Modeling Theory and Software Architecture of the E-CELL Simulation System

Kouichi Takahashi [12]
shafi@sfc.keio.ac.jp
Tom Shimizu [14]
tom@sfc.keio.ac.jp
Katsuyuki Yugi [13]
t95980ky@sfc.keio.ac.jp
Masaru Tomita [13]
mt@sfc.keio.ac.jp

[1] Laboratory for Bioinformatics
[2] Graduate School of Media and Governance
[3] Department of Environmental Information
Keio University, 5322 Endo, Fujisawa 252-8520, Japan
[4] Department of Zoology, University of Cambridge, CB2 3EJ, United Kingdom

1 Introduction

We have been developing E-CELL, a generic software environment for whole cell modeling and simulation. E-CELL simulates the behavior of model cell by computing interactions among molecular species it contains. E-CELL allows the user to define chemically discrete compartments such as membranes, chromosomes and cytoplasm. The collection of substance quantity values in all of the cellular compartments comprises internal representation of the cell state, which can be monitored and/or manipulated using the various graphical user interfaces. E-CELL attempts to provide a framework not only for modeling metabolic pathways, but also for higher-order cellular phenomena such as gene regulation network, DNA replication, and other occurrence in the cell cycle. A major challenge of the project is to develop a method of modeling which is sufficiently robust to accommodate the drastic difference in time scale among these processes.

2 Modeling Theory

The E-CELL employs *structured Substance-Reactor model* to construct and simulate the cell model. The structured Substance-Reactor model consists of three classes of objects, *Substance*, *Reactor*, and *System*, representing molecular species, reactions and functional/physical compartments, respectively. To model chromosomes and other genetic materials, the system also has sophisticated data structures such as *Genome*, *GenomicElement*, and *Gene*.

Assuming rapid equilibrium in each compartment, a state of the cell at a point in time and dynamics of the cell are represented as a set of concentration vectors and reaction kinetics, respectively. Therefore, simulation of the cell can be viewed as integration of ordinary differential equations (ODEs). The numerical integration is performed in distributed form with *Stepper* and *Integrator* class objects in the system. With this design, the system is capable of multi-compartment, multi-scheme and multi-phase numerical integration. In other words, time step sizes and integration algorithm can vary among different compartments, and therefore parallel computation can be applied to "stiff" differential systems like the cell.

The problem of *stiffness* occurs in a system where there are very different scales of the time variable on which the dependent variables are changing. The stiffness often results in various kinds of numerical errors, some of which are fatal for the simulation. One of such problems is called *Less Than*

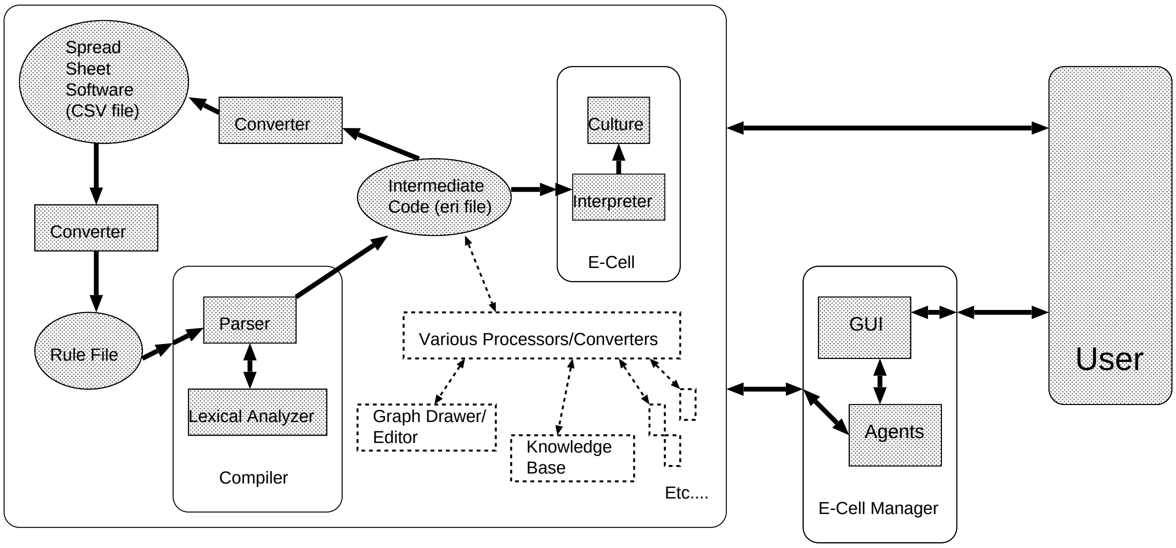

Figure 1: The E-CELL Simulation Environment.

Zero (LTZ) problem, in which concentration of substance becomes a negative value. More precisely, it occurs when

$$S_j(t) \ < \ \{\sum_i (\nu_{ij} \ v_i(t)) + \sum_k (\nu_{kj} \ v_k(t))\} \cdot \Delta t \tag{1}$$

where $S_j(t)$ is concentration of jth substance, $v_i(t)$ is velocity of ith reaction, ν_{ij} is stoichiometric constant of the ith reaction for the jth substance, and Δt is time step size. The LTZ problem is especially complicated when the substance is consumed by two or more reactions as their substrate.

We are trying to cope with the LTZ problem using the following three techniques: multi-compartment numerical integration, adaptive step size control and partial recalculation.

3 Software Architecture

The E-CELL System is written in C++ and designed with object oriented MVC model, which facilitates independent development of the simulation engine, user interfaces and the cell model. Each reaction kinetics are encapsulated into the *Reactor* class objects. In addition to general reaction schema provided by the system, task-specific reaction schema can be defined by the user using C++ language without detailed knowledge about implementation of the other parts of the system.

The E-CELL simulation environment consists of several software components such as core simulation system (E-CELL System), E-CELL Manager, rule file compiler, and other various data converters/processors (Fig. 1). Besides the system specific E-CELL Rule format, the user can now write simulation rules using any commercially/publicly available spread sheet software. The rules are compiled into .eri (E-CELL Rule Intermediate) format, which can be interpreted by the simulation system. The core system reads compiled rule, starts simulation and outputs logged data. The simulation process can be automated using E-CELL Script language(.ecs). The whole process of rule writing, simulation, and analysis is managed by the E-CELL Manager.

Acknowledgments

This work was supported in part by a Grant-in-Aid for Scientific Research on Priority Areas 'Genome Science' from The Ministry of Education, Science, Sports and Culture in Japan.