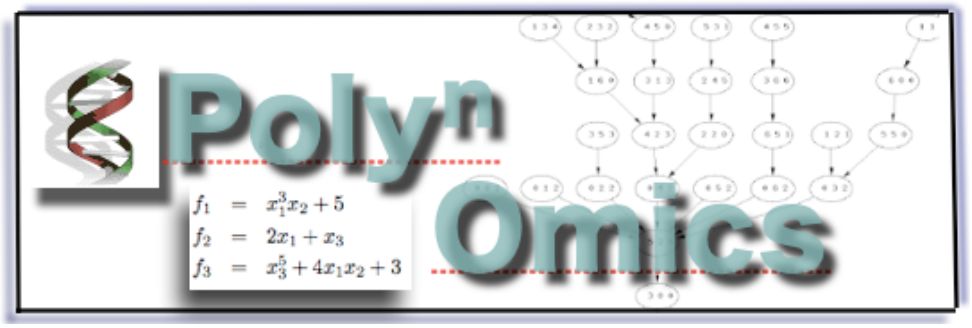
WORKSHOP ON SOFTWARE DEVELOPMENT FOR ALGEBRAIC MODELS



Elena Dimitrova
Luis Garcia-Puente
Franziska Hinkelmann
Abdul Jarrah
Brandilyn Stigler
Paola Vera-Licona
Reinhard Laubenbacher

SAMSI
February 24-26, 2009

TABLE OF CONTENTS

INTRODUCTION

OVERVIEW

Algebraic models (polynomial dynamical systems, Boolean networks, logical models, Petri nets, etc.) are becoming increasingly popular in cases where not enough data is available to construct detailed differential equations models. There now exist several large-scale models of this type for a variety of molecular networks.

A major problem with the construction of large-scale algebraic models is that there are no sophisticated tools available, comparable to ODE tools. Most importantly, the tool of fitting ODE model parameters to available data is key in continuous model construction, but completely absent for algebraic models. Also, tools like bifurcation analysis, sensitivity analysis, stability analysis are all unavailable to the algebraic modeler. However, we already have or can develop several or all of these tools in the polynomial dynamical systems framework.

The general process of constructing an algebraic model is similar to that of constructing a continuous model. In the continuous case, based on prior biological information, one constructs a wiring diagram. To construct an ODE model, one decides on a particular form of the ODE for each variable, or leaves it generic, if nothing is known about the mechanisms involved. Then one specifies those parameters that are known in each ODE. Typically, one is left with unknown parameters. Using available data, one can then estimate those parameters by fitting the model to the data. Having done that, one can analyze the model and the parameter choices.

COMPONENTS

1. The most common model types are Boolean networks and multistate logical models. There is now an algorithm (paper in preparation--Alan, Abdul, R.) that translates a logical model into a PDS. We are working on the other direction, to translate a PDS into a logical model. This implies that all tools available in the polynomial framework are in principle available to the entire algebraic model community. Everything that follows is in PDS language.

2. Parameter estimation. There are of course no parameters per se in a PDS model, unless one views the coefficients of the individual monomials as parameters. (Making this analogy, however, might lead to problems in interpretation.) The analog of a parameter in an algebraic model is a function on a node or a partial function. We can fit a function or a partial function to available discretized data. This can be done applying one of our network inference algorithms to the node at hand. Having several different methods

available corresponds to having different parameter estimation methods.

Regarding the possible choice of a special function form, we have Abdul's algorithm to choose a parameter from the parameter space (f+I) that is a nested canalyzing function. These have been shown to particularly well-suited as choice.

I believe that if we recast network inference as parameter estimation, then we will have an easier time convincing people of its usefulness. And, of course, this includes estimating ALL parameters, i.e., reverse-engineering.

3. Simulation. Currently we have a primitive version of DVD, which is sufficient for many purposes.

4. Sensitivity analysis. This corresponds to varying a parameter over its range and seeing whether it affects the dynamics in significant ways. We do not have a systematic tool to do this. However, being able to describe the whole parameter space (f+I) for a given parameter makes this possible in principle. Another view of this issue is that sensitivity analysis could include the study of model dynamics over the entire Groebner fan. This represents sensitivity to “parameters” used in the modeling process.

5. Bifurcations. Here we need to be able to study the effect of varying a parameter on the steady states of the model. Again, in principle, we should be able to do this, but no method exists at the moment. Another aspect of this is that we need to understand the dynamics represented in the entire model space f+I.

