31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

A Method for Analysis of Simultaneous Equations in Cell Models

Takao Shimayoshi, Akira Amano and Tetsuya Matsuda

***Abstract*— Some models of cellular physiological functions are formulated as ordinary differential equations that contain multiple systems of simultaneous nonlinear equations. Simulation of such a model described in a declarative representation format requires determination of equations to be simultaneously solved with specification of independent and parameter variables in the model. In this report, a method for extracting systems of simultaneous equations in a cell model is presented. The present method analyzes a graph representation of a model and extracts the subgraphs that represent equation systems to be simultaneously solved, by efficiently interactive selection of independent variables of the model.**

I. Introduction

Modelling and simulation of cellular functions such as electrophysiology, ion dynamics, biomechanics, and signal transduction are important techniques for analysis of the mechanisms. The cell models are generally formulated as ordinary differential equations (ODE) to represent transitions of cell states such as membrane voltage, ion concentrations, and open probabilities of ion channels. Some of them additionally contain a few systems of simultaneous nonlinear equations to express chemical and mechanical equilibrium, conservation of mass, and so on. To simulate such a model, the simultaneous equations should be solved in each step of ODE calculation. In general, simultaneous solving of an algebraic equation system requires specification of a consistent set of independent variables and parameter variables, where an independent variable can independently change the value to find a solution of the simultaneous equations and a parameter variable is fixed at a certain value in solving the simultaneous equations.

To share and exchange cell models, a representation formats, CellML [1] is proposed. CellML can describe the mathematical definition of the model formally and declaratively using MathML. A set of mathematical equations does not define in itself which variables are independent or parameter variables. In addition, experimental protocols can change the set of independent and parameter variables. For an example, the cell tension is an independent variable and the cell length is a parameter in isometric contraction experiments, while vice versa in isotonic experiments. Therefore, simulation of such a cell model requires, in addition to the declarative model definition in CellML, a consistent set of indepenedent and parameter variables, in order to determine equations to be simultaneously solved.

In this report, a method for analysis of simultaneous equations in ODE is presented, in order to allow simulation of declaratively described cell models that contain simultaneous equations. In the present method, a model is represented as a graph and then systems of simultaneous equations are extracted from a cell model on the graph representation with efficiently specifying independent and parameter variables from the model. An existing graph theoretical method for structural analysis of simultaneous equations, which the present method utilizes, is fist introduced in section II, and then the present method is described in section III. An example of application of the present method to a cell model is finally reported in section IV.

A part of this work was supported by Biomedical Cluster Kansai.

T. Shimayoshi is with ASTEM Research Institute of Kyoto, Kyoto, 600–8813, Japan, and also with Research Organization of Science and Engineering, Ritsumeikan University, Shiga, 525–8577, Japan simayosi@astem.or.jp

A. Amano is with Department of Bioinformatics, Ritsumeikan University, Shiga, 525–8577, Japan

T. Matsuda is with Graduate School of Informatics, Kyoto University, Kyoto, 600–8501, Japan

II. Review of Structural Analysis Method

A. Structural Solvability of an Equation System

Iri et al. [2]–[4] proposed a graph theoretical method for analyzing the structural solvability of a system of simultaneous equations. The structural solvability of a system is defined that the system has a structure which admits a unique solution for arbitrary values of the parameters. The method analyzes an equation system in *standard form*:

$$\begin{cases} y_i = f_i(\boldsymbol{x}, \boldsymbol{u}) & (i = 1, \ldots, M) \\ u_k = g_k(\boldsymbol{x}, \boldsymbol{u}) & (k = 1, \ldots, K) \end{cases} \tag{1}$$

where $x_j\,(j{=}1,\ldots,N)$ are independent variables of the system, $u_k\,(k{=}1,\ldots,K)$ dependent variables, $y_i\,(i{=}1,\ldots,M)$ parameters, K, M and N are the sizes of vectors, and $f_i\,(i{=}1,\ldots,M)$ and $g_k\,(k{=}1,\ldots,K)$ are sufficiently smooth real-valued functions.

