Stochastic π-Calculus Revisited

Luca Cardelli
Microsoft Research

with Radu Mardare
University of Aalborg

also featuring the Biological Computation Group
Microsoft Research

ICTAC 2013-09-05, Shanghai

The Eras of Programming Semantics

First Era: Natural language definitions

- Flexible but (often) inprecise (FORTRAN, LISP, Algol, ...)

Second Era: Denotational Semantics (+ logic/algebraic approaches)

- Precise but hard to adapt to some features (typically concurrency)

Third Era: Structural Operational Semantics

- *Free lunch!* Precise *and* flexible (ML, Java, CSP, π-Calculus, type systems, ...)

Fourth Era: Quantitative Semantics

- New families of languages being developed for
  - modeling of natural systems
  - compilation and execution of physical systems
  - (specification of engineering systems: not in this talk)
- Denotational/Mathematical/Logical approaches: deep, in progress
- Operational Approaches: fairly successful so far, but *no longer a free lunch*

Quantitative Languages include

Process algebras (and automata theories) for performance evaluation

- Initial inspiration and still growing application area
- The typical model here is discrete-time Markov chains (*transition probabilities*) and hybrid models with continuous time

Process algebras for Systems Biology

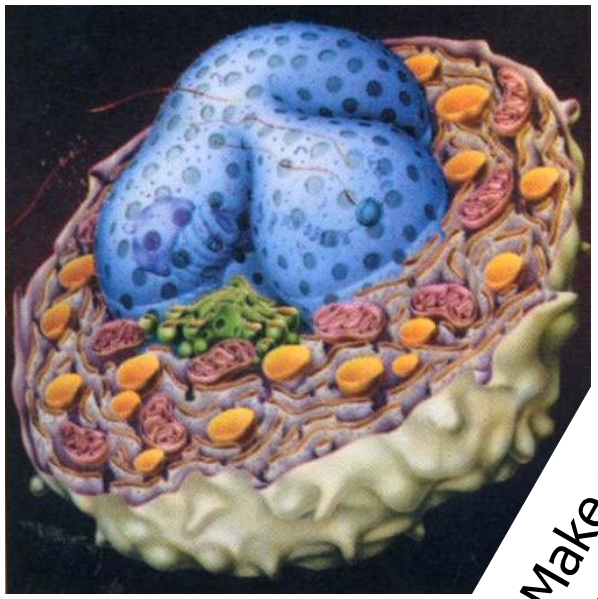
- Stochastic π-Calculus, (Bio)PEPA, ...
- The semantic model here is typically *continuous-time* Markov chains (transition *rates*): we want to know how fast the system runs. E.g. to relate to chemical kinetics

Process algebras for Synthetic Biology

- DNA Computing, gene assembly, ...
- Chemistry as an executable programming language

Applications in Systems Biology

Abstract Machines of Biochemistry



H.Lodish & al. Molecular Cell Biology 4th ed.

Regulation

Gene Machine
Nucleotides

Make proteins

Send signals

Direct construction

Confine genome and regulators

Hold receptors, host reactions

Enact fusion, fission

Protein Machine
Aminoacids

Membrane Machine
Phospholipids

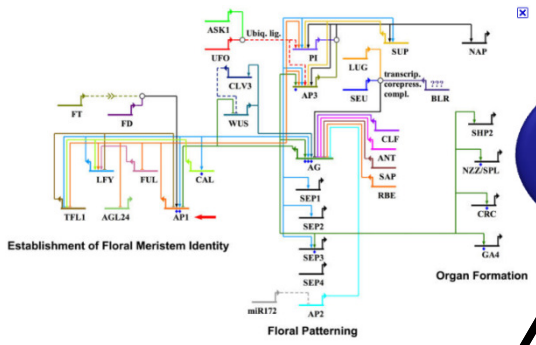
Metabolism, Propulsion
Signaling, Transport

Glycan Machine
Sugars

Surface and
Extracellular Features

Confinement, Storage
Bulk Transport

Systems Biology (Networks)

