THiMED: Time in Hierarchical Model Extraction and Design*

Natasa Miskov-Zivanov[1,2], Peter Wei[1], and Chang Sheng Clement Loh[1]

[1] Electrical and Computer Engineering Department, Carnegie Mellon University
[2] Computer Science Department, Carnegie Mellon University,
Pittsburgh, USA
nmiskov@andrew.cmu.edu

Abstract. We describe our approach to modeling timing of cell signaling systems in which existing information about the system spans from detailed mechanistic knowledge to much coarser observations about cause and effect. The results for several models emphasize the fact that the selection of timing implementation can have both qualitative and quantitative effects on the model's transient behavior and its steady state.

Keywords: timing, cell signaling, stochastic model, delay.

1 Introduction

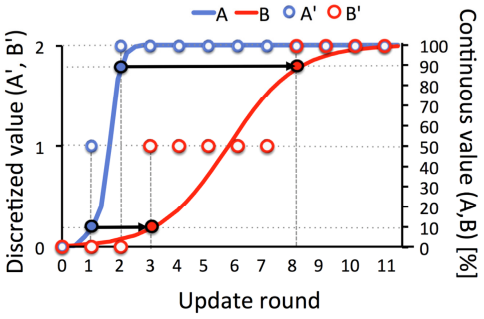
Time of occurrence and duration of events often play an important role in decision making in cell signaling networks [1]. Although timing of events can be modeled using reaction rates, exact element regulations are not always well understood, and even more, rates of reactions are not known. Still, to better understand how the overall system works, it is important to capture in the model much of the available knowledge about the system. When experimental observations provide insights into indirect cause-effect relationships only, and do not explain many of the detailed interaction mechanisms [2], our modeling approach accounts for (*i*) thresholds in element activity, thus discretizing model variables [3], (*ii*) relative delays between events and in element responses to regulation changes, thus capturing critical event timing.

2 Approach

We model system elements using multi-valued variables, and by using this approach we are able to capture multiple layers of cell signaling: interactions between receptors and external stimuli, intracellular signaling, gene regulation, cell's response to stimuli, and feedback to cell receptors [1][2]. Such an approach has been shown valuable in providing critical insights into system's transient behavior, when models are coarse-grained in parts or in whole due to available knowledge. To increase accuracy of the model, in our approach we allow for implementation of timing details that capture

* This work is supported in part by DARPA award W911NF-14-1-0422.

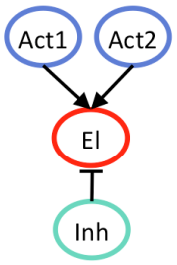
P. Mendes et al. (Eds.): CMSB 2014, LNBI 8859, pp. 260–263, 2014.



A	B	B next
0	0 or 1 or 2	0
1	0	1(d=2)
1	1 or 2	1
2	0	1
2	1	2(d=6)
2	2	2

```
if (A>0 and B=0) then
      wait #2 (A>0): B<=1
else if (A=2 and B=1) then
      wait #6 (A=2): B<=2
else if (A=0) then
      B<=0
```

(a)



El Inh \ Act1 Act2	0 0	0 1	1 0	1 1
0 0	0	1 (d=1)	1 (d=1)	1
0 1	0	1 (d=2)	1 (d=2)	1 (d=1)
0 2	0	0	0	0
1 0	1	2 (d=1)	2(d=1)	2
1 1	0 (d=2)	2 (d=2)	2 (d=2)	2 (d=1)
1 2	0	0 (d=1)	0 (d=1)	0 (d=2)
2 0	2	2	2	2
2 1	1 (d=1)	1 (d=2)	1 (d=2)	2
2 2	1	1 (d=1)	1 (d=1)	1 (d=2)

(b)

Fig. 1. Examples of delay representation. (a) Discretization and corresponding delay description. (b) Different delays defined for different regulator values.

relative delays between events. Once the delays are described formally (e.g., using delay truth tables [3]), our tool translates them into variable update rules. We identified three different methods to model delays that occur between a change in given element regulation (i.e., change in combination of regulator values and current element value), and a corresponding change in the element's value. We describe our approaches to delay modeling using the following two examples.

Example 1. Assuming that there are two elements, A and B, and that A positively regulates B, the time needed for B to respond to different changes in A may be different, depending on current values of A and B. Figure 1(a) (left) shows one scenario in which A increases from very low level (around 0% its maximum value) to high level (100% its maximum value). While B can relatively quickly follow the initial change in A, it takes longer for B to come close to 100% of its highest value. The two lines representing A and B can be discretized, assuming thresholds for values 1 and 2 (e.g., reaching 10% of highest activity or concentration can be a threshold for value 1 and reaching 90% of highest activity or concentration can be a threshold for value 2). Figure 1(a)(left) outlines discretization example for A and B, and the table in Figure 1(a)(middle) shows how current values of A and B can determine next value for B. Delays in changing B value are indicated by "d=2" and "d=6" which represent 2 time-unit and 6 time-unit delays, respectively. Our tools translate these tables into executable rules. In addition, these relationships can be described in code and translated into an executable model from the code. The description in Figure 1(a)(right) is very

suitable for implementing in Hardware Description Languages (HDLs) that can be translated in an automated way into executable circuit models. Similar work on emulation of biological networks in Field Programmable Gate Arrays has been described in [4,5].

