A Domain Specific Language Based on Membrane Computing for Synthetic Biology

Marian Gheorghe

Verification and Testing Group
Department of Computer Science
University of Sheffield, UK

2nd Int School on Biomolecular and Biocellular Computing (ISBBC'13) Madrid, 23-25/09 2013

Domain Specific Languages for Bio-inspired Computing and Synthetic Biology

Marian Gheorghe

Verification and Testing Group
Department of Computer Science
University of Sheffield, UK

2nd Int School on Biomolecular and Biocellular Computing (ISBBC'13) Madrid, 23-25/09 2013

Summary

1. Membrane computing
   1. Roots & definition
   2. Main research topics

2. Domain specific language for bio-inspired computing
   1. Kernel P systems
   2. Examples: partition problem & firing squad problem

3. Probabilistic P systems
   1. Features of a domain specific language
   2. Formal verification & natural language patterns
   3. Simple example

1. Membrane computing – Introduction, roots and definition

Membrane computing (I)

- **A computational model**, belonging to **natural computing**, abstracting from the functioning and structure of the living cells (**Membrane or P systems**)

- **Three** essential features: a certain **structure** of **membranes** delimiting **regions** (compartments, cells), some ***multisets of objects*** and finite ***sets of rules*** associated to regions

- **Evolving** in steps, from one configuration to another one according to a given **strategy** (more often is **maximal parallelism** = the objects not allocated to rules in a computation step can not be allocated to any of the rules)

- **Rules** can transform objects, move objects, and even modify the membrane structure (creation/division/dissolution/moving)

Gh Păun: JCSS 2000 (TUCS 1998)

Membrane computing (II)

- Membrane systems generalise (include) L systems and DNA computing, (PC) grammar systems

- Similar models:
  - cellular automata, networks of evolutionary processors, calculus with cilliates, reaction systems
  - pi-calculus, mobile ambients, brane calculus, Petri nets

Rozenberg, Bäck & Kok: Handbook of NC, 2011
Kari, Rozenberg: CACM, 2008

Membrane computing – Basic definition

$P = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, i_0)$ - P system

where

O – an alphabet (finite set)

μ – a membrane structure with n membranes (regions)

$w_1, \ldots, w_n$ – multisets over O; w_i – initial values

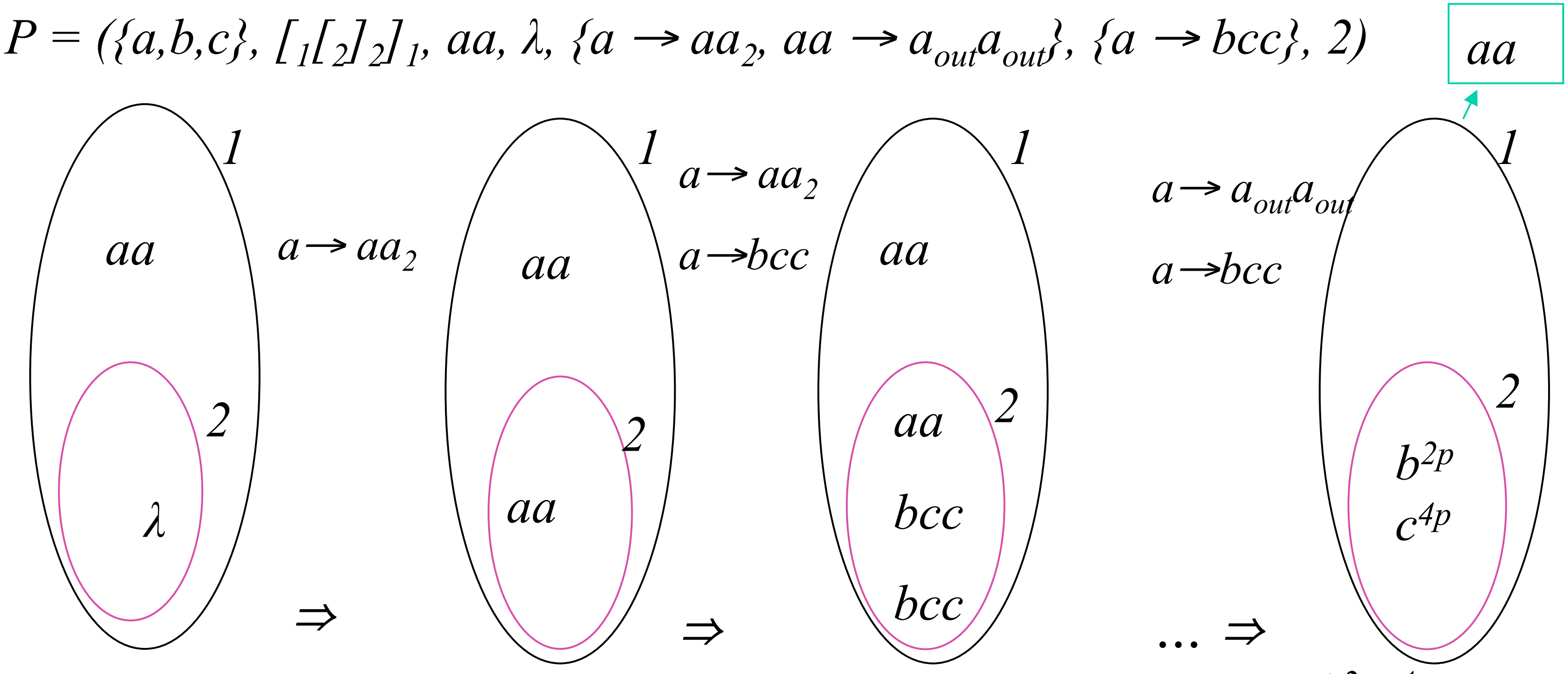
$R_1, \ldots, R_n$ – sets of rules

i_0 – the output cell

R_i – evolution and communication rules: $v \rightarrow w$; v, w – strings over O + some indications of target regions in w

Simple example

$P = (\{a,b,c\}, [_1[_2]_2]_1, aa, \lambda, \{a \rightarrow aa_2, aa \rightarrow a_{out}a_{out}\}, \{a \rightarrow bcc\}, 2)$



$w = b^{2p}c^{4p}, p \geq 0$

Characteristics: distributed device; non-determinism; maximal parallelism

Membrane systems variants

$P = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, i_0)$ - P system

- **Objects**: simple symbols, strings
- **Rules**: specific communication functions, splicing (DNA), probabilities, conditions, use priorities and/or states, structure changes…
- **System's behaviour**: maximal or limited parallelism, sequential, asynchronous, stochastic…
- **System structure**: tree, graph (tissue, neural…), population

1.2 What has been studied?

Basic membrane systems. Computational power

$NOP_n(x)$ – the family of **sets of natural numbers computed by membrane systems** with at most n membranes; * - arbitrary number of components;

x = ncoo – context free rules (non-cooperative, at most one element on lhs), x = coo – context dependent rules

(1) $NOP_*(ncoo) = NOP_1(ncoo) = NCF$ – basic P systems

(2) $NOP_*(coo) = NOP_1(coo) = NRE$

Observations.

- Only one component (membrane/region) is used (no hierarchy)
- No communication
- It (partially) reflects Chomsky hierarchy
- Simulate context-free grammars/matrix grammars

Population P systems – Some results

Components dynamically connected & environment.

$NOPP_{n,k}(x,y)$ – the family of sets of natural numbers computed by Population P systems with at most n components and at most k components in each connected component; * - arbitrary no of components;

x = ncooCom – non-cooperative communicating rules *(a;b,in), (a;b,enter), (b,exit);* x = cooCom - cooperative communicating rules; y = $n\alpha$ - without bond making rules $(i,x_1;x_2,j)$

(1) $NOPP_{*,*}(ncooCom,n\alpha) = NOPP_{1,1}(ncooCom,n\alpha) = NCF$

– population P systems with evolution and symport/antiport/uniport rules

(2) $NET0L \subseteq NOPP_{4,2}(ncooCom, \alpha_1)$

(3) $NOPP_{2,2}(cooCom,n\alpha) = NRE$

(4) $NOPP_{4,1}(cooCom,n\alpha) = NRE$

Bernardini, Gheorghe: JUCS, 2004;
Bernardini, Gheorghe, Margenstern, Verlan: TCS, 2008

Problems

- Computational power & descriptional complexity
- Structural operational semantics and relationships with other computational models: process algebra and Petri Nets (another semantics)
- Formal verification - Model checking: Maude, NuSMV (basic systems), Spin (kP systems) and Prism (stochastic systems)
- P systems testing
- Model various systems (applications)

Ref: Handbook on MC, 2010; http://ppage.psystems.eu/

2. Domain specific language for bio-inspired computing: Kernel P systems

Kernel P systems – Motivation (Research plan)

Why a new model - kP system ?

- need to have a slightly more general model allowing various classes of P systems or concepts utilised to be mapped into this model
- create a framework where systems using these models can be analysed (formally verified – initially model checking and testing)
- solve satisfactorily problems already coded with other P systems
- integrate with current tools

Initially introduced in February 2012, BWMC, as a PhD project

Kernel P systems – Informal definition

kP system, for short, is characterised by

- a dynamic structure, as a network of compartments
- multisets of objects in each compartment
- rules can have **guards** and include
  - rewriting and communication rules
  - structural rules (ex. membrane division, dissolution)
- each compartment has an explicit execution strategy

kP system basic definition

$$k\Pi = (A, \mu, C_1, \ldots, C_n, i_0)$$

where

- A – an alphabet (finite set)
- $\mu = (V, E)$ – a membrane structure, a graph with vertices denoting compartments and edges defining connected compartments
- $C_1, \ldots, C_n$ – compartments, where each $C_i = (t_i, w_i)$, $1 \leq i \leq n$, consists of
  - a compartment type, t_i, and
  - an initial multiset, w_i;
  - each compartment type is a tuple, $t_i = (R_i, \sigma_i)$, $1 \leq i \leq n$, with R_i, a set of rules and σ_i, an execution strategy of the rules from R_i
- i_0 – the output cell

kP system rules

A rule from a compartment $C_l = (t_l, w_l)$ of $k\Pi$, has one of the following types

- rewriting and communication: $x \rightarrow y \{g\}$, where x is a non-empty multiset over A, y is a multiset and g is a guard, $y = (a_1, t_1) \ldots (a_h, t_h)$; t_i is either t_l or is the type of a compartment C_i which is linked with C_l
- membrane division: $[x]_\alpha \rightarrow [y_1]_{(\alpha,1)} \ldots [y_h]_{(\alpha, h)} \{g\}$, with $\alpha = t_l$ and $(\alpha, i) = t_i$; compartment C_α is divided into $C_{(\alpha,1)}, \ldots, C_{(\alpha, h)}$; links are inherited
- membrane dissolution: $[x]_\alpha \rightarrow \lambda$; the compartment and its links are destroyed

A guard, g, is

- a relational term of the form $op\ a^n$, where op is a relational operator, a is an element of A and n is a positive integer; if w is the current multiset then $|w|_a$ denotes the number of occurrences of a w - $op\ a^n$ denotes $|w|_a\ op\ n$;
- relational terms can be connected, as usual in Boolean expressions, by Boolean operators $\vee$, $\wedge$ or negated by ⁄.

