Computational Design and Verification of Genetic Regulatory Circuits

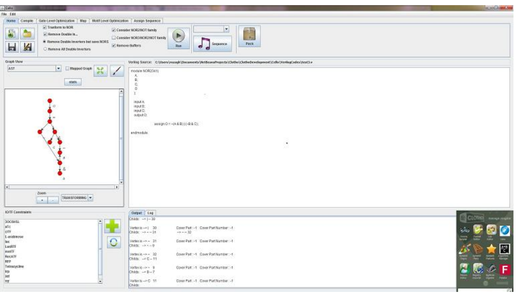
Rishi Ganguly[3] , Roza Ghamari[1], Evan Appleton[3], Swapnil P. Bhatia[1],Boyan Yordanov[2], Ebru Aydin Gol[2], Calin Belta[2], Douglas M. Densmore[1]

[1]*Department of Electrical and Computer Engineering, Boston University*
[2]*Department of Systems Engineering, Boston University*
[3]*Department of Bioinformatics, Boston University*

Abstract

In our research efforts, we are interested in developing a framework that allows us to analyze the behavior of synthetic gene networks formally, from high level specifications resembling a natural language. We have developed a software tool suite (*www.clothocad.org*) that enables a user to specify a genetic circuit behavior using an appropriate HDL (Hardware Description Language) such as Verilog. The back-end compiler automatically generates genetic-regulatory network (GRN) topologies with assigned "parts" from our libraries that realizes the specifications. The next step in our flow is to integrate available characterization data for the assigned parts and generate mathematical models capturing all possible behaviors of the generated GRNs. We delineate algorithms to create finite abstractions of these mathematical models, and run standard model checking algorithms against the specifications re-expressed in Linear Temporal Logic (LTL) formulations. A parameter scan of input signal concentrations allows us to quantifiably suggest the optimal GRN among all the generated ones.

Process Overview



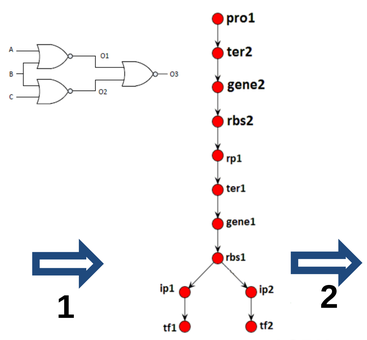
Cello

Step 1:

• Use a clotho-based IDE called “Cello” for Verilog programming to structurally/behaviorally specify a genetic circuit.

• Cello compiler optimizes both “gate level” (reducing the number of boolean algebraic terms) and “motif level” (removal of redundant genetic elements) and produces Abstract Genetic Regulatory Circuits (AGRN)

1



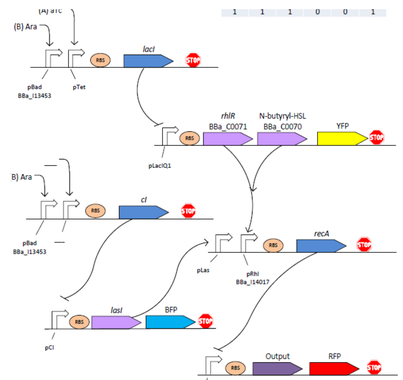
AGRN

Step 2:

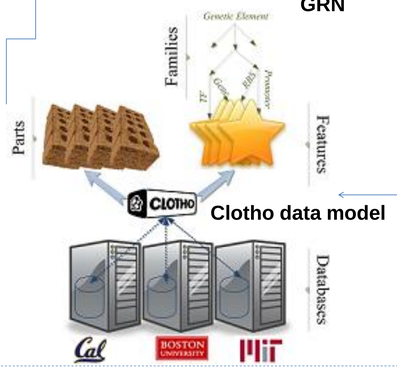
• Map the network elements (such as Promoters, genes, Ribosome Binding sites) with “parts” from our database

• Produce assembly-ready genetic regulatory networks(GRN)