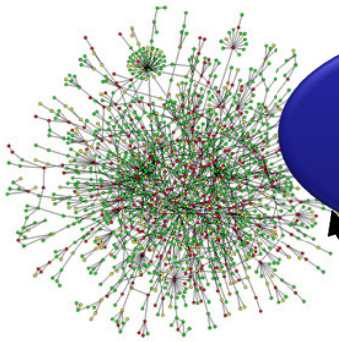


Gene Machine

Gene Regulatory Networks

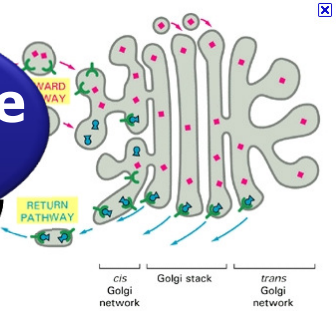
Biochemical Networks

Protein Machine

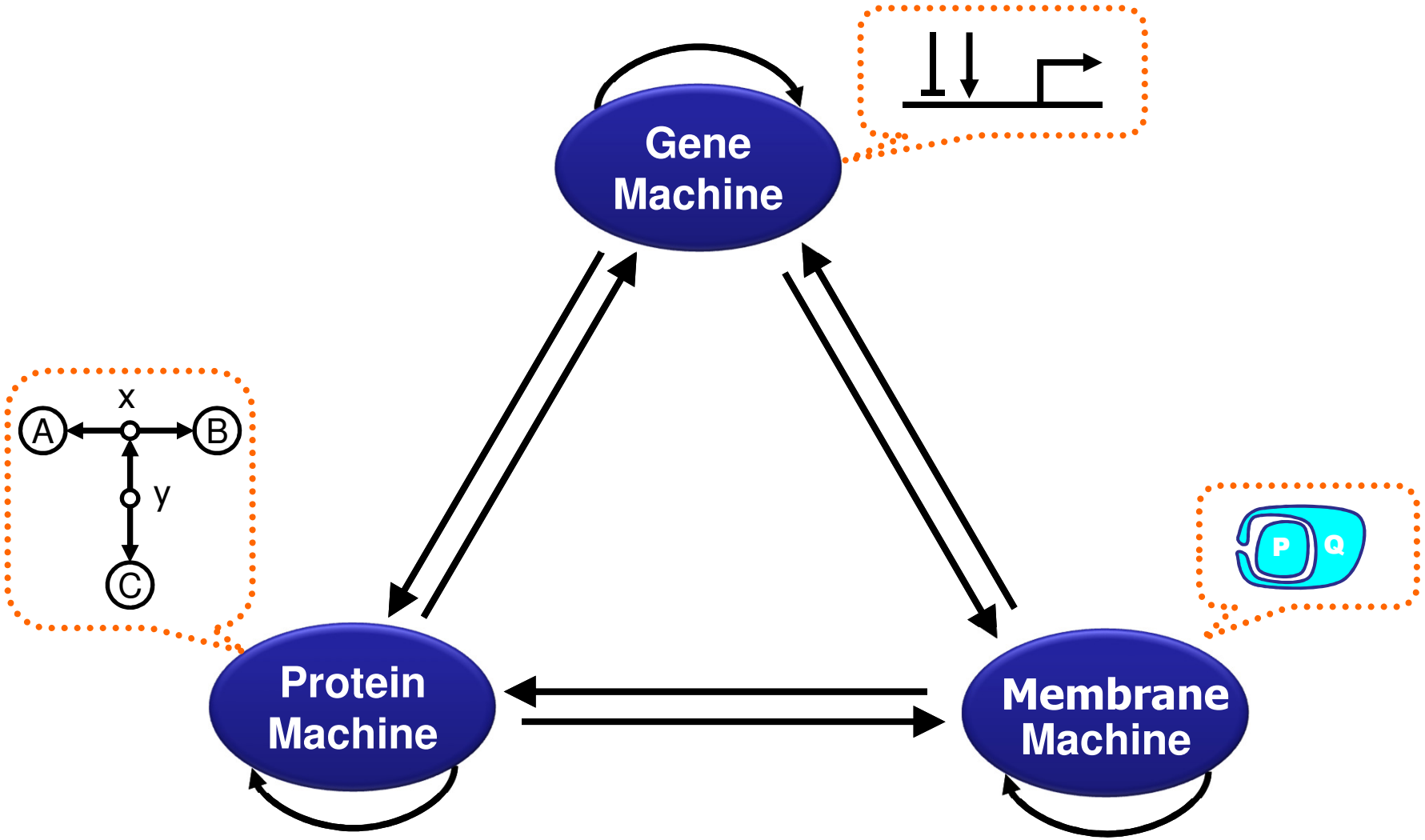


Transport Networks

Membrane Machine



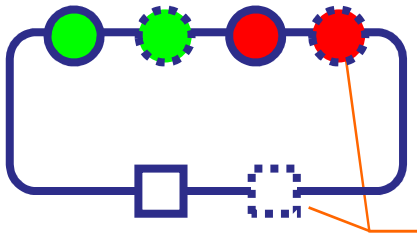
Biological Modeling (Languages)



The Informal Model of Protein Interaction

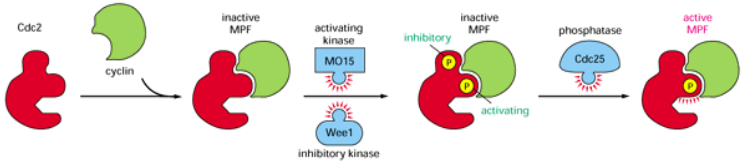