Execution strategy - σ

For a compartment type, $t = (R, \sigma)$, and $r, r_1, \ldots, r_s$, labels of rules, the execution strategy, σ, is defined by the following regular-like expression:

- $\sigma = \lambda$ - no rule from the compartment instantiating t is executed
- $\sigma = \{r\}$ - the rule r is executed
- $\sigma = \{r_1, \ldots, r_s\}$ – one of the rules labelled $r_1, \ldots, r_s$, is non-deterministically chosen and executed; if none is applicable then none is executed (similar to a skip)
- $\sigma = \{r_1, \ldots, r_s\}^*$ – the rules are applied an arbitrary number of times, including 0; this is arbitrary parallelism
- $\sigma = \sigma_1 \& \ldots \& \sigma_s$ – $\sigma_1, \ldots, \sigma_s$ – are executed sequentially; if one is not applicable the execution stops, equivalent to a jump at the end of the sequence
- $\sigma = \{r_1, \ldots, r_s\}^T$ – the rules are executed according to the maximal parallelism strategy

kP system example

Component types $t_i = (R_i, \sigma_i)$, $1 \leq i \leq 3$, where

$R_1 = \{r_1{:}a \rightarrow a(b,2)(c,3)\ \{\geq p\},\ r_2{:}p \rightarrow p,\ r_3{:}p \rightarrow \lambda\}$; $\sigma_1 = Lab(R_1)^T$ (*max par*)
$R_2 = \{r_1{:}b \rightarrow (b,0)c\ \{\geq p\},\ r_2{:}p \rightarrow p,\ r_3{:}p \rightarrow \lambda\}$; $R'_2 = \{r_4{:}[\,]_2 \rightarrow [\,]_2[\,]_2\{\geq b^2 \wedge \geq p\}$;
$\sigma_2 = Lab(R_2)^T \,\&\, Lab(R'_2)$, (*max par* followed by *seq*); and
$R_3 = \Phi$, $\sigma_3 = \lambda$.

$k\Pi = (\{a,b,c,p\}, \mu, C_1, \ldots, C_4, C_4)$, $\mu = C_1\text{-}C_2\text{-}C_3\text{-}C_4$.
$C_1 = (t_1, w_{1,0} = a^3p)$, $C_2 = C_3 = (t_2, w_{2,0} = w_{3,0} = p)$, $C_4 = (t_3, w_{4,0} = \lambda)$

$(a^3p, p, p, \lambda) \Rightarrow (a^3p, b^2p, bp, c^3) \Rightarrow (a^3, b^2c^2, b^2c^2, bcp, c^6)$
$\mu = C_1\text{-}C'_2\text{-}C_3\text{-}C_4$; $C_1\text{-}C''_2\text{-}C_3\text{-}C_4$; C'_2 & C'_2 inherit C_2 links
$(r_1r_1r_1r_2, r_2, r_2, \lambda)$ $(r_1r_1r_1r_3, r_1r_1r_3 \,\&\, r_4, r_1r_2, \lambda)$

$(a^3, b^2c^2, b^2c^2, bcp, c^6) \Rightarrow (a^3, b^2c^2, b^2c^2, c^2, c^6)$ – final;
$(\lambda, \lambda, \lambda, r_1r_3, \lambda)$

Compartment C_4 contains c^6 and environment: $b^3\, b$ in the final configuration.

2.1. Partition problem

NP-complete problems

- P systems with active membranes and electric charges mapped into kP systems and solutions to some NP-complete problems provided (3-colouring, subset sum, partition etc)

- Scenario
  - Generate enough space
  - Generate (almost) all the solutions
  - Check the validity of each solution
  - Provide answer

Partition problem

Given a finite set V, a function *weight* on V with positive integer values,

$weight : V \rightarrow Z^+$,

and a number K, check whether there exists a partition W_1, W_2 of V, such that $weight(W_1)=weight(W_2)=K$.

$weight(W)$ means the sum of the weights of the elements of W.

Partition problem - codification

Given $V=\{v_1, \ldots, v_n\}$ with $weight(v_i) = k_i$,
we build a kernel P system with two compartments, where

$c(V)$ denotes the multiset with elements v^k, where v is an element of V such that $weight(v) = k$

Partition problem – solution

Solution sketch:

1. Two compartments, C_1 and C_2, connected, with $w_1=S$, $w_2=A_1c(V)$
2. The subsets, W_1, W_2, of V, are generated with elements from B_1, …, B_n; each B_i, corresponds to a v_i from V; the subset W$_1$, is generated by using membrane division rules

$[A_i]_2 \rightarrow [B_i\ A_{i+1}]_2\ [A_{i+1}]_2$, $1 \leq \mathrm{i} \leq \mathrm{n}\text{-}1$,

$[A_n]_2 \rightarrow [B_n\ X]_2\ [X]_2$ (in *n* steps) - W_2 is its complement

1. The weights of the two complementary subsets, W_1 and W_2, are matched up by rewriting rules $v_i\ v_j \rightarrow v\ \{=B_i \wedge /=B_j \wedge = X \vee /=B_i \wedge =B_j \wedge = X\}$

2. The solution, if exists, is communicated to compartment ‘1’ by using a rule $Y \rightarrow (T,1)\ \{<v_1 \wedge \ldots \wedge <v_n \wedge =v^k\}$

Properties.

Generic: Eventually there will be an answer, either “*yes*” or “*no*”

Partition problem – weak variant

Check whether there exists a subset *W* of *V*, such that *weight(W)=K*

Obs. Follow almost the same algorithm, but a new subset is generated when its current weight is less than *K* & the solution validity is checked at each step.

Weak partition problem – solution

Solution sketch:

1. Two compartments, C_1 and C_2, connected, with $w_1=S$, $w_2=A_1c(V)$
2. The subsets W of V, are generated with elements from B_1, …, B_n; each B_i, corresponds to a v_i from V; the subset W, is generated by using membrane division rules

$[A_i]_2 \rightarrow [B_i\ A_{i+1}]_2\ [A_{i+1}]_2\ \{<v^k\}$, $1 \leq \mathrm{i} \leq \mathrm{n}\text{-}1$,

$[A_n]_2 \rightarrow [B_n]_2\ []_2\ \{<v^k\}$; (in n steps).

1. The v_i's are transformed into v's when B_i appears in a compartment, i.e., in W

   $v_i \rightarrow v\ \{=B_i\}$;

2. The solution, if exists, is communicated to compartment '1' by using a rule

   $Y \rightarrow (T,1)\ \{((=B_1 \wedge \geq v_1) \vee \ldots \vee (=B_n \wedge \geq v_n)) \wedge =v^k\}$

Properties.

Generic: Eventually there will be an answer, either "*yes*" or "*no*"

Invariant: If B_i does not appear (B_i appears) in a compartment after a membrane division stage then v_i will stay there in all the subsequent steps (then v_i will disappear since the next step).

2.2. Synchronisation problem

Application: synchronisation problem

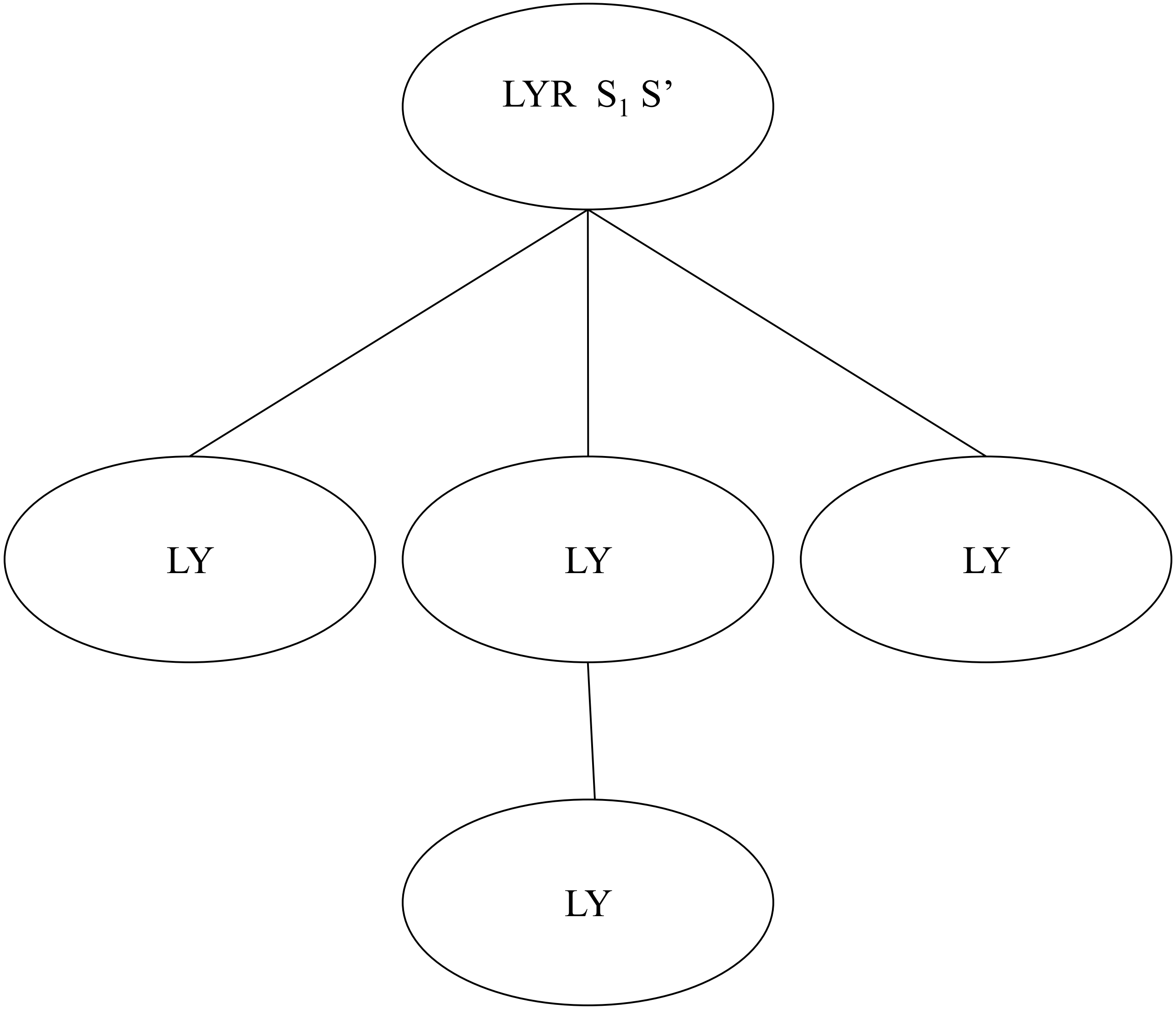
Synchronise the compartments of a membrane system (initially defined for one dimensional cellular automaton)