To represent the equation system as a graph, *representation graph* was defined. The representation graph of the equation system (1) is a directed graph $G{=}(V,E)$, where the vertex set $V = X \cup Y \cup U$ ($X = \{x_1,\ldots,x_N\}$, $U = \{u_1,\ldots,u_K\}$, $Y = \{y_1,\ldots,y_M\}$ correspond to variables) and the arc set E, which consists of arcs for each y_i from effective arguments of function f_i into the vertex y_i and for each u_k similarly. A vertex is called *maximal* if the in-degree is zero, and *minimal* if the out-degree is zero. All vertices in X are maximal and all vertices in Y are minimal. A *Menger-type linking* from X to Y is defined as a set of pairwise vertex-disjoint directed paths from a vertex in X to a vertex in Y. The size of a linking is defined as the number of paths from X to Y in the linking. A linking is perfect if the size is equal to both $|X|$ and $|Y|$.

Finally, the following theorem has been proved:

Theorem 1: An equation system in the standard form (1) is structurally solvable, if a Menger-type perfect linking from X to Y exists in the representation graph.

B. Algorithm for Menger-type Maximum Linking

Even and Tarjan reported in [5] that a Menger-type maximum linking can be found in $O(|V|^{\frac{1}{2}}|E|)$ time by Edmonds-Karp algorithm [6], [7] for the maximum flow problem. The method is summarized below.

In order to find a Menger-type maximum linking in a graph $G{=}(V, E)$ from a subset X of the vertices to a subset Y, construct a network $N{=}(\bar{V}, \bar{E}, c(e))$ as follows, where $c(e)$ gives the capacity of each arc e of the network:

1) Each vertex $v \in V$ is split into two vertices v^- and v^+, and then a new arc e connects from v^- to v^+ and $c(e) = 1$.
2) All arcs that formerly led to v now lead to v^-, and all arcs that emanated from v now emanate from v^+. For each arc e of them, $c(e) = \infty$.
3) Two vertex s and t are introduced for the source and sink of the network, and then new arcs connect from s to $v^-(v \in X)$ and from $v^+(v \in Y)$ to t. For each arc e of them, $c(e) = \infty$.

In summary, the network $N{=}(\bar{V}, \bar{E}, c(e))$ is defined:

$$\begin{aligned}
\bar{V} &= \{v^- : v \in V\} \cup \{v^+ : v \in V\} \cup \{s, t\} \\
\bar{E} &= \{(v^-, v^+) : v \in V\} \cup \{(u^+, w^-) : (u, w) \in E\} \\
&\quad \cup \{(s, v^-) : v \in X\} \cup \{(v^+, t) : v \in Y\} \\
c(e) &= \begin{cases} 1, & \text{if } e \in \{(v^-, v^+) : v \in V\} \\ \infty, & \text{otherwise.} \end{cases}
\end{aligned}$$

A maximum flow of the network N, which can be found by Edmonds-Karp algorithm, gives a Menger-type maximum linking in G from X to Y.

C. Structural Solvability in Numerical Simulation

Iri et al. [2], [4], [8] proposed an analysis method for structural solvability in numerical simulations also. The method employs the concepts of *system definition* and *operation definition*. A system definition in *standard form* is a set of equations, all of which are given as $z = f(u, v, \dots)$ and have distinct left-hand side variables. A system definition in standard form is expressed using the representation graph. An operation definition is the assignment of types to subsets of variables (and corresponding vertices of the representation graph). A variable is of *type S*, if the corresponding vertex is maximal and the value is constant. A variable is of *type G*, if the corresponding vertex is minimal and the variable is required equal to the specified value. If a variable that is fixed to the prescribed value is neither minimal nor maximal, the vertex is virtually split into two vertices, one of which is connected with all the outgoing arcs and labeled as type S, and the other of which is connected with all the incoming arcs and labeled as type G. Maximal vertices that are not of type S are automatically labeled as *type A*. As a result, the modified representation graph reveals an equation system in the standard form (1) by regarding variables of type A as x_j, variables of type G as y_i, variables of type S as constants, and the remaining variables as u_k. Consequently, the structural solvability can be verified by Theorem 1.

III. Analysis of Simultaneous Equations

In this section, an interactive method for configuring a cell model that contains simultaneous equations is presented. A preparation for a model is described first, and then the algorithm for extraction of simultaneous equations from a model is defined.

A. Preparation of Model Equations

A minor modification is required for a model to fit the structural analysis method in section II. Equations of a cell model may not be defined in standard form; not all the equations are given as $z = f(u, v, \dots)$, or multiple equations have the same left-hand side variables. Such equations can be turned into the equivalent standard form by following conversions.