6. Stochastic models. This is becoming increasingly important. Logical models are already stochastic with respect to the update schedule. People have also been considering function-stochastic networks. Franziska has been writing code to simulate a special class of function-stochastic systems, namely linear systems with delays.

7. Code for experimental studies. Alan has written code to generate many random Boolean networks with special graph topologies and special functions. He has also code to do large-scale studies of the dynamics of such networks. And we have a high school student who is working on a new version of DVD that has several stochastic features.

An initial package need not contain all those components, so we need to focus on what can be done quickly.

PROPOSED WORKSHOP GOALS AND AGENDA

The goal of the workshop is to restart our software development process and to make progress in producing a version of our code that is usable by others. The hope is that an integrated package can be produced that contains the basic functionalities of data discretization, parameter estimation, simulation, and possibly parameter estimation using nested canalyzing functions, could be produced quite quickly. Having software that translates between PDS and logical models would be extremely useful as well.

Day 1.

Evening: Working Dinner. Discussion of plan, package architecture, and functionalities to be included.

Day 2

8-9:30 Brainstorming; formation of break-out groups
9:30-12 Break-out sessions/code development
12-1:30 Lunch
1:30-4 Break-out sessions/code development
4-6 Discussion: each group gets time to present to whole group ideas and questions.
6-8 Working dinner

Day 3

8-12 Break-out sessions/code development
12-1:30 Lunch
1:30-4 Break-out sessions/code development
4-6 Assessment of progress made and plan to complete project.

OVERVIEW

OVERVIEW OF SOFTWARE BY REINHARD

Main Objectives: The idea is to have two main components: *Model Building* and *Simulation*.

MODEL BUILDING

INPUT

1. Biological Info
2. Data
    a. **Data Discretization**

        Different Methods

        Choice of thresholds

OUTPUT

Models

Parameter Estimation

Wiring Diagram and/or Dynamic Model

- Paola
- Elena
- Brandy

- Abdul

How does the user choose which modeling method?

What is the default output?

Simulation

INPUT: Model

OUTPUT: dynamics

- DVD
- DVD2
- Stochastic Features:
    - Update schedule
    - Function Stochasticity

Elena (Groebner Fan)

This method constructs Stochastic PDS's

Flow chart:

Time Series (V) -> I= I(V) -> G fan (I)

Under the assumption that the size of the Groebner cone relates the cone´s biological relevance.

Hence points are being picked to construct each one of the **polynomial functions**

Now, from these stochastic polynomial functions a **wiring diagram** is being inferred assigning to each interaction a probability value with respect to the sizes of the cones where such interaction appears.

This code is implemented in Macaulay

Brandy (MinSets + Polynomial Inference + Grobner Fan)

For each minset *V* do

- Compute ideal *I* of *Dv* (data restricted to coordinates defined by *V*)
- Compute the Groebner fan G of *I*
- Compute a transition function f in terms of *V*
- Compute NF ={ f%GB: GB in G}
- Select most frequently occurring normal form(s)

This code has been implemented in Macaulay

In general this method has as input only the data, however Brandy has some code "problem dependent"

When we know whether an edge should be present or absent

Where prior biological information is available

Paola (Evolutionary Algorithm, Boolean Models)

This modeling approach is design to build Boolean models where functions do not necessary fit the data exactly under the consideration that the input data may contain noise

The code is implemented in C++ (Visual studio and Linux versions available)

Input data is in compatible with the input format of MinSets master files

Input

Mandatory: Boleanized Time Courses

- Wildtype Time Courses or
- Knockout Time Courses

Optional: Prior Information about structure of wiring diagram

- Prior biological information (e.g. prior knowledge from literature)
- Wiring diagram structure from previously applying another modeling method (e.g. heterogeneous data available like static data, then another reverse engineering ad hoc can be used for such data and input the wiring diagram generated by such method)

Output

Highest scored Boolean Models

Abdul (Nested Canalizing Functions)

Franziska (Simulation for Random Delay Nets -Linear and Monomial Systems)

Calculates length and probability of limit cycles and feedback loops for a given random delay network

Written in C++

INTERFACE

User Types

Guest will be able to run a **toy example** from the server (similar to what can be done in DVD currently) and will be able to register in order to use the software for their own data

Registered users will be able to use the software with **their own uploaded data**