cf. BioCalculus [Kitano&Nagasaki], κ-calculus [Danos&Laneve]

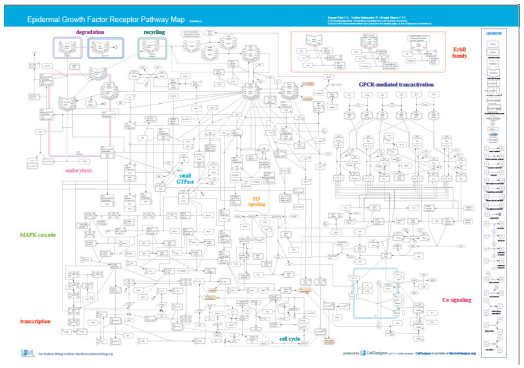
On/Off switches

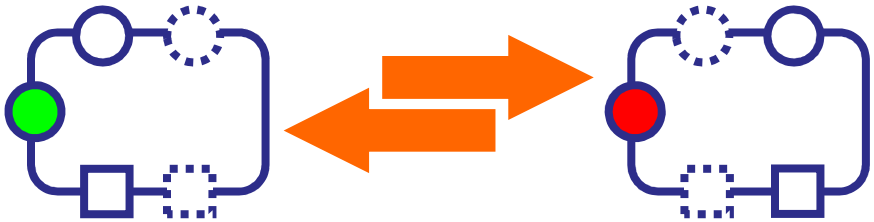


Inaccessible

Binding Sites

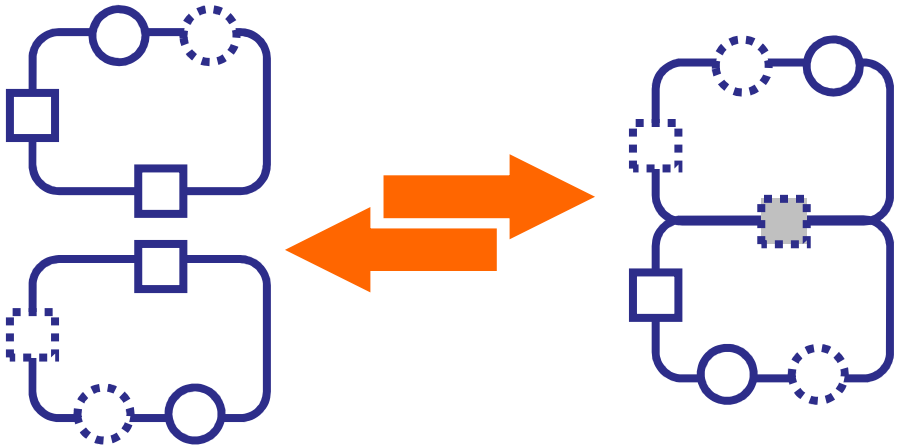






Switching accessible switches

- Depending on current state
- May cause other switches and binding sites to become (in)accessible.