Problem: membrane system with *n* compartments (tree structure), and given initial states; provide a solution to the problem of finding a state of the system whereby every region will contain the same multiset and this is obtained for the first time

Solution idea: a nondeterministic solution consisting of

a. a signal travels down on the longest path
b. returns back by counting the number of regions
c. goes down by spreading the same number of objects at each level and decreasing by one when a new level is entered
d. finally the synchronisation state is reached

Bernardini, Gheorghe, Margenstern, Verlan: IJFCS, 2008 – can be rewritten as kP system model



$S_1 \to (S_1, in)$

$S' \to (S_2, here)$

$S_2 \to (S_2, in)$ ____________

$S_1\ S' \to (S_3 a, out)$

$S_3 \to (S_3 a, out)$

$a \to (a, out)$

$S_3 R \to (S_5 b, here)$

$a \to (b, here)$____________

$LYS_5 b \to (S_5, in^*)(S_6, here)$

$LYS_5 b \to (S_7, here)$

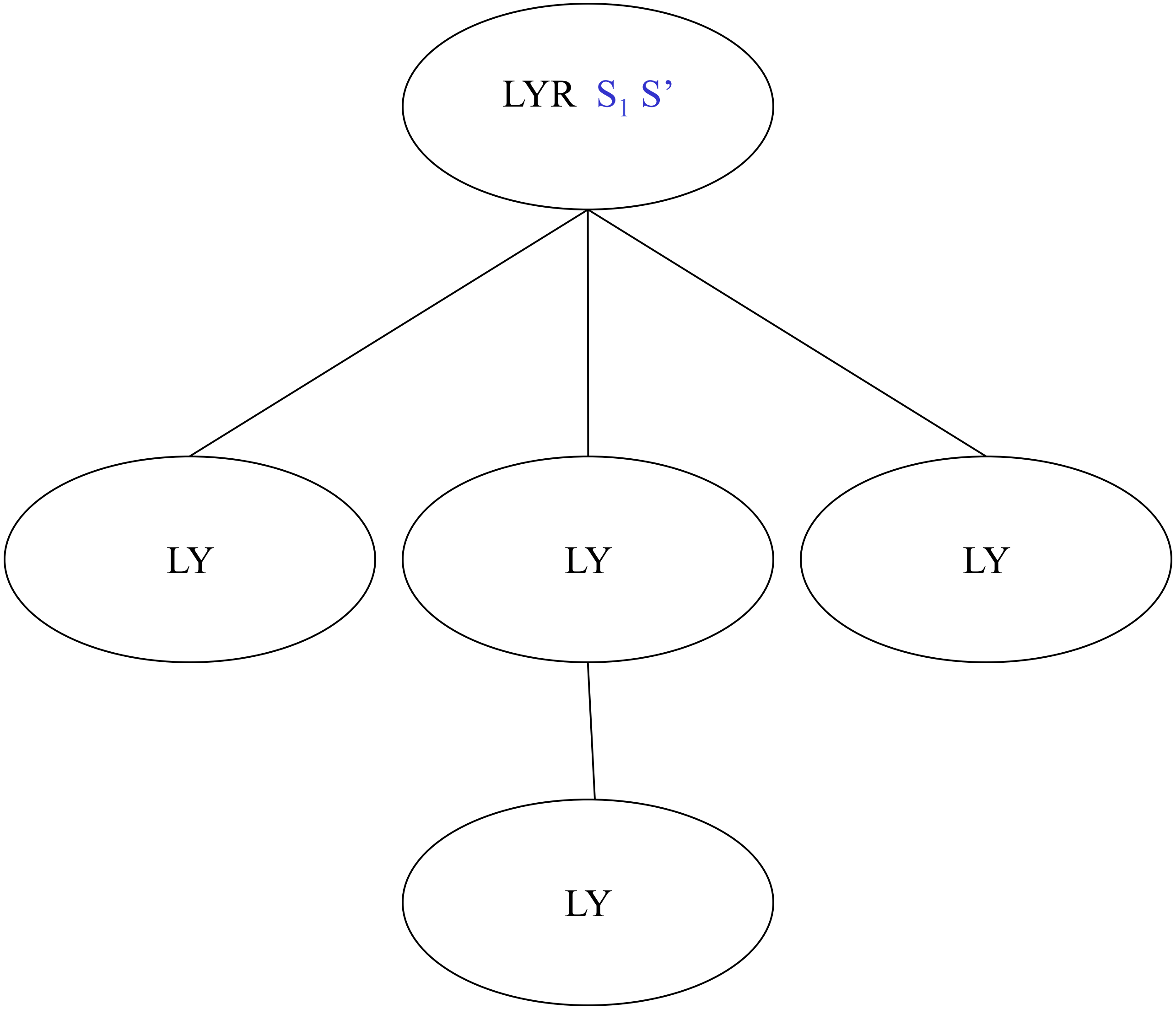
$b \to (b, in^*)(b', here)$_____

$S_6 b' \to (S_6, here)$

$S_7 b \to (S_7, here)$__________

$S_6 \to F$

$S_7 \to F$



$S_1 \rightarrow (S_1, in)$

$S' \rightarrow (S_2, here)$

$S_2 \rightarrow (S_2, in)$

$S_1\ S' \rightarrow (S_3 a, out)$

$S_3 \rightarrow (S_3 a, out)$

$a \rightarrow (a, out)$

$S_3 R \rightarrow (S_5 b, here)$

$a \rightarrow (b, here)$

$LYS_5 b \rightarrow (S_5, in^*)(S_6, here)$

$LYS_5 b \rightarrow (S_7, here)$

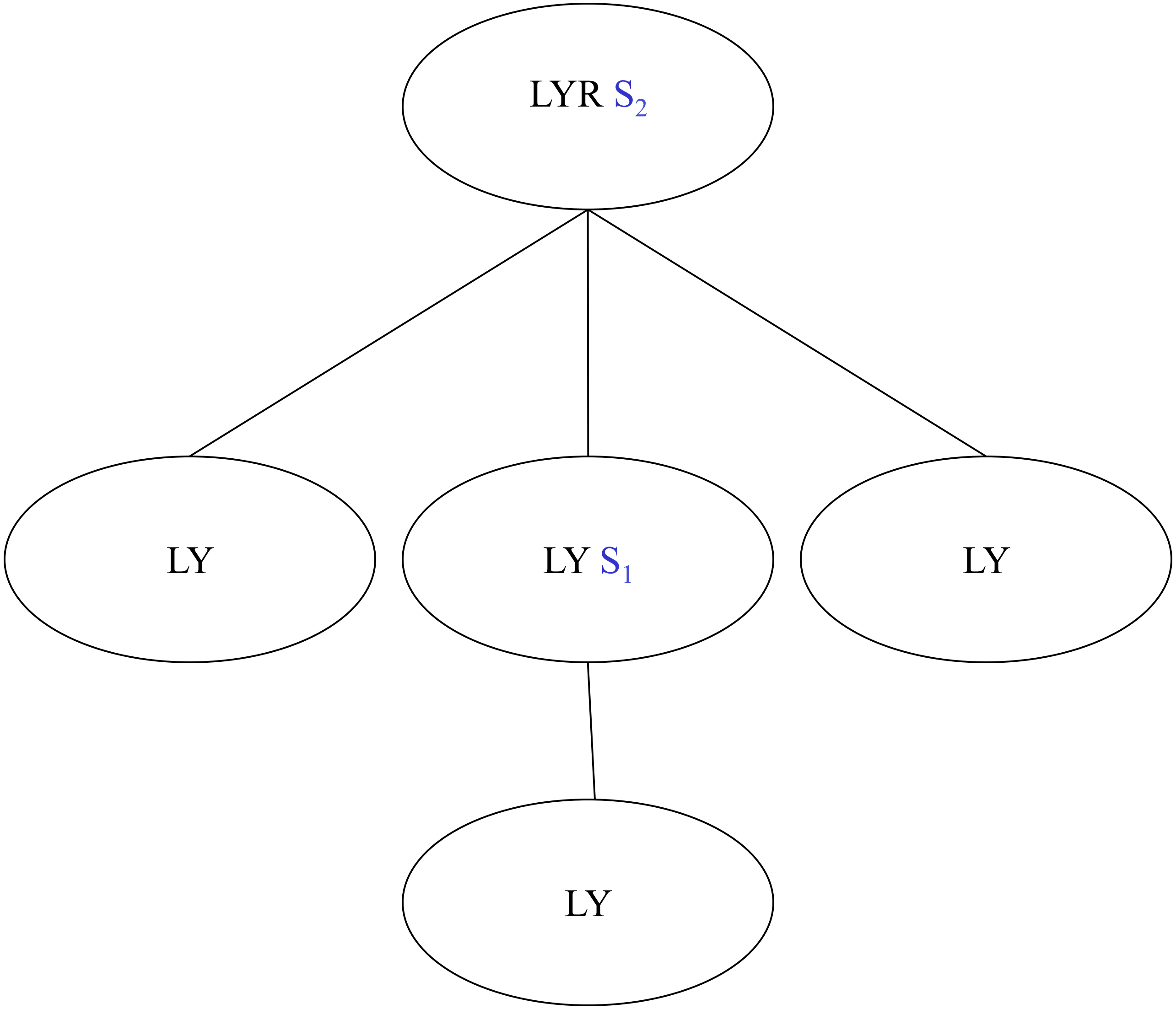