1) If the one side of an equation is an explicit number b:
$$f(u, v, \dots) = b,$$
the equation is converted into standard form by introducing a new variable β of type G in place of b as
$$\beta = f(u, v, \dots),$$
where the value of β is required to be b.
2) If the both sides of an equation are not a single variable, i.e. given as
$$f(u, \dots) = g(v, \dots),$$
the standard form can be obtained by introducing a new variable ϕ of type G to be zero and transposing the equation as
$$\phi = f(u, \dots) - g(v, \dots).$$
3) If more than one equation exist for the same left-hand side variable w:
$$\left\{ \; w = f_i(\boldsymbol{v}_i) \quad (i = 1, \dots, N) \right. ,$$
these equations are reformulated by introducing new variables ϕ_i of type G to be zero and transposing w in all equations except for one:
$$\begin{cases} \phi_i = f_i(\boldsymbol{v}_i) - w \quad (i = 1, \dots, N{-}1) \\ w = f_N(\boldsymbol{v}_N). \end{cases}$$

B. Algorithm for Extraction of Simultaneous Equations

Model equations to be simultaneously solved during ODE calculation are undetermined without specification of independent and parameter variables in the model. Therefore, extraction of simultaneous equations from a model requires definition of the types of the variables.

A simulation of ODE is achieved in general by iteration of evaluating model equations $\boldsymbol{y}' = F(\boldsymbol{y}, t)$ with the initial or assigned values of $\boldsymbol{y}$, where $\boldsymbol{y}$ is a vector of differential variables. That is, the value of each differential variable is given at each step of ODE calculation. Recall that a vertex of type S or G is assigned to a variable that has the predetermined value at solving of the simultaneous equation system. Therefore, differential variables are of type S or G.

After expression of a model using the representation graph, extraction of simultaneous equation systems is achieved with interactively specifying variable types by the following algorithm:

1) Assign type G to all differential variables and introduced variables in adaptation into standard form.
2) Let the user choose parameter variables from all left-hand side variables of equations (except variables that are already assigned types) and assign type G to them.
3) Delete outgoing arcs of type G vertices, and then recursively delete vertices that become maximal by the deletion.
4) Delete all vertices that are unreachable to any vertices of type G.
5) Repeat the following steps until the types of all maximal vertices are assigned.
    a) Let the user choose a non-constant variable from maximal vertices and assign type A to it.
    b) Assign type S to maximal vertices that can not be of type A. A maximal vertex can be of type A, if the size of the Menger-type maximum linking is equal to the number of vertices of type A in assuming that the vertex is of type A.
    c) Delete vertices of type S and then recursively delete vertices that become maximal by the deletion.

As the result of this procedure, one or more subgraphs of the representation graph are extracted, in which all the maximal vertices are of type A and all the minimal of type G. The each subgraph represents a system of equations to be simultaneously solved. For a simulation of the model, all the variables of type S and G should be given the initial values, and all the variables of type A require the appropriate initial values in order to initiate a root-finding algorithm for simultaneous equations.

An illustration of the present algorithm follows. A sample model is defined as:

$$\begin{cases} v_5 = f_5(v_1, v_2, v_3) \\ v_7 = f_7(v_3) \\ v_6 = f_6(v_{11}) \\ v_8 = f_8(v_3, v_4) \\ v_9 = f_9(v_5) \\ v_{10} = f_{10}(v_5, v_6) \quad (= 0) \\ v_{11} = f_{11}(v_7) \\ v_{12} = f_{12}(v_7, v_8) \end{cases} \tag{2}$$

where v_{10} is introduced in place of 0. The representation graph is shown in Fig. 1. Vertex 10 of the representation graph is assigned type G in step 1. Now assume that v_{11} (vertex 11) is selected as a parameter variable (type G) in step 2. Then, vertex 6 is deleted in step 3 and vertex 4, 8, 9 and 12 are deleted in step 4, automatically. At this point, candidate variables for type A is narrowed down to three variables v_1, v_2 and v_3. If the user choose v_2 as an independent variable in step 5a, then vertex 2 is assigned type A, and vertex 1 is identified as type S through step 5b and hence deleted. Finally, the last candidate, v_3 (vertex 3) is chose as an independent variable and assigned type A. The resulting Menger-type linking is indicated by bold strokes. Consequently, the system has two systems of simultaneous equations; one system to be solved for the independent variable v_2 consists of equations on v_5 and v_{10}, and the other for the independent variable v_3 of equations on v_7 and v_{11}.