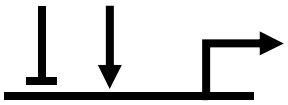
Binding accessible sites

- Depending on current state
- May cause other switches and binding sites to become (in)accessible.

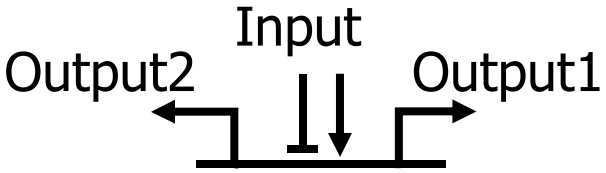
The Informal Model of Gene Interaction

Excitation

Inhibition



Transcription



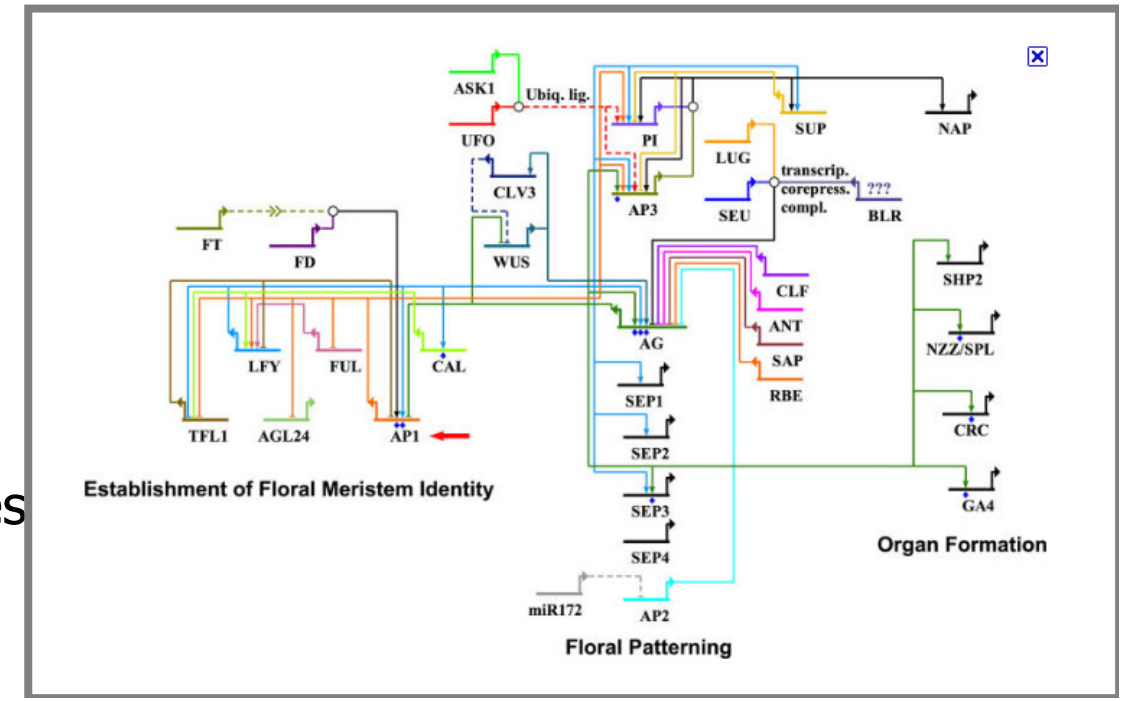
Coding region

Gene

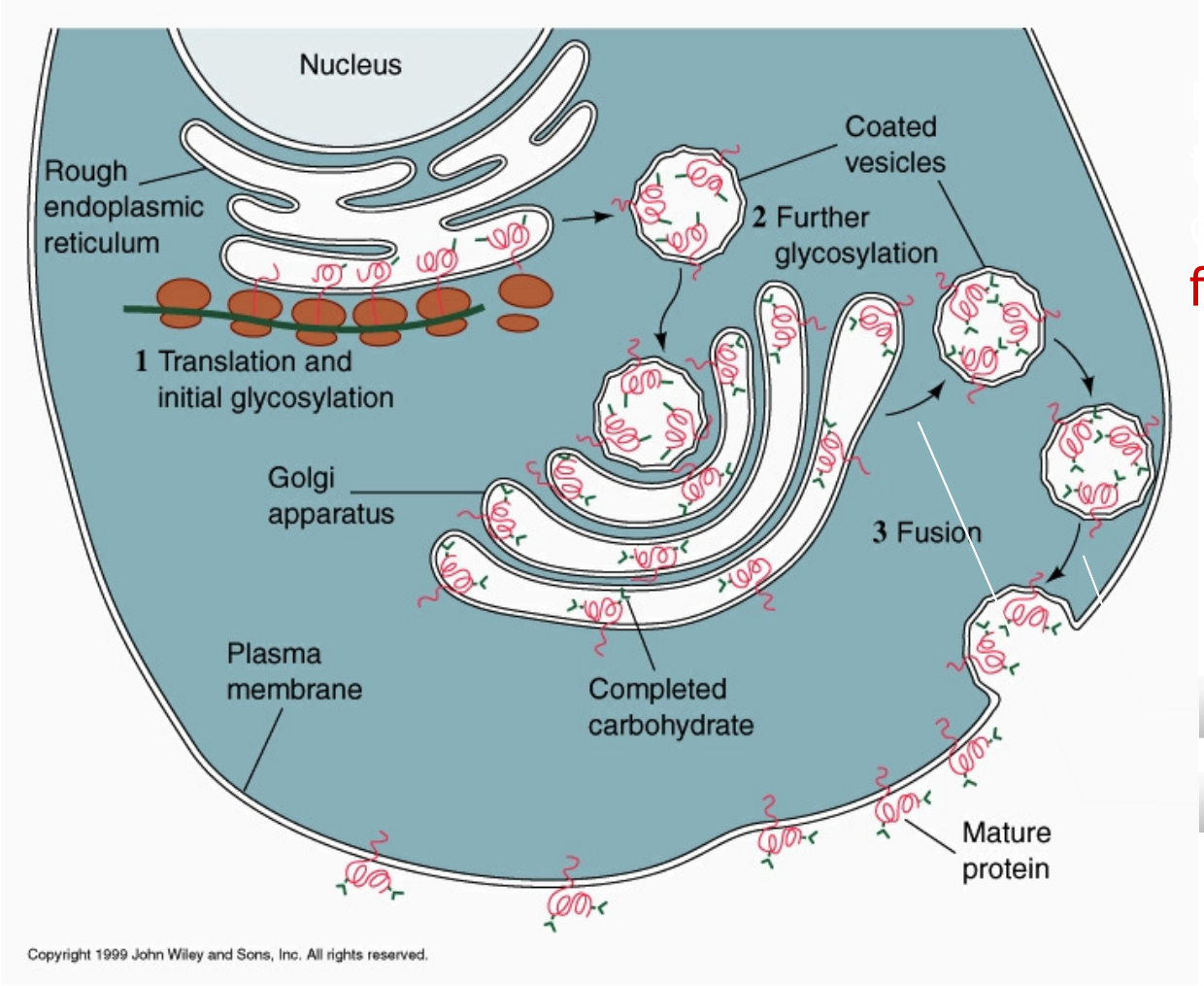
Regulatory region

Regulation of a gene influences transcription. The regulatory region has precise DNA sequences meant for binding regulators.

Transcription produces molecules (RNA or, through RNA, proteins) that bind to regulatory region of other genes (or that are end-products).

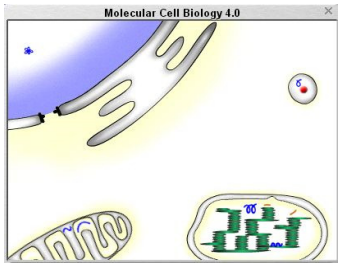


The Informal Model of Membrane Interaction



Voet, Voet & Pratt
Fundamentals of Biochemistry
Wiley 1999. Ch10 Fig 10-22.