$b \rightarrow (b, in^*)(b', here)$

$S_6 b' \rightarrow (S_6, here)$

$S_7 b \rightarrow (S_7, here)$

$S_6 \rightarrow F$

$S_7 \rightarrow F$



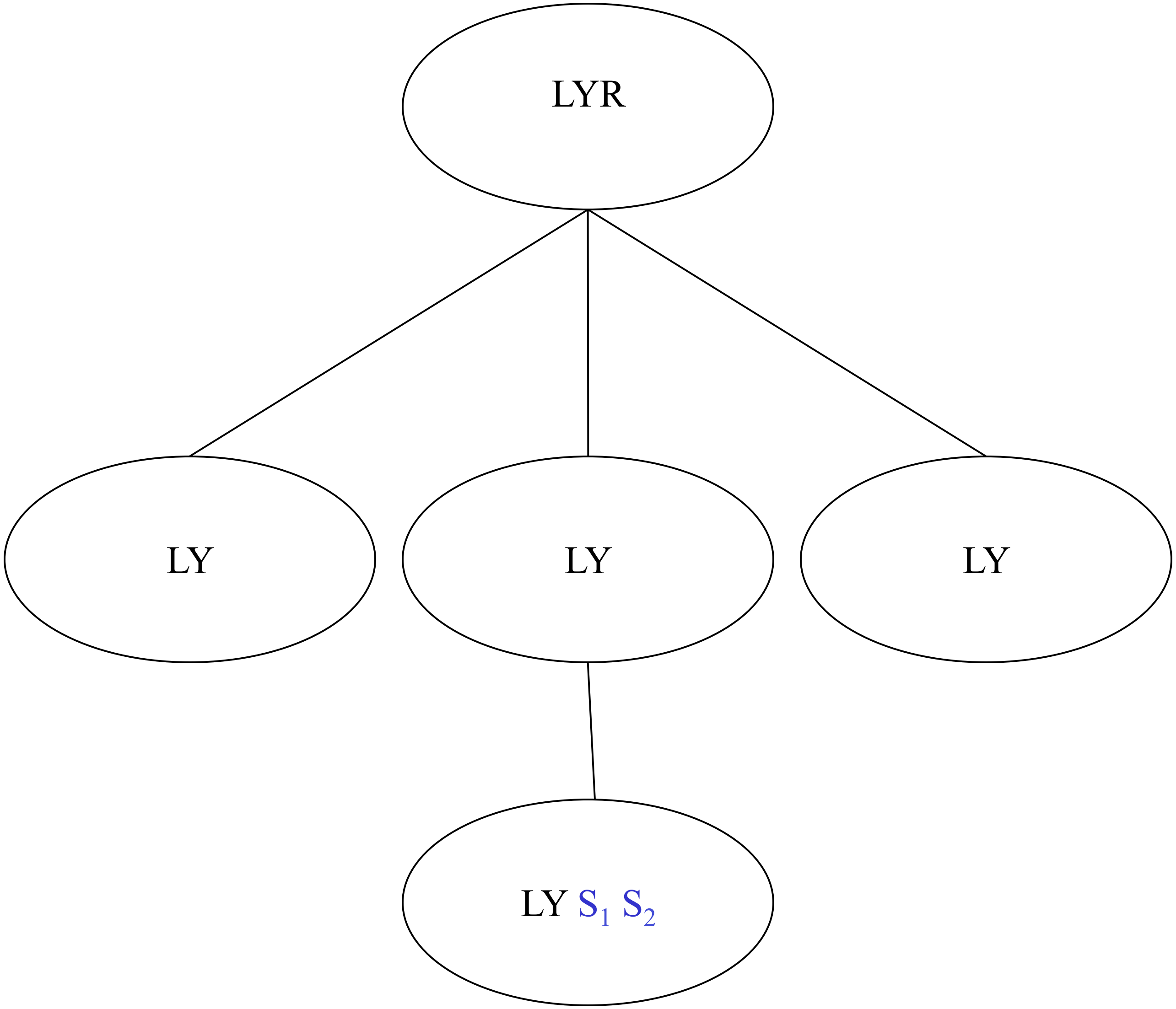
$S_1 \rightarrow (S_1,in)$
$S' \rightarrow (S_2,here)$
$S_2 \rightarrow (S_2,in)$ ____________

$S_1\ S' \rightarrow (S_3a,out)$
$S_3 \rightarrow (S_3a,out)$
$a \rightarrow (a,out)$
$S_3R \rightarrow (S_5b,here)$
$a \rightarrow (b,here)$____________

$LYS_5b \rightarrow (S_5,in^*)(S_6,here)$
$LYS_5b \rightarrow (S_7,here)$
$b \rightarrow (b,in^*)(b',here)$_____

$S_6b' \rightarrow (S_6,here)$
$S_7b \rightarrow (S_7,here)$__________

$S_6 \rightarrow F$
$S_7 \rightarrow F$



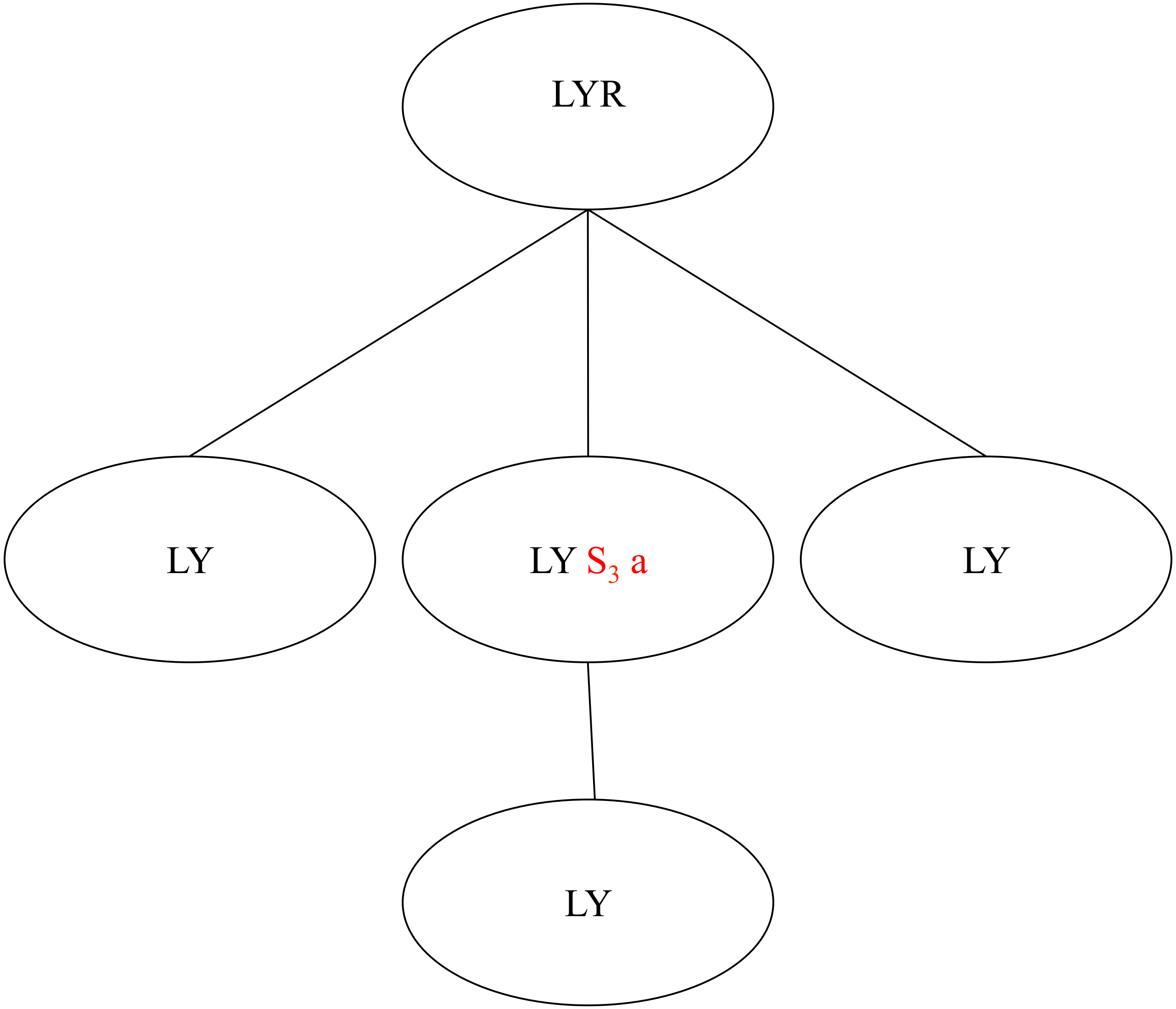
$S_1 \rightarrow (S_1,\text{in})$
$S' \rightarrow (S_2,\text{here})$
$S_2 \rightarrow (S_2,\text{in})$ ___________

$S_1\ S_2 \rightarrow (S_3a,\text{out})$
$S_3 \rightarrow (S_3a,\text{out})$
$a \rightarrow (a,\text{out})$
$S_3R \rightarrow (S_5b,\text{here})$
$a \rightarrow (b,\text{here})$___________

$LYS_5b \rightarrow (S_5,\text{in*})(S_6,\text{here})$
$LYS_5b \rightarrow (S_7,\text{here})$
$b \rightarrow (b,\text{in*})(b',\text{here})$____

$S_6b' \rightarrow (S_6,\text{here})$
$S_7b \rightarrow (S_7,\text{here})$_________

$S_6 \rightarrow F$
$S_7 \rightarrow F$



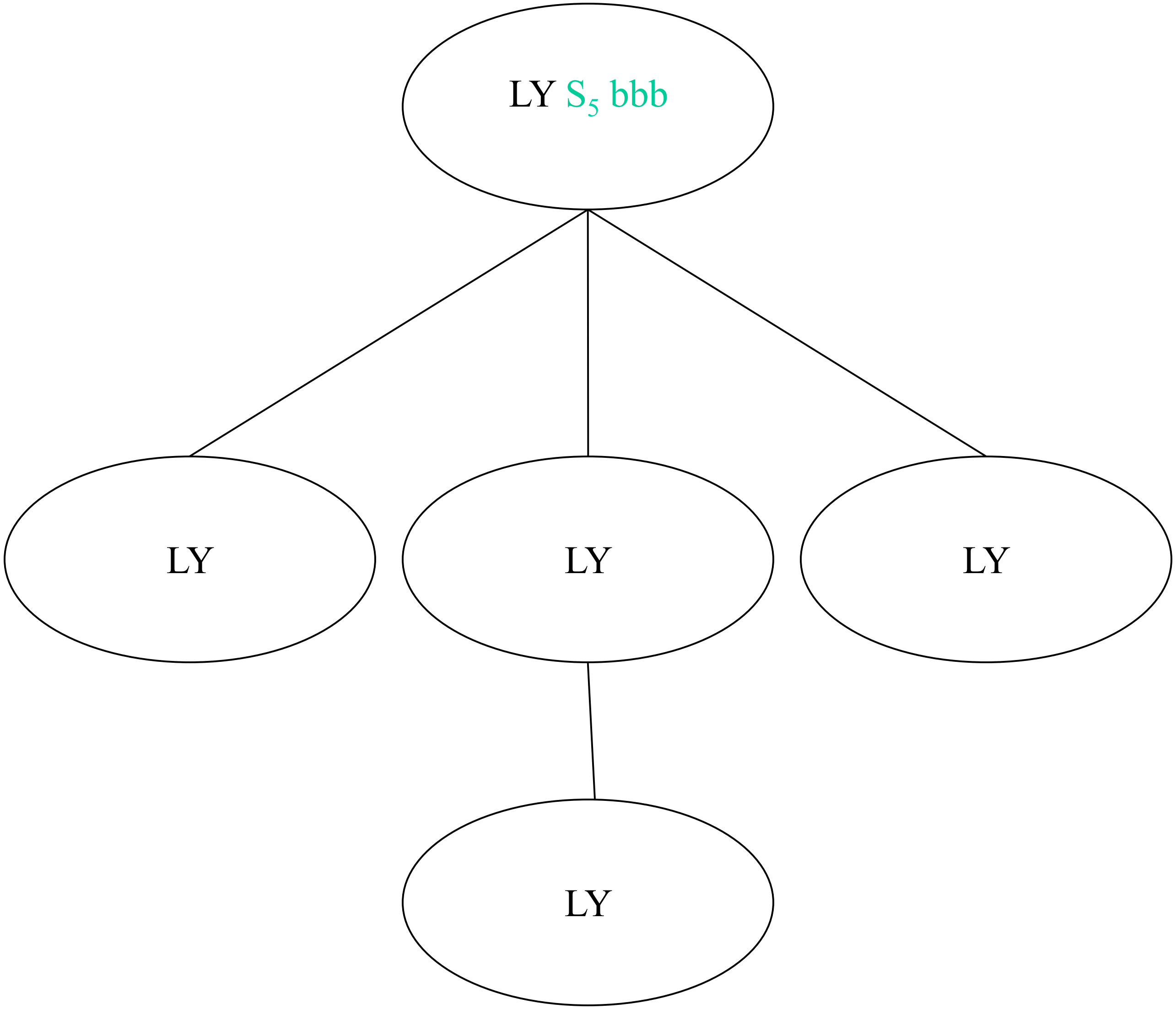
$S_1 \rightarrow (S_1,in)$
$S' \rightarrow (S_2,here)$
$S_2 \rightarrow (S_2,in)$ ___________