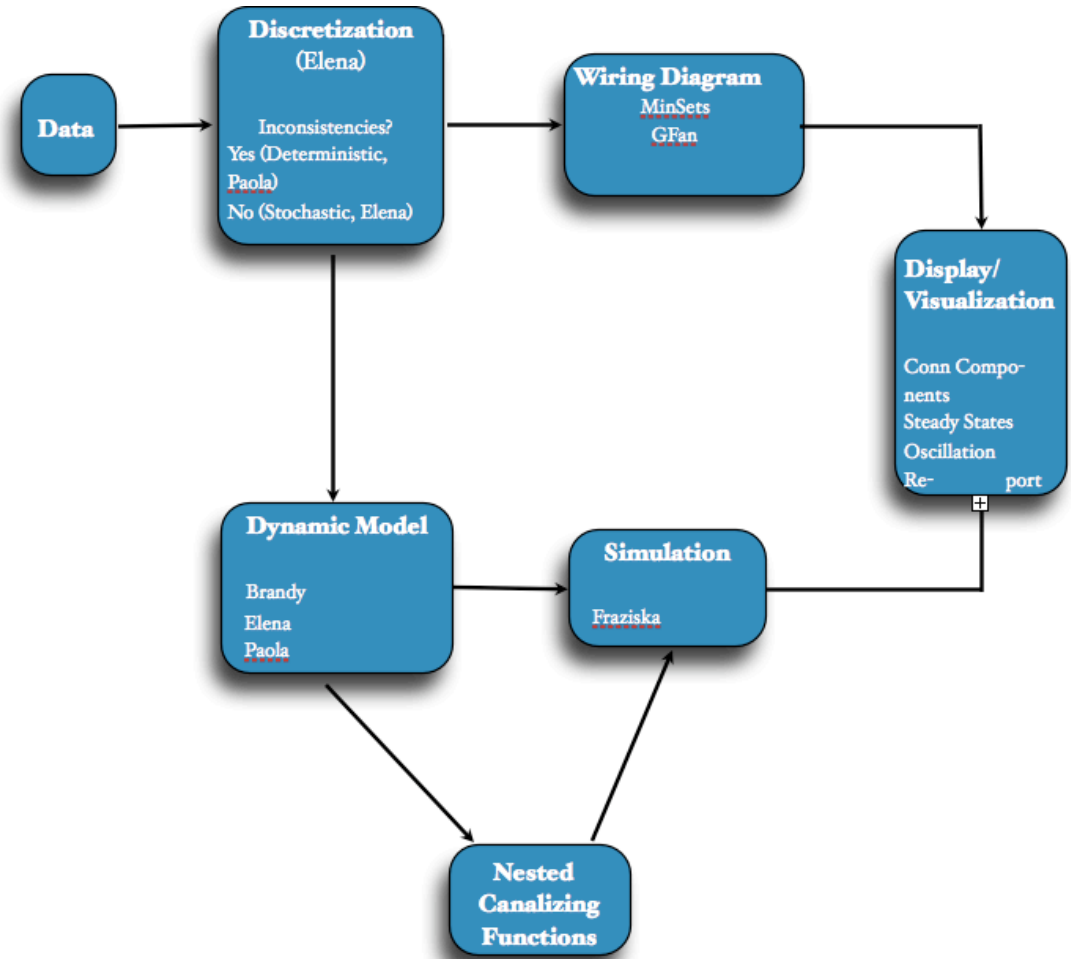
Input from the User

Type of Data

- Boolean
- Non Boolean (multi-state, continuous)

Type of Model

- Static
  - Default: If n< 100 Elena's method and MinSets Otherwise
- Dynamic
  - Deterministic
    - Default: Parallel Update
    - Sequential Update (with update schedule)
    - Consistent Data (Brandy's GFan)
    - Inconsistent Data (Paola's EA)
  - Stochastic
    - Function (Elena's => Fran)
    - Update (Brandy/Paola => Fran)
    - Function + Update (Elena + Adding Identity Function => Fran)


Data
Discretization
(Elena)
Inconsistencies?
Yes (Deterministic,
Paola)
No (Stochastic, Elena)
Wiring Diagram
MinSets
GFan
Display/
Visualization
Conn Compo-
nents
Steady States
Oscillation
Re- port
Dynamic Model
Brandy
Elena
Paola
Simulation
Fraziska
Nested
Canalizing
Functions