2

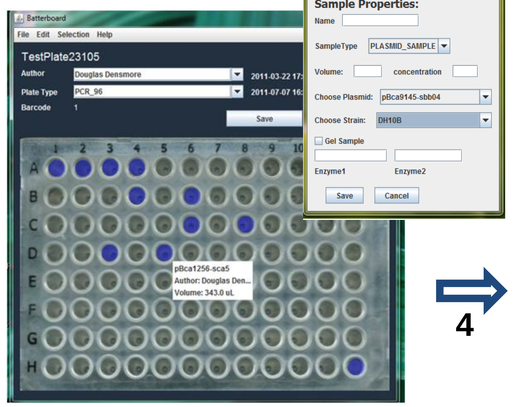


GRN



Clotho data model

3



Batterboard

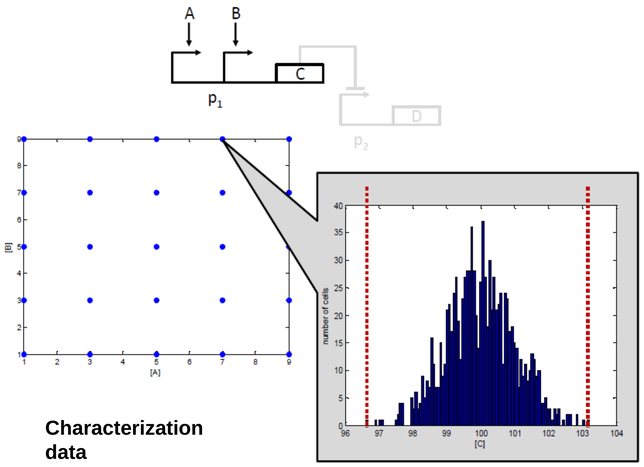
Step 3:

• Concurrent to step 1 and 2: compile a registry of standard biological “parts”.

• Categorizing these parts to biologically relevant features (such as a promoter, a gene-coding region, RBS, etc) .

• Prepare plasmid-preps comprising parts, create samples with specified volume, concentration, etc, use sample tracking tool: “batterboard”.

4



Characterization data

Step 4:

• Characterization of inducible promoters in our database by inferring rate of expression using external inducer molecules.

• Measure the concentration of output protein (tagged with fluorescent protein) by means of flow-cytometry (using a FACS machine)

• Construct a lattice of data points, each being a different combination of inducer concentrations

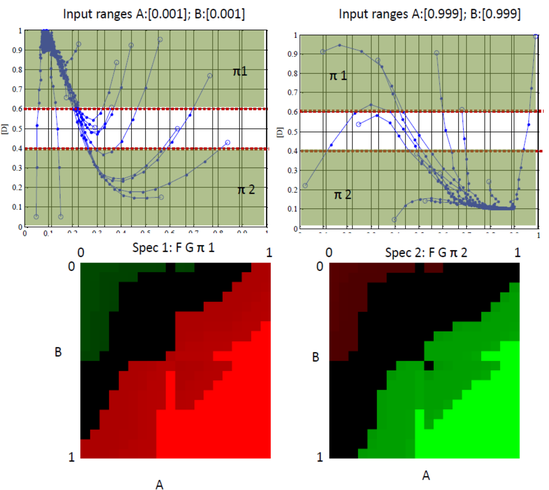
5

Step 5:

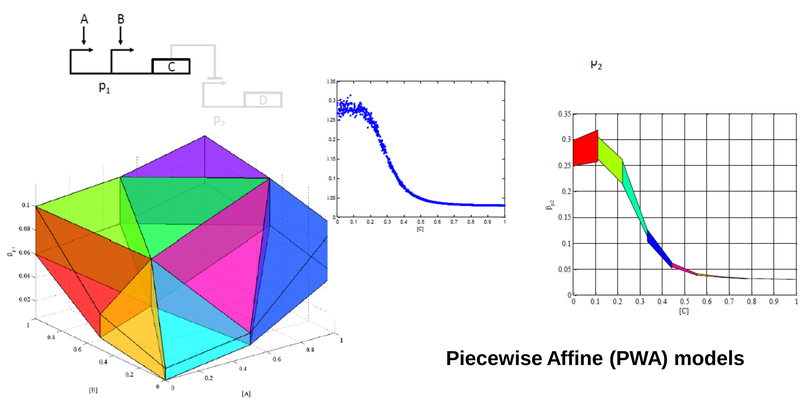
• Integrate characterization data from step 4 with the generated circuit topologies from step 2.

• Generate Piecewise affine (PWA) models capturing all the experimental data points. The data is separated into different regions (shown in different colors) and captured through a single polytope in each region.

6



Conclusion : From the generated models from step 5, we can use Linear Temporal Logic(LTL)-based model-checking algorithms to predict, under what input concentrations and other experimental conditions, will a particular circuit produce output(s) according to user defined specification. From several possible circuit-configurations, our algorithm can thus enable us to choose the most optimal one, which we can then go ahead and build in the lab. Shown here are sample trajectories and the computed input space regions satisfying or violating a particular specification.



Piecewise Affine (PWA) models

www.synberc.org