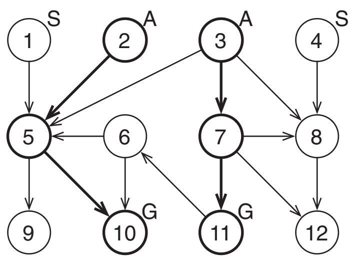

Fig. 1. Example of a Representation Graph

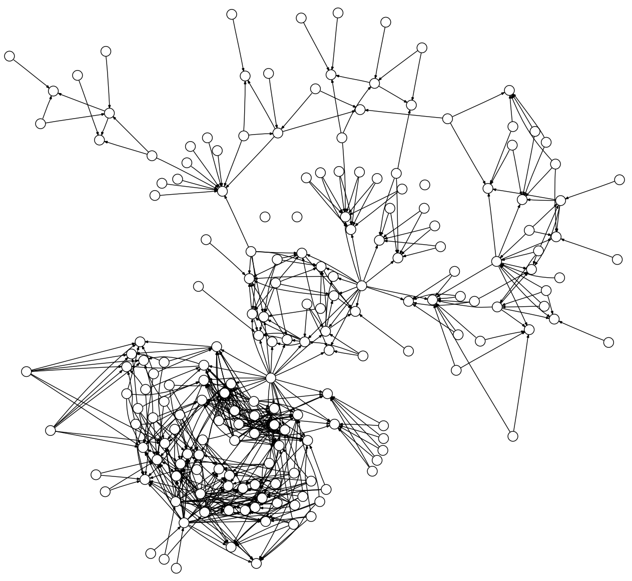
Fig. 2. Representation Graph of Saucerman03 Model

IV. Example of Application to a Cell Model

The present method was applied to a cell model on β-adrenergic control by Saucerman et al [9]. In this application, equations in the original papers were used without any modification.

The model includes twelve equations that have the left-side 0. These equations were first transformed into standard form by introducing variables of type G. The representation graph of the model in standard form is shown in Fig. 2. The model has no parameter variables in the left-hand side except for the introduced variables. Then, after the step

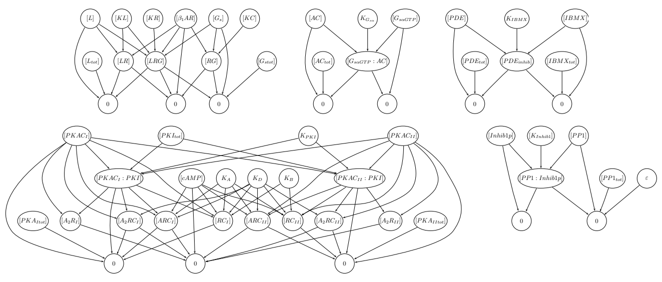

Fig. 3. Simplified Representation Graph of Saucerman03 Model

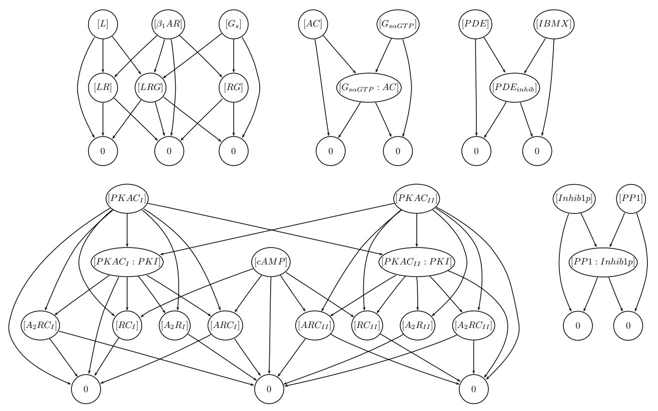

Fig. 4. Extracted Representation Subgraphs of Saucerman03 Model

4 of the present algorithm (sec. III-B), the representation graph was simplified into the graph shown in Fig. 3. At that time, 32 variables were picked out as candidates for independent variables. After choosing $[L]$, $[\beta_1 AR]$, $[G_s]$, $[AC]$, $[G_{s\alpha GTP}]$, $[PDE]$, $[IBMX]$, $[PKAC_I]$, $[PKAC_{II}]$, $[cAMP]$, $[Inhib1p]$, $[PP1]$ as independent variables by repeating the step 5 of the algorithm, five subgraphs shown in Fig. 4 were extracted. The each subgraph represents a system of simultaneous equations to be solved. Consequently, the model can be simulated with solving the simultaneous equations.