fusion fission



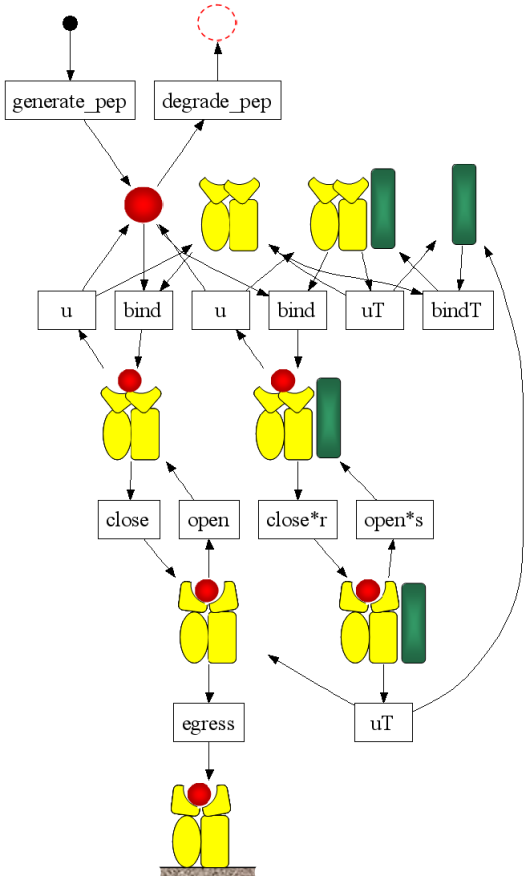
Taken from MCB CD

Fusion

Fission

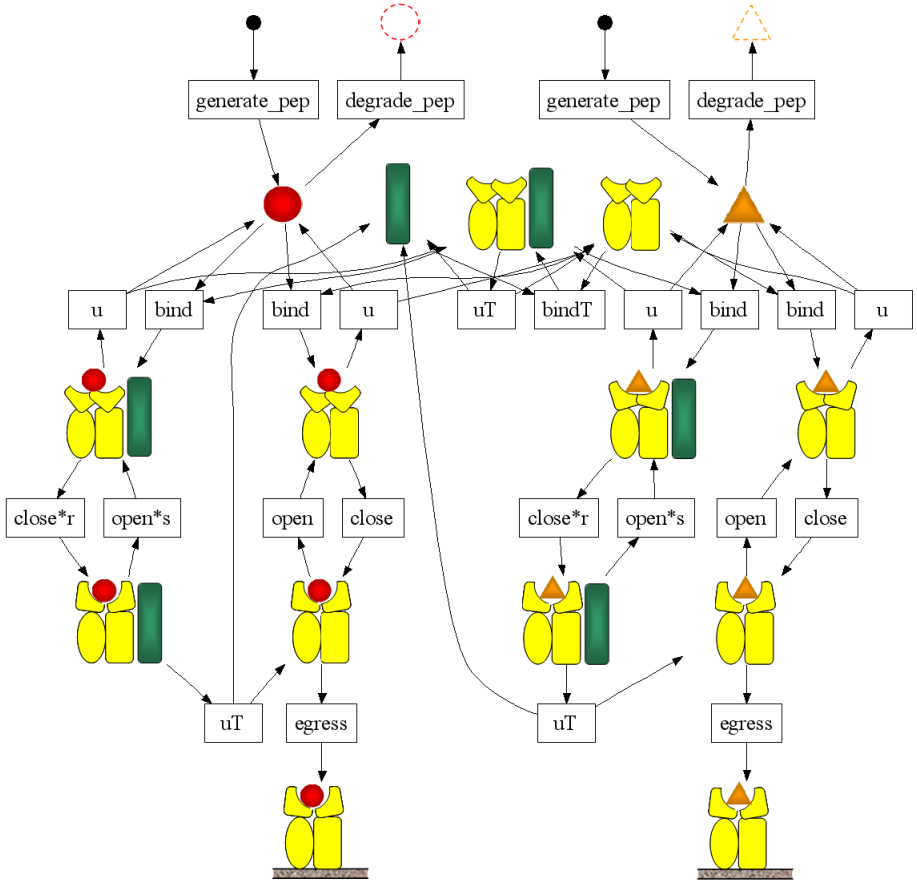
Biological Modelling Today

Describe individual reactions

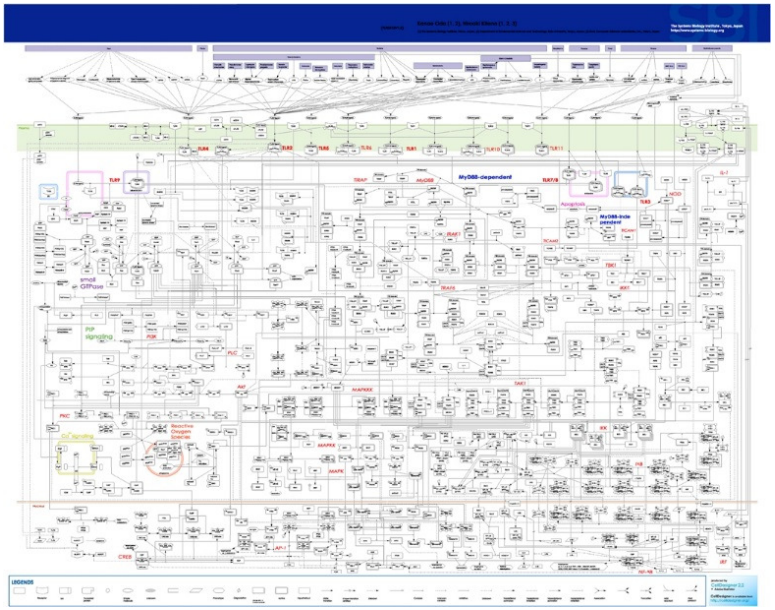


Biological Modelling Today

Add reactions for each new protein.