$S_1\ S_2 \rightarrow (S_3 a,out)$
$S_3 \rightarrow (S_3 a,out)$
$a \rightarrow (a,out)$
$S_3 R \rightarrow (S_5 b,here)$
$a \rightarrow (b,here)$___________

$LYS_5 b \rightarrow (S_5,in^*)(S_6,here)$
$LYS_5 b \rightarrow (S_7,here)$
$b \rightarrow (b,in^*)(b',here)$____

$S_6 b' \rightarrow (S_6,here)$
$S_7 b \rightarrow (S_7,here)$_________

$S_6 \rightarrow F$
$S_7 \rightarrow F$



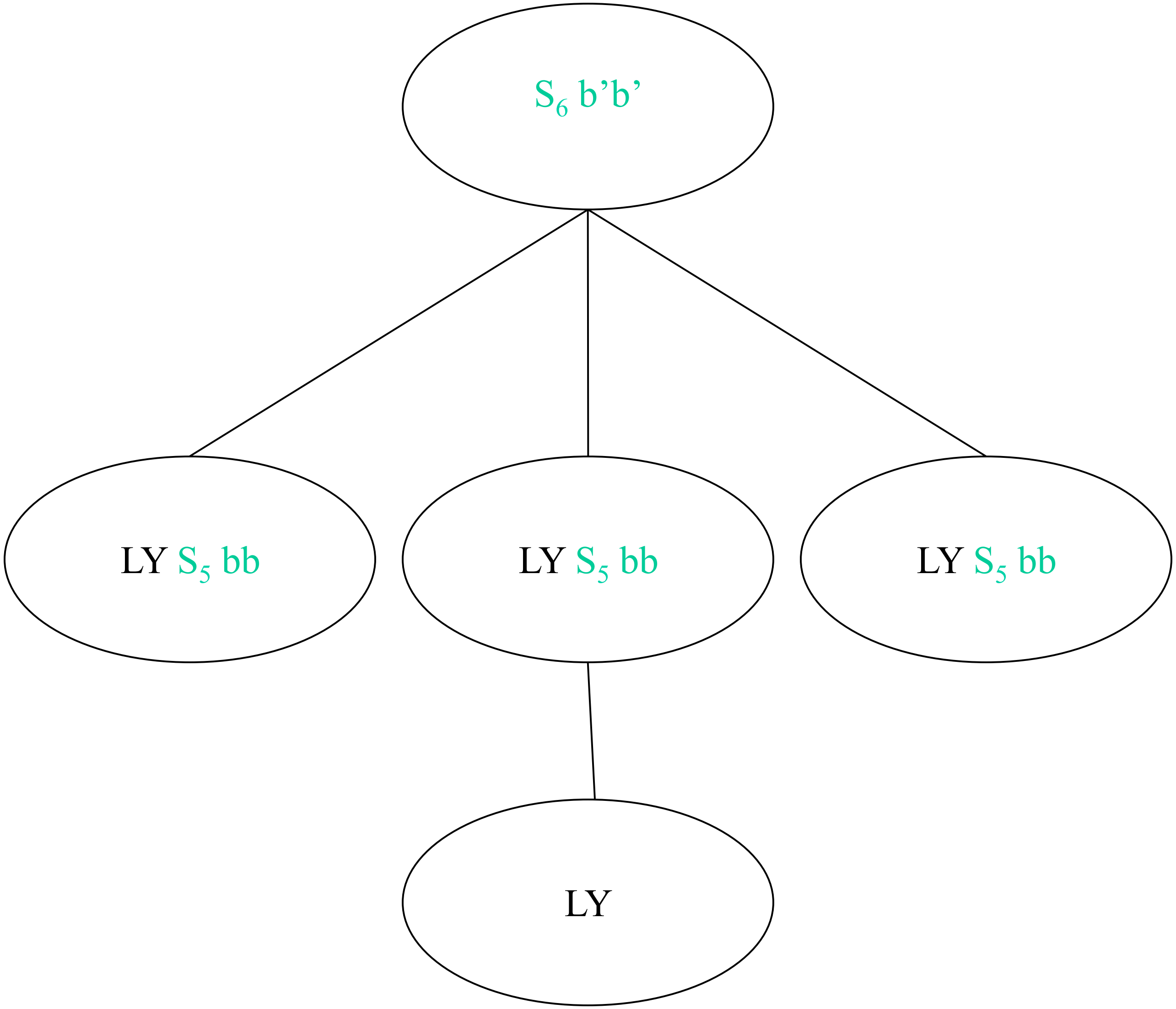
$S_1 \rightarrow (S_1,in)$
$S' \rightarrow (S_2,here)$
$S_2 \rightarrow (S_2,in)$ ___________

$S_1\ S_2 \rightarrow (S_3a,out)$
$S_3 \rightarrow (S_3a,out)$
$a \rightarrow (a,out)$
$S_3R \rightarrow (S_5b,here)$
$a \rightarrow (b,here)$___________

$LYS_5b \rightarrow (S_5,in^*)(S_6,here)$
$LYS_5b \rightarrow (S_7,here)$
$b \rightarrow (b,in^*)(b',here)$____

$S_6b' \rightarrow (S_6,here)$
$S_7b \rightarrow (S_7,here)$_________

$S_6 \rightarrow F$
$S_7 \rightarrow F$



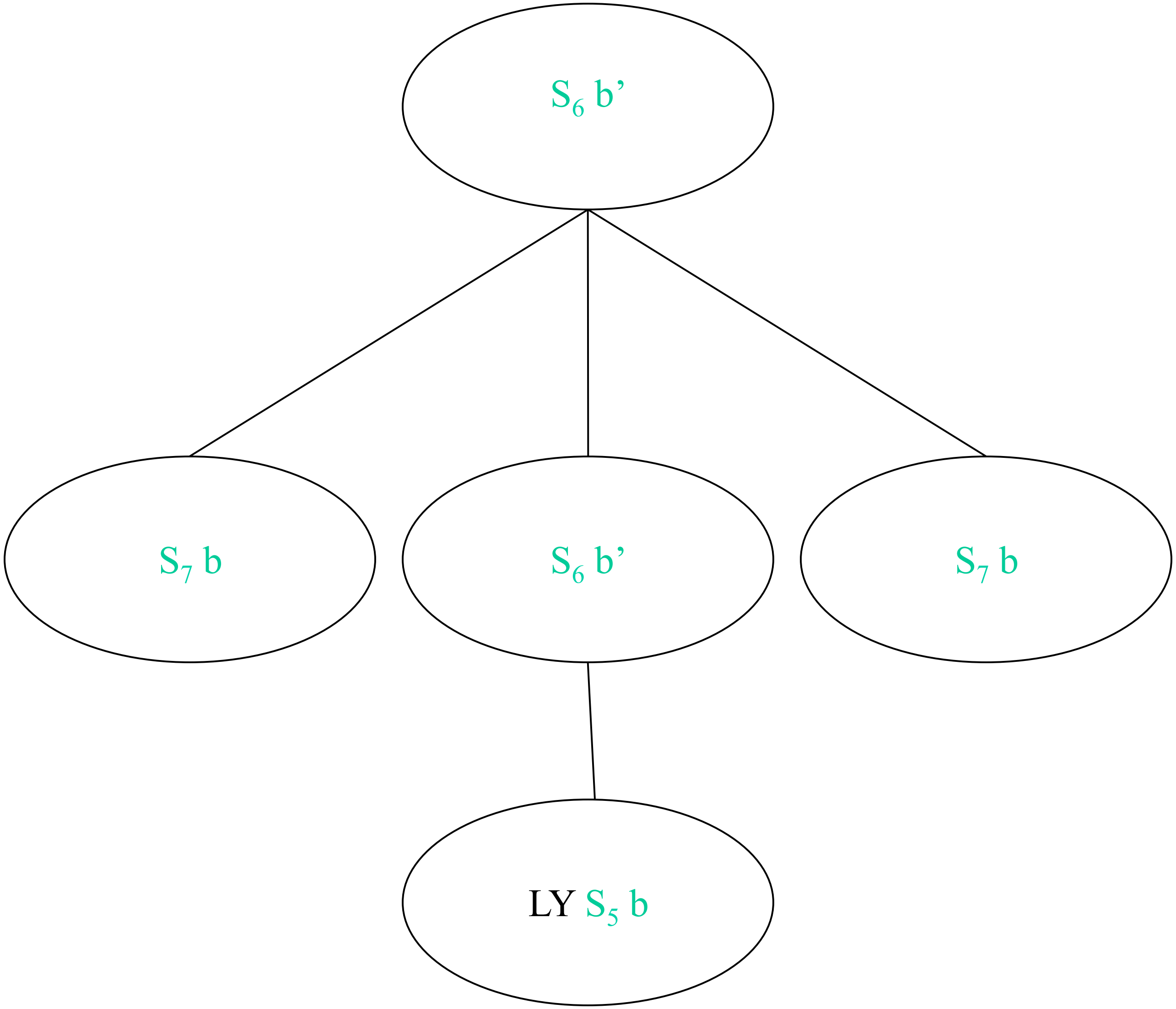
$S_1 \rightarrow (S_1,in)$
$S' \rightarrow (S_2,here)$
$S_2 \rightarrow (S_2,in)$

$S_1\ S_2 \rightarrow (S_3a,out)$
$S_3 \rightarrow (S_3a,out)$
$a \rightarrow (a,out)$
$S_3R \rightarrow (S_5b,here)$
$a \rightarrow (b,here)$

$LYS_5b \rightarrow (S_5,in^*)(S_6,here)$
$LYS_5b \rightarrow (S_7,here)$
$b \rightarrow (b,in^*)(b',here)$