Example 2. Given a small regulatory network with four components (Figure 1(a), left), element (El), its two positive regulators (Act1 and Act2), and negative regulator (Inh), we draw a table (Figure 1(a), right) listing all combinations of regulator values including previous value of element El, and show the resulting new value for El. Note that El and Inh have three different levels of activity, 0 (not active), 1 (low activity), and 2 (high activity), while both activators are modeled only with two levels, 0 (no activity) and 1 (active). Table entries of the form "d=1" or "d=2" indicate that the transition from one value to another occurs after 1 time-unit delay or after 2 time-unit delays, respectively. For example, when El has value 2, Inh has value 1 and both activators, Act1 and Act2, have value 0, El will change value from 2 to 1 with some, short delay. For the same values of El and Inh, when one of the activators has value 0 and the other one value 1, then El will change from 2 to 1 with a longer delay.

The delay assumptions can be implemented in executable model generation and in simulation in several different ways, as shown in the following.

2.1 Forward Propagation

In the first delay implementation, all regulator value combinations that satisfy the same transition requirement in terms of previous and next element value and delay interval (i.e., all delay truth table entries with same output value, for example, "1(d=1)") are lumped into a single function. Such implementation assumes that measuring delay (lapsed time) is not reset even when the actual conditions change, as long as the outcome is same (e.g., when El=2, Inh=1, Act1=0, Act2=1, and then Act1 changes value to 1 and Act2 changes value to 0, the effect on El remains the same, and thus counting of steps to satisfy 2 time-unit delay, "1(d=2)", remains the same). This approach allows for minimizing element update functions, since multiple table entries can be lumped into a single function. Besides minimizing the function, this also requires smaller number of variables to be propagated from one simulation round to the next (thus the name for the method).

2.2 Backward Propagation

In contrast to the first approach, if the conditions change before the required delay interval has lapsed, even when the new output is same for the new conditions, measuring of delay interval is reset. This delay modeling approach requires different "memory" implementation compared to the first approach. In other words, this approach requires that, depending on how many delay steps are defined, the simulator checks variable values in the corresponding number of previous rounds. In this case, functions that are to be computed are simpler compared to the previous approach (forward propagation), but the number of variables increases.

2.3 Buffer Insertion

The third approach implements delays as "buffers" that add steps to the pathway, thus delaying propagation of any value of a regulator (for any combination with other regulators) to some or all of its downstream elements. In other words, the table created for this case will not have delay entries (e.g., "`1(d=2)`") but instead only discrete numbers without indication of delays. This approach can be used when modeling pathway sections without crosstalk or in the case where only indirect causal relationships are known while the overall timing of the pathway still needs to match the timing of other pathways in the network. This delay modeling approach was applied previously in [1] and it resulted in a good match with experimental results for situations where there are multiple competing pathways without significant crosstalk.

2.4 Simulation

We have also worked on simulation approaches to accurately account for these different delay modeling methods. Depending on the simulator setup, delay values in cell signaling models can be assumed exactly as defined, or can represent upper bounds or mean delay values.

3 Results

We applied the described timing modeling approaches in development and analysis of two models, T cell differentiation model [1] and immune crosstalk in malaria infection in mosquitoes [2]. We have shown that, depending on the delay implementation method, different delay values can affect results both qualitatively and quantitatively, and can change both transient behavior and steady state of individual elements, as well as of the system as a whole.

References

1. Miskov-Zivanov, N., et al.: The duration of T cell stimulation is a critical determinant of cell fate and plasticity. In Science Signaling 6, ra97 (2013)
2. Vodovotz, Y., et al.: Modeling Host-Vector-Pathogen Immuno-inflammatory Interactions in Malaria. In: Complex Systems and Computational Biology Approaches to Acute Inflammation, pp. 265–279. Springer, New York (2013)
3. Miskov-Zivanov, N., et al.: Dynamic behavior of cell signaling networks: model design and analysis automation. In: Proc. of Design Automation Conference (DAC), Article 8, 6 p. (June 2013)
4. Miskov-Zivanov, N., et al.: Regulatory Network Analysis Acceleration with Reconfigurable Hardware. In: Proc. of International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 149–152 (September 2011)
5. Miskov-Zivanov, N., et al.: Emulation of Biological Networks in Reconfigurable Hardware. In: Proc. of ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM-BCB), pp. 536–540 (August 2011)