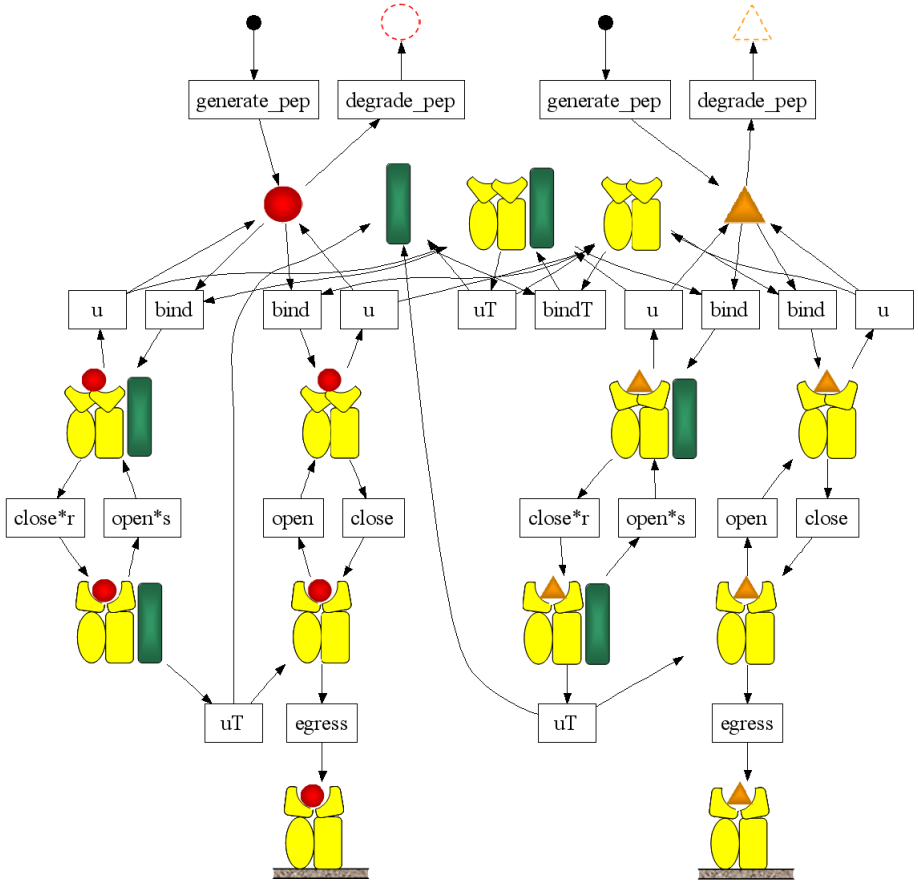


Leading to…

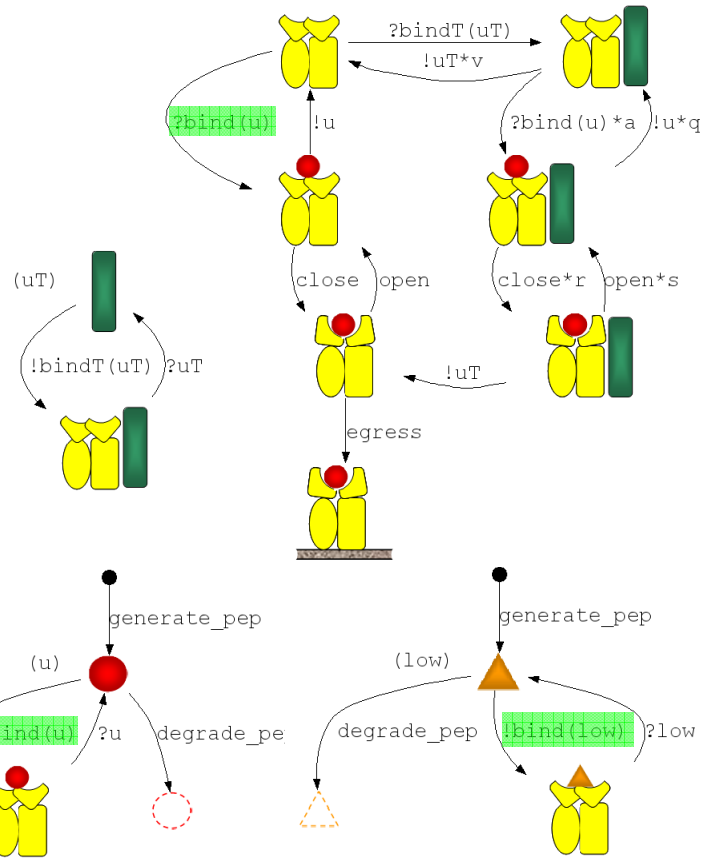


Biological Modelling Tomorrow

Traditional approach:
model the reactions

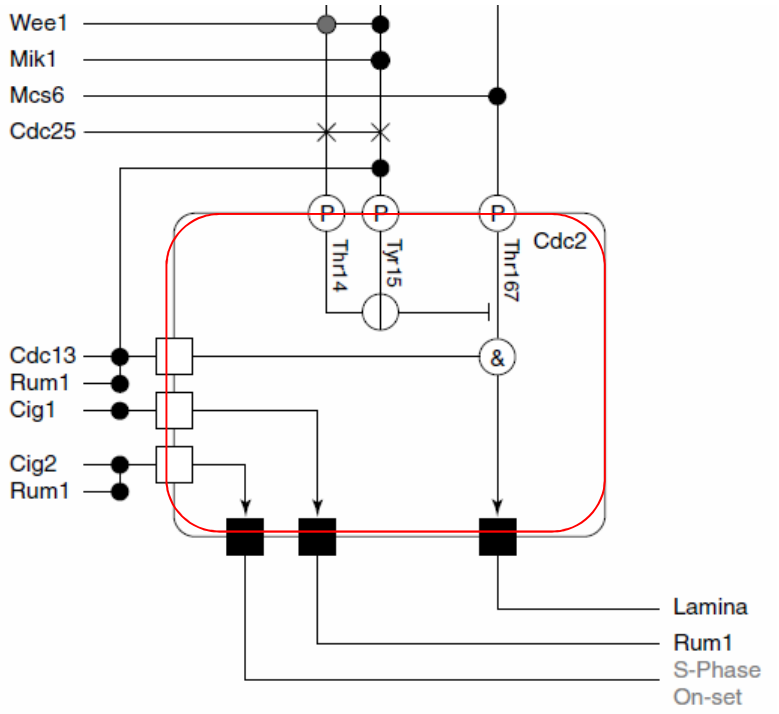


Stochastic Process Algebra:
model the components



Problem: Molecules with State

- Combinatorial explosion of species, reactions, and their state space.



(b) Proposed improvements of graphical representation of fission yeast Cdc2

BIOSILICO Vol. 1, No. 5 November 2003

A graphical notation for biochemical networks

Hiroaki Kitano

Problem: Connected Molecules

- Further combinatorial explosion