$S_6b' \rightarrow (S_6,here)$
$S_7b \rightarrow (S_7,here)$

$S_6 \rightarrow F$
$S_7 \rightarrow F$



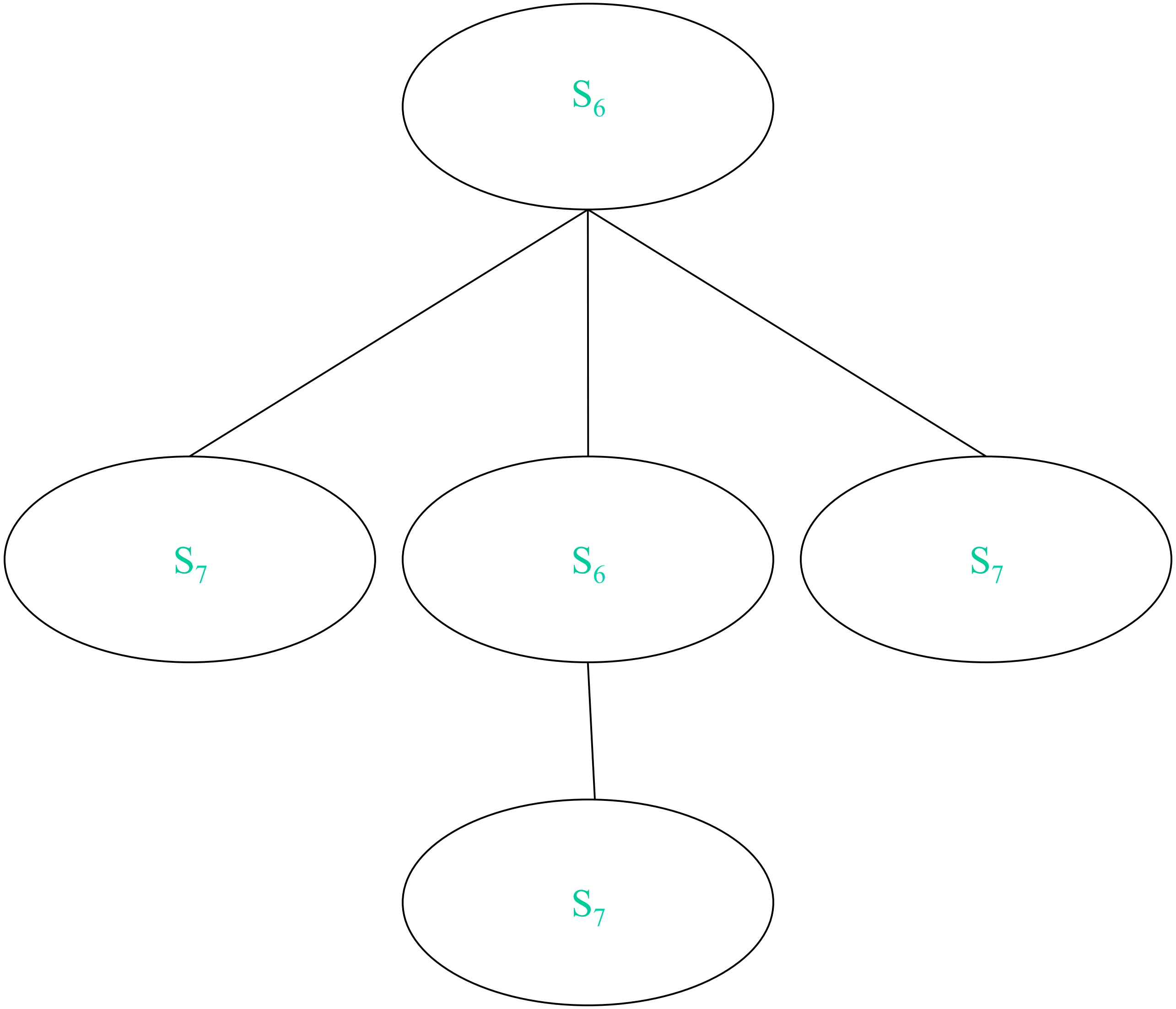
$S_1 \rightarrow (S_1,in)$
$S' \rightarrow (S_2,here)$
$S_2 \rightarrow (S_2,in)$

$S_1\ S_2 \rightarrow (S_3a,out)$
$S_3 \rightarrow (S_3a,out)$
$a \rightarrow (a,out)$
$S_3R \rightarrow (S_5b,here)$
$a \rightarrow (b,here)$

$LYS_5b \rightarrow (S_5,in^*)(S_6,here)$
$LYS_5b \rightarrow (S_7,here)$
$b \rightarrow (b,in^*)(b',here)$

$S_6b' \rightarrow (S_6,here)$
$S_7b \rightarrow (S_7,here)$

$S_6 \rightarrow F$
$S_7 \rightarrow F$



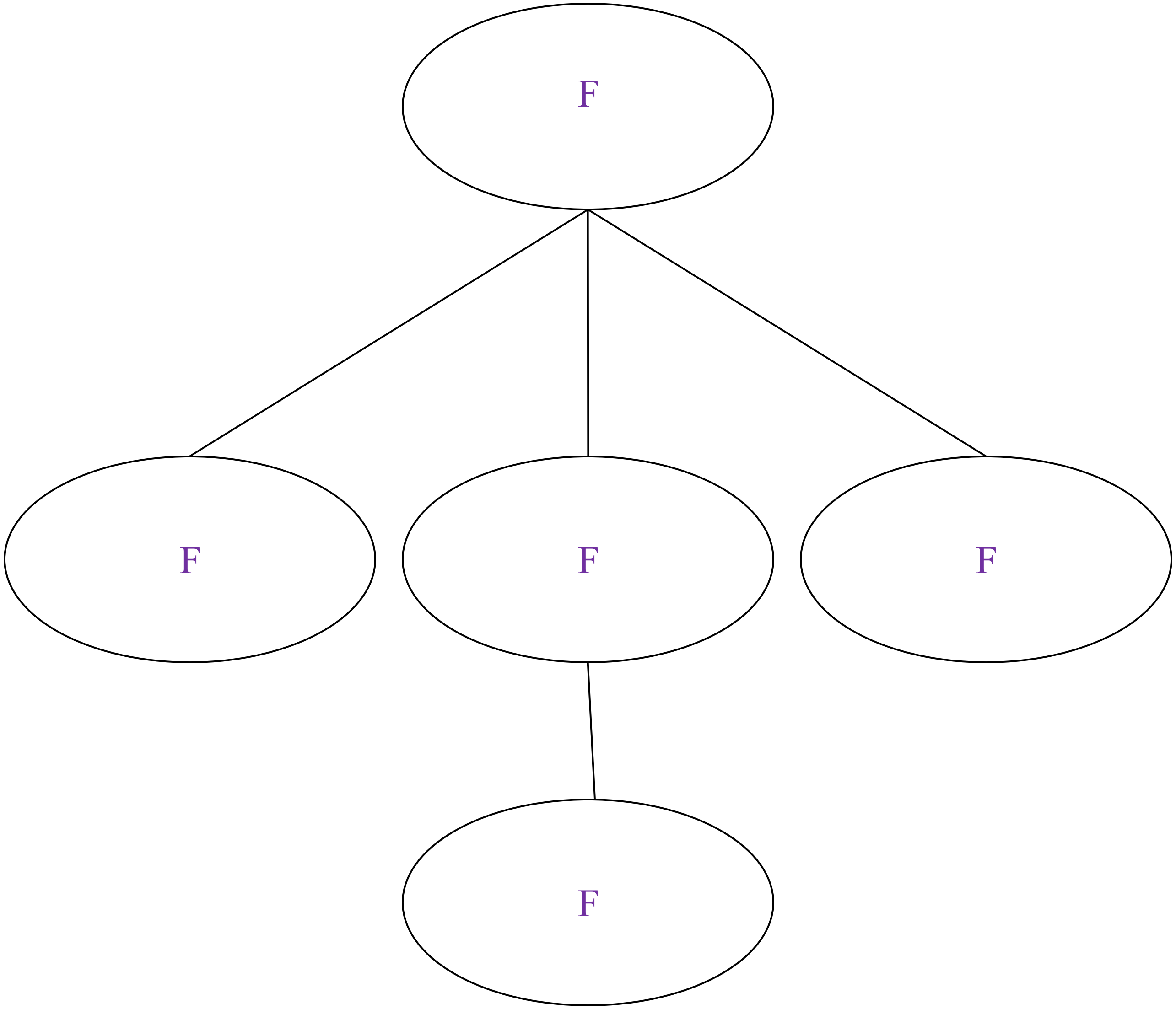
$S_1 \rightarrow (S_1,\text{in})$
$S' \rightarrow (S_2,\text{here})$
$S_2 \rightarrow (S_2,\text{in})$

$S_1\ S_2 \rightarrow (S_3a,\text{out})$
$S_3 \rightarrow (S_3a,\text{out})$
$a \rightarrow (a,\text{out})$
$S_3R \rightarrow (S_5b,\text{here})$
$a \rightarrow (b,\text{here})$

$LYS_5b \rightarrow (S_5,\text{in*})(S_6,\text{here})$
$LYS_5b \rightarrow (S_7,\text{here})$
$b \rightarrow (b,\text{in*})(b',\text{here})$

$S_6b' \rightarrow (S_6,\text{here})$
$S_7b \rightarrow (S_7,\text{here})$

$S_6 \rightarrow F$
$S_7 \rightarrow F$



$S_1 \rightarrow (S_1, in)$
$S' \rightarrow (S_2, here)$
$S_2 \rightarrow (S_2, in)$

$S_1\ S_2 \rightarrow (S_3 a, out)$
$S_3 \rightarrow (S_3 a, out)$
$a \rightarrow (a, out)$
$S_3 R \rightarrow (S_5 b, here)$
$a \rightarrow (b, here)$

$LYS_5 b \rightarrow (S_5, in^*)(S_6, her)$
$LYS_5 b \rightarrow (S_7, here)$
$b \rightarrow (b, in^*)(b', here)$

$S_6 b' \rightarrow (S_6, here)$
$S_7 b \rightarrow (S_7, here)$

$S_6 \rightarrow F$
$S_7 \rightarrow F$

Kernel P systems – Observations

- A **generic modelling** framework

- **Direct mappings** of neural-like P systems, P systems with active membranes and electrical charges, generalised communicating P systems

- Two **specification languages** – P-lingua oriented and a special syntax (kP-lingua) with Spin mapping

- **Formal verification** and basis for testing

- **Natural language queries** (CMC2013: paper and poster)

Joint project with F Ipate (Bucharest) with 2 PhD students; collaborators: M Pérez-Jiménez group (Sevilla)

3. Probabilistic P systems

Probabilistic P systems

- A **generic framework** consisting of compartments
- **Rules with probabilities**
- A **specific execution strategy** – based on Gillespie
- **Similar to** stochastic process algebras and stochastic Petri nets

Romero-Campero, Gheorghe, Krasnogor: IJFCS, 2009

Gene regulatory network: P system model

Alphabet : *gene, rna, protein, act, rep, act-gene, rep-gene*

Compartment: *b*. Initial values: *gene*, act^{10}, rep^{10}

Rules

r_1: $[gene]_b \rightarrow [gene + rna]_b$, $c_1=0.347\ min^{-1}$

r_2: $[rna]_b \rightarrow [rna + protein]_b$, $c_2=0.174\ min^{-1}$

r_3: $[rna]_b \rightarrow [\,]_b$, $c_3=0.347\ min^{-1}$

r_4: $[protein]_b \rightarrow [\,]_b$, $c_4=0.0116\ min^{-1}$

r_5: $[act + gene]_b \rightarrow [act\text{-}gene]_b$, $c_5=6.641\ min^{-1}$

r_6: $[act\text{-}gene]_b \rightarrow [act + gene]_b$, $c_6=0.6\ min^{-1}$

r_7: $[act\text{-}gene]_b \rightarrow [act\text{-}gene + rna]_b$, $c_7=3.47\ min^{-1}$

