization phase required from 1 to 12 minutes for each network version. All the simulations were run on a machine equipped with an Intel(R) Xeon(R) CPU E5-2650 v4 clocked at 2200 Mhz and 16 GBs RAM, and the Ubuntu 16.04 operating system.

In general, our simulations led to interesting differences in the regulation of the purine pathway, suggesting that most of chemical reactions are highly favored in PLP-specific cells versus naive lymphocytes as shown in Fig. 9(a). In fact, all metabolic reactions, with the exception of the reactions from Guanine to Xanthine (Guani:Xa) and from Adenosine to Inosine (Adeno:Ino), are speeded up in PLP-specific condition, having a lower average delay time generated by our framework (see Fig. 9(a)). Further, the reaction from Deoxyadenosine to Deoxyinosine (DeoxA:DeoxI) had comparable delay times between naive and PLP-specific condition. Overall, the observed speed-up in the PLP-specific condition resulted in a greater production of the fundamental elements of the pathway dAMP, dGMP and urate (Fig. 9(b)). Notably, the increased urate, dGMP and dAMP production in PLP-specific network reflects our metabolomics data and a well-known metabolic feature of proliferating lymphocytes [28], validating the potentiality of our methodology in simulating metabolic processes.

V. Conclusion

This paper presented a framework based on languages, techniques, and tools well established in the field of EDA to simulate and automatically parametrize the SPN model of metabolic networks. In particular, we applied the framework to study the purine metabolism pathway starting from metabolomics data obtained from naive lymphocytes and autoreactive T cells implicated in the induction of experimental autoimmune disorders. Thanks to the automatic parametrization of the model, we were able to reproduce the experimental results obtained in-vitro and to simulate the system under different conditions. This was not possible by using the tools at the state of the art, which require the manual insertion of the reaction parameters to simulate the network evolution. From a biological point of view, the obtained simulation results suggest that the entire purine pathway is speeded-up in PLP-specific cells versus naive lymphocytes, according to our experimental data and literature.

References

[1] R. Steuer and B. H. Junker, "Computational models of metabolism: stability and regulation in metabolic networks," *Advances in chemical physics*, vol. 142, p. 105, 2009.

[2] I. Koch, "Petri nets in systems biology," *Software and Systems Modeling*, vol. 14, no. 2, pp. 703–710, 2015.

[3] G. Russo, M. Pennisi, R. Boscarino, and F. Pappalardo, "Continuous petri nets and microrna analysis in melanoma," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 5, pp. 1492–1499, 2018.

[4] B. Behinaein, K. Rudie, and W. Sangrar, "Petri net siphon analysis and graph theoretic measures for identifying combination therapies in cancer," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 1, pp. 231–243, 2018.

[5] M. Heiner and I. Koch, "Petri net based model validation in systems biology," in *ICATPN*, vol. 3099. Springer, 2004, pp. 216–237.

[6] A. Sackmann, D. Formanowicz, P. Formanowicz, I. Koch, and J. Blazewicz, "An analysis of the Petri net based model of the human body iron homeostasis process," *Computational Biology and Chemistry*, vol. 31, no. 1, pp. 1–10, 2007.

[7] C. Chaouiya, "Petri net modelling of biological networks," *Briefings in bioinformatics*, vol. 8, no. 4, pp. 210–219, 2007.

[8] L. Albergante, J. Timmis, L. Beattie, and P. M. Kaye, "A petri net model of granulomatous inflammation: implications for il-10 mediated control of leishmania donovani infection," *PLoS computational biology*, vol. 9, no. 11, p. e1003334, 2013.

[9] L. Napione et al., "On the use of stochastic petri nets in the analysis of signal transduction pathways for angiogenesis process," in *International Conference on Computational Methods in Systems Biology*. Springer, 2009, pp. 281–295.

[10] M. Heiner, M. Herajy, F. Liu, C. Rohr, and M. Schwarick, "Snoopy–a unifying petri net tool," in *Int. Conf. on Application and Theory of Petri Nets and Concurrency*. Springer, 2012, pp. 398–407.

[11] P. Balazki, K. Lindauer, J. Einloft, J. Ackermann, and I. Koch, "Monalisa for stochastic simulations of petri net models of biochemical systems," *BMC bioinformatics*, vol. 16, no. 1, p. 215, 2015.

[12] J. Fisher and T. A. Henzinger, "Executable cell biology," *Nature Biotechnology*, vol. 25, pp. 1239 – 1249, 2007.

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

[14] ——, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of computational physics*, vol. 22, no. 4, pp. 403–434, 1976.

[15] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *The journal of chemical physics*, vol. 121, no. 9, pp. 4059–4067, 2004.

[16] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova, "The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior," *Computational biology and chemistry*, vol. 30, no. 1, pp. 39–49, 2006.

[17] V. H. Thanh, C. Priami, and R. Zunino, "Efficient rejection-based simulation of biochemical reactions with stochastic noise and delays," *The Journal of chemical physics*, vol. 141, no. 13, p. 10B602_1, 2014.

[18] C. Dittamo and D. Cangelosi, "Optimized parallel implementation of gillespie's first reaction method on graphics processing units," in *Computer Modeling and Simulation, 2009. ICCMS'09. International Conference on*. IEEE, 2009, pp. 156–161.

[19] K.-T. Cheng and A. Krishnakumar, "Automatic generation of functional vectors using the extended finite state machine model," *ACM TODAES*, vol. 1, no. 1, pp. 57–79, 1996.

[20] N. Bombieri, R. Distefano, G. Scardoni, F. Fummi, C. Laudanna, and R. Giugno, "Dynamic modeling and simulation of leukocyte integrin activation through an electronic design automation framework," in *International Conference on Computational Methods in Systems Biology*. Springer, 2014, pp. 143–154.

[21] C. N. CoelhoJr. and H. D. Foster, *Assertion-based Verification*. A: Springer, 2008, vol. 4.

[22] N. Bombieri et al., "On the evaluation of transactor-based verification for reusing TLM assertions and testbenches at RTL," in *Proc. of ACM/IEEE DATE*, vol. 1, 2006, pp. 1–6.

[23] Synopsys Inc., "Assertion-based verification," 2003, white paper, www.synopsys.com.

[24] M. Boulé and Z. Zilic, *Generating hardware assertion checkers: for hardware verification, emulation, post-fabrication debugging and on-line monitoring*. A: Springer, 2008.

[25] IEEE, "Property specification language - psl," 2017, https://standards.ieee.org/findstds/standard/1850-2010.html.

[26] Y. Abarbanel et al., "Focs–automatic generation of simulation checkers from formal specifications," in *Computer Aided Verification*. Springer, 2000, pp. 538–542.

[27] N. Bombieri, F. Fummi, V. Guarnieri, G. Pravadelli, F. Stefanni, T. Ghasempouri, M. Lora, G. Auditore, and M. Marcigaglia, "Reusing rtl assertion checkers for verification of systemc tlm models," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 31, no. 2, pp. 167–180, 2015.

[28] T. Eleftheriadis, G. Pissas, A. Karioti, G. Antoniadi, S. Golfinopoulos, V. Liakopoulos, A. Mamara, M. Speletas, G. Koukoulis, and I. Stefanidis, "Uric acid induces caspase-1 activation, il-1β secretion and p2x7 receptor dependent proliferation in primary human lymphocytes," *Hippokratia*, vol. 17, no. 2, p. 141, 2013.