V. Discussion

In the example application (sec. IV), the model definition written in the published paper was used, not that in the CellML file registered in CellML repository. In the CellML file, simultaneous equations are altered to an approximate expression with differential equations so that the equations can be calculated without solving simultaneous equations. In other words, the CellML file does not exactly describe the original model. The alteration could be valid as an input file for simulators that do not support multiple simultaneous equations. However, the alteration is inappropriate as a formal representation of the published model. Moreover, such alterations may collapse modularity of model components. The proposed method enables to simulate a model in the original expression that contains simultaneous equation systems. Therefore, a CellML file of a model can be a formal representation and an executable input file for simulators at the same time.

A definition of a consistent set of independent and parameter variables, which is required for simulation of a model, should be separately described from a model definition in CellML, because the set of independent and parameter variables varies according to experimental protocols. PEPML [10], which we previously proposed as a representation format of experimental protocols, is capable of the definition.

The present method can not be applied to models that contain partial differential equations, because partial differential equations require the boundary condition additionaly. Since Theorem 1 gives only the necessary condition for the structural solvability of an equation system, an obtained set of independent and parameter variables by the present method may not be consistent, if more than one equation result in the same relationship, precisely, if the partial derivatives of equations in the simultaneous system are not algebraically independent.

VI. Conclusion

A method for analyzing simultaneous equations in a cell model was presented. The present method extracts subgraphs that represent systems of simultaneous equations from the representation graph of the model, with efficiently specifying independent variables. This method allows simulating declarative representations of cell models that contain equations to be simultaneously solved. This method will enhance usability of CellML and furthermore facilitate cell modeling and simulation.

References

[1] A. A. Cuellar, C. M. Lloyd, P. F. Nielsen, D. P. Bullivant, D. P. Nickerson, and P. J. Hunter, "An overview of CellML 1.1, a biological model description language," *SIMULATION*, vol. 79, no. 12, pp. 740–747, Dec. 2003.

[2] M. Iri, J. Tsunekawa, and K. Murota, "Graph-theoretic approach to large-scale systems of equations : Structural solvability and block-triangularization," *Transactions of Information Processing Society of Japan*, vol. 23, no. 1, pp. 88–95, Jan. 1982, in Japanese.

[3] K. Murota and M. Iri, "Structural solvability of systems of equations — a mathematical formulation for distinguishing accurate and inaccurate numbers in structural analysis of systems," *Japan Journal of Applied Mathematics*, vol. 2, pp. 247–271, 1985.

[4] K. Murota, *Systems Analysis by Graphs and Matroids: Structural Solvability and Controllability*, ser. Algorithms and Combinatorics. Springer-Verlag, Dec. 1987, no. 3.

[5] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM Journal on Computing*, vol. 4, no. 4, pp. 507–518, Dec. 1975.

[6] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," *Soviet Math. Doklady*, vol. 11, pp. 1277–1280, 1970.

[7] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, vol. 19, no. 2, pp. 248–264, Apr. 1972.

[8] M. Iri, J. Tsunekawa, and K. Yajima, "The graphical techniques used for a chemical process simulator "JUSE GIFS"," in *Proceedings of IFIP Congress*, vol. 2, 1971, pp. 1150–1155.

[9] J. J. Saucerman, L. L. Brunton, and A. P. Michailova, "Modeling β-adrenergic control of cardiac myocyte contractility in silico," *Journal of Biological Chemistry*, vol. 278, no. 48, pp. 47 997–48 003, Nov. 2003.

[10] T. Shimayoshi, A. Amano, and T. Matsuda, "A generic representation format of physiological experimental protocols for computer simulation using ontology," in *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Engineering in Medicine and Biology Society, Aug. 2007, pp. 382–385.