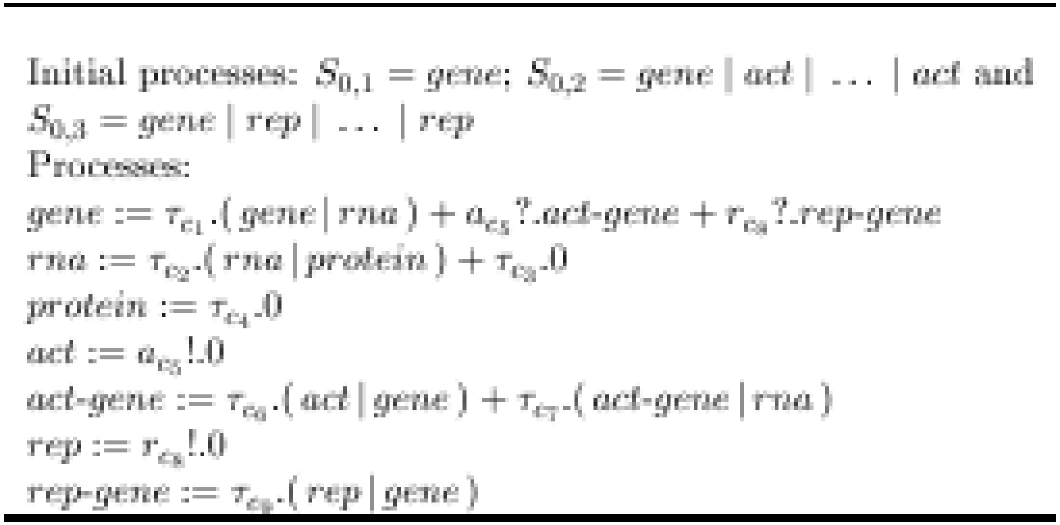
r_8: $[rep + gene]_b \rightarrow [rep\text{-}gene]_b$, $c_8=6.641\ min^{-1}$

r_9: $[rep\text{-}gene]_b \rightarrow [rep + gene]_b$, $c_9=0.6\ min^{-1}$

Lac operon in E coli:

WS Hlavacek, MA Savageau; J Molecular Biology, 1995

Stochastic Pi-calculus and Petri Nets



Initial processes: $S_{0,1} = gene$; $S_{0,2} = gene \mid act \mid \ldots \mid act$ and $S_{0,3} = gene \mid rep \mid \ldots \mid rep$

Processes:

$gene := \tau_{c_1}.(\, gene \mid rna \,) + a_{c_5}?.act\text{-}gene + r_{c_8}?.rep\text{-}gene$

$rna := \tau_{c_2}.(\, rna \mid protein \,) + \tau_{c_3}.0$

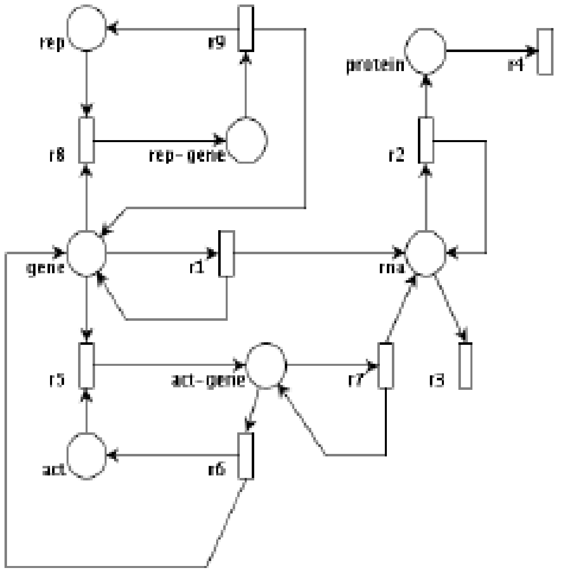
$protein := \tau_{c_4}.0$

$act := a_{c_5}!.0$

$act\text{-}gene := \tau_{c_6}.(\, act \mid gene \,) + \tau_{c_7}.(\, act\text{-}gene \mid rna \,)$

$rep := r_{c_8}!.0$

$rep\text{-}gene := \tau_{c_9}.(\, rep \mid gene \,)$



3.1 Domain specific language based on probabilistic P systems

DSL based on P systems - LPP

- A **framework** consisting of compartments, defined out of **modules** (sets of rules)

- Compartments distributed across a **lattice** with communication mechanisms

- All included in IBW together with a verification component and optimisation module – Blakes et al, Bioinformatics, 2011

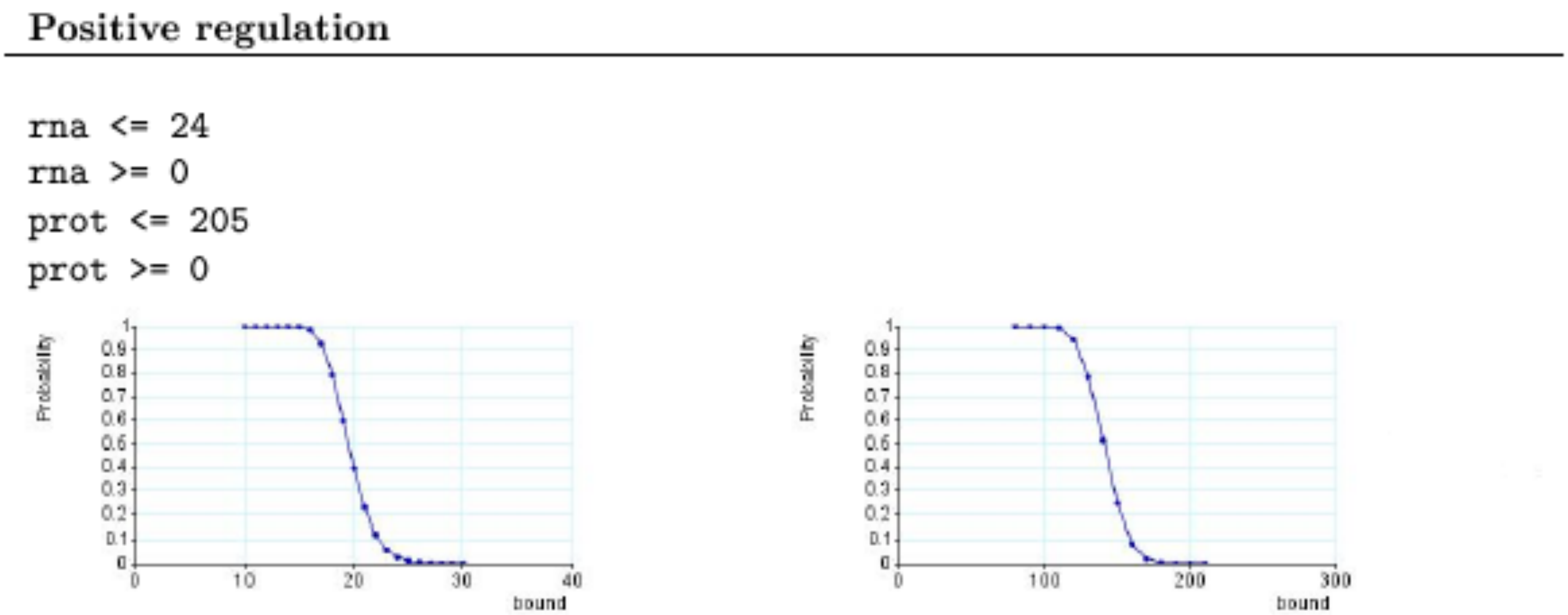
http://www.infobiotic.org/
Blakes et al: Infobiotics Workbench (42pp); book chapter in Applications of P Systems in Biology, 2013, Springer

Extracting properties - Daikon

- From experiments or simulations – data series obtained
- Extract various properties, including invariants - Daikon
- For the gene regulatory network, positive regulation then

$$0 \leq rna \leq 24$$
$$0 \leq protein \leq 205$$
$$\&$$
$$rna,\ protein > rep$$

Invariants checking – Positive regulation



Positive regulation

```
rna <= 24
rna >= 0
prot <= 205
prot >= 0
```

… more likely **rna**’s between 0 and 15, **prot**eins between 0 and 150

Prism model checker

3.2 Current DSL – Main features

Current work on DSL platform

- DSL for synthetic biology
    - Mechanism to specify **interactions/reactions** (GEC – MS research) - **processes**
    - A way to specify properties of the genetic material (Eugene – California University) defining **devices**
    - DSL: a hierarchy of devices with processes in **systems**, **cells**, **colonies** etc

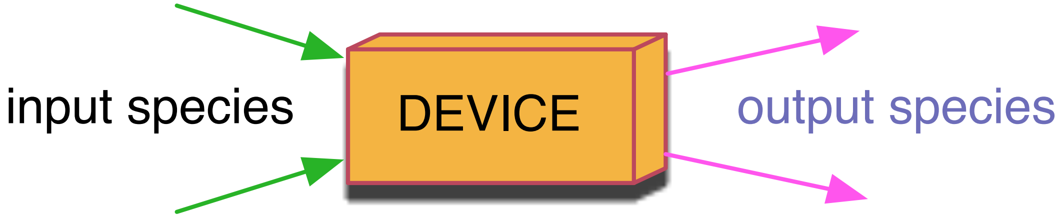
- Stochastic model checker
    - Verification (Prism, MC2,…)
    - Use of natural language patterns
    - Integration with the DSL

- Automatic definition of parts and devices –**atgc** compiler

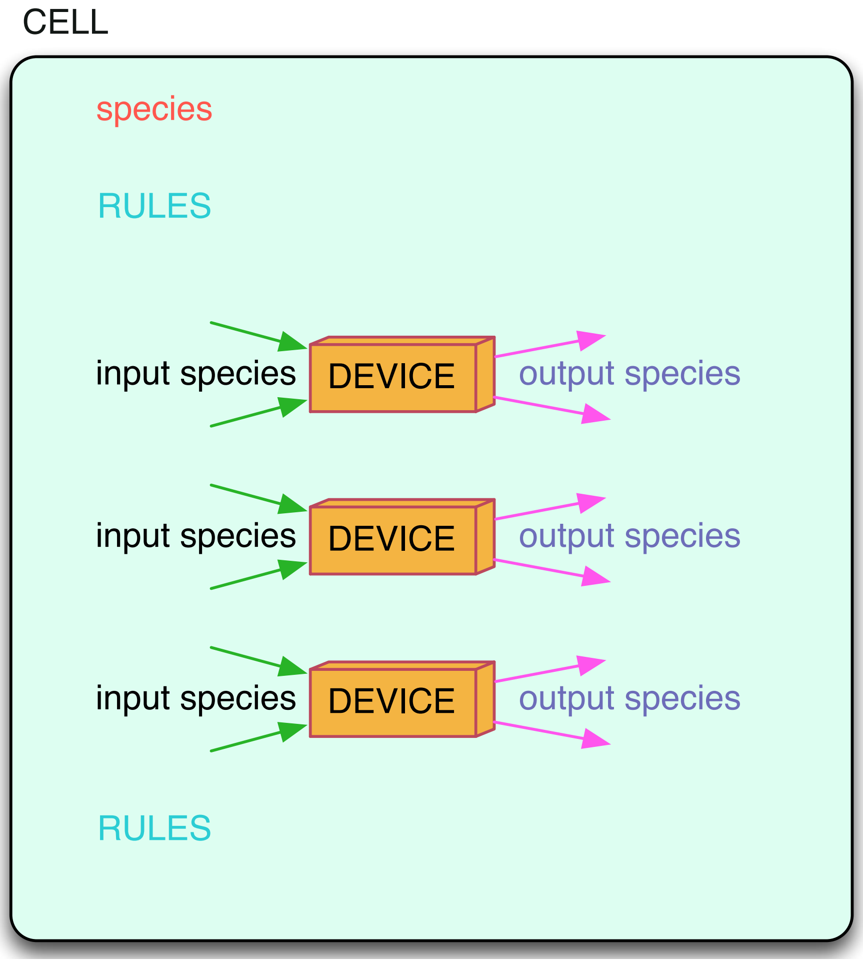
Natural language: Property patterns

Existence	::=	The concentration of S becomes greater than the concentration of F [Barbuti et. al., 2005].
Absence	::=	It is not possible to activate X in any pathways [Donaldson and Calder, 2012].
Universality	::=	GTP level is always less than k [Antoniotti et. al., 2003].
Until	::=	The protein A degrades before binding to the protein B [Heath et. al., 2006].
Response	::=	If reaction R is possible, then eventually the reaction R happens [Eker et. al., 2002].
Steady-State	::=	In the long run there are precisely n $MAPK$s activated [Kwiatkowska et. al., 2008].
Oscillation	::=	Oscillation terminates in species $X \in \{A, B, C\}$ [Ballarini et. al., 2009].
Reward	::=	Expected time to reach a state in which all gates have finished executing. [Lakin et. al., 2012].

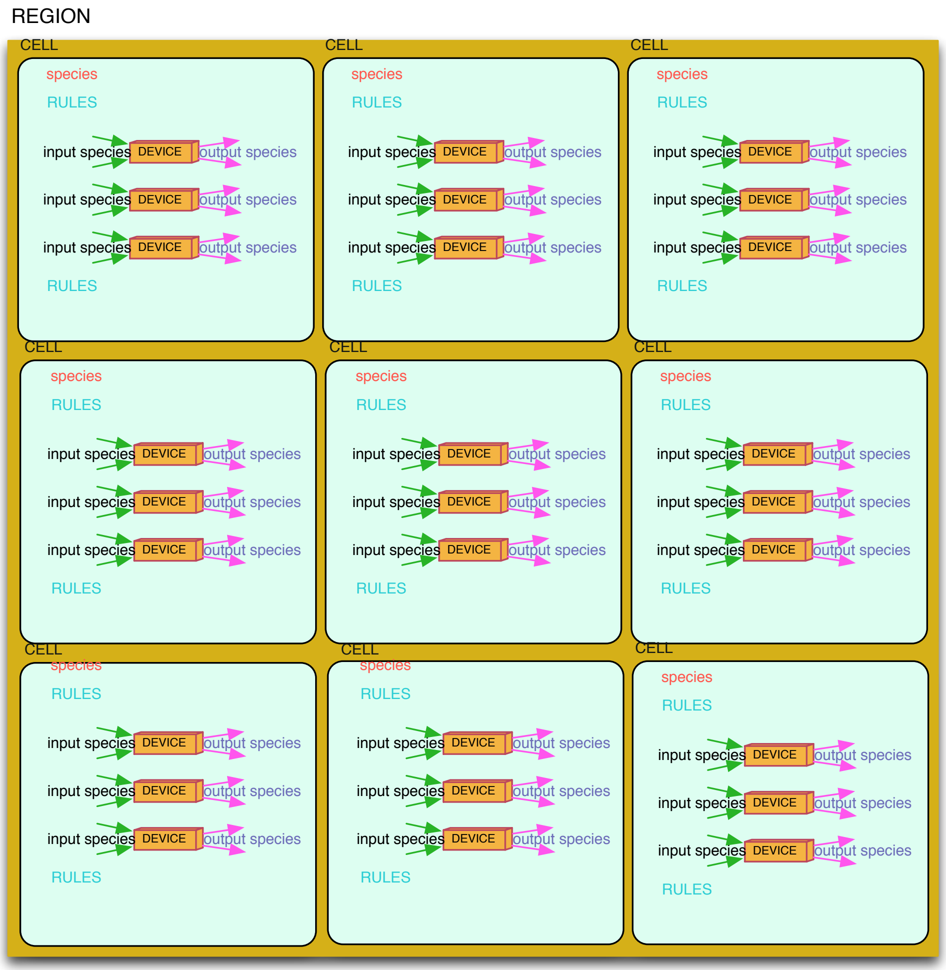
Property patterns for the DSL (1)



Property patterns for the DSL (2)



Property patterns for the DSL (3)



REGION

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

CELL

species

RULES

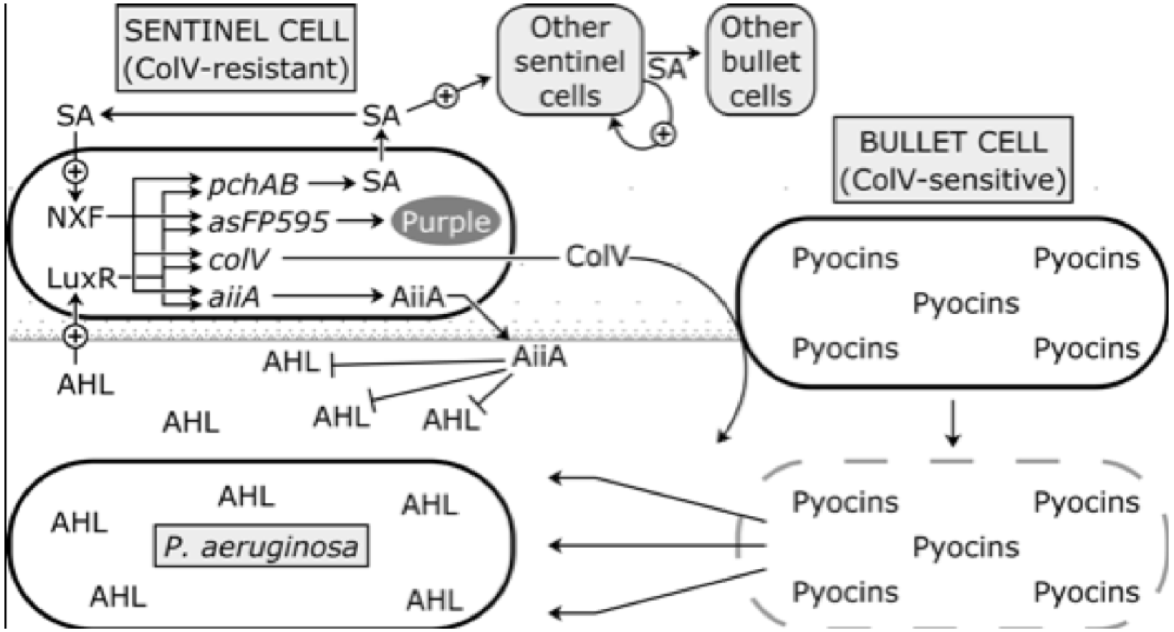
input species DEVICE output species

input species DEVICE output species

input species DEVICE output species

RULES

Case study



SENTINEL CELL
(ColV-resistant)
SA
SA
Other sentinel cells
SA
Other bullet cells
NXF
LuxR
pchAB
SA
asFP595
Purple
colV
ColV
aiiA
AiiA
AiiA
AHL
AHL
AHL
AHL
AHL
BULLET CELL
(ColV-sensitive)
Pyocins
Pyocins
Pyocins
Pyocins
Pyocins
AHL
AHL
AHL
P. aeruginosa
AHL
AHL
Pyocins
Pyocins
Pyocins
Pyocins
Pyocins

Property patterns at the Region level

Region-level Queries:

- VERIFY [SA IN SENTINEL > 0 uM] WILL EVENTUALLY HOLD
- VERIFY [SA IN E_COLI > 0 uM] WILL EVENTUALLY HOLD
- VERIFY [SA IN ALL CELLS > 0 uM] WILL EVENTUALLY HOLD

- VERIFY [AHL IN PSEUDOMONAS > 0 uM] IS FOLLOWED BY
  [SA IN SENTINEL NEIGHBOUR OF PSEUDOMONAS > 0 uM]
- VERIFY [SA IN SENTINEL > 0 uM] IS FOLLOWED BY
  [SA IN NEIGHBOUR OF SENTINEL > 0 uM]
- VERIFY [SA IN SENTINEL > 0 uM] IS FOLLOWED BY
  [COLV IN SENTINEL > 0 uM]
- VERIFY [COLV IN SENTINEL > 0 uM] IS FOLLOWED BY
  [PYOSINS IN BULLET > 0 uM]
- VERIFY [PYOSINS IN SENTINEL > 0 uM] IS FOLLOWED BY
  [AHL IN PSEUDOMONAS = 0 uM]

- VERIFY [AHL IN PSEUDOMONAS > 0 uM] IS FOLLOWED BY
  [AHL IN PSEUDOMONAS = 0 uM]

Conclusions

- Membrane computing as a nature-inspired computing paradigm
- Kernel P systems and probabilistic P systems – bases for DSL's
- Each DSL comes with its own set of verification tools
- Some are still "work in progress"

(PhD) Research proposals

- Kernel P systems:
  - Efficient algorithms to translate various classes of P systems (especially those with good examples) into kP systems
  - Good tools for simulation, including parallel platforms
  - Improved Spin formal verification platform
- Synthetic biology DSL: probabilistic verification tools

(PhD) Research proposals

- Kernel P systems:

  - Efficient algorithms to translate various classes of P systems (especially those with good examples) into kP systems
  - Good tools for simulation, including parallel platforms
  - Improved Spin formal verification platform

- Synthetic biology DSL: probabilistic verification tools

INTERESTED: contact me at m.gheorghe@sheffield.ac.uk

Projects, Publications

- EPSRC grants: Infobiotics at Nottingham & Roadblock (Nottingham, Sheffield, Warwick)
- Romanian Research Council CNCS-UEFISCDI, PN- II-ID-PCE-2011-3-0688: MuVet

- http://ppage.psystems.eu/
- http://www.dcs.shef.ac.uk/~marian

- Application of MC in Biology, Springer (in press)

Acknowledgements

Many Thanks for the invitation!

MuVet, Roadblock, RAE UK – partially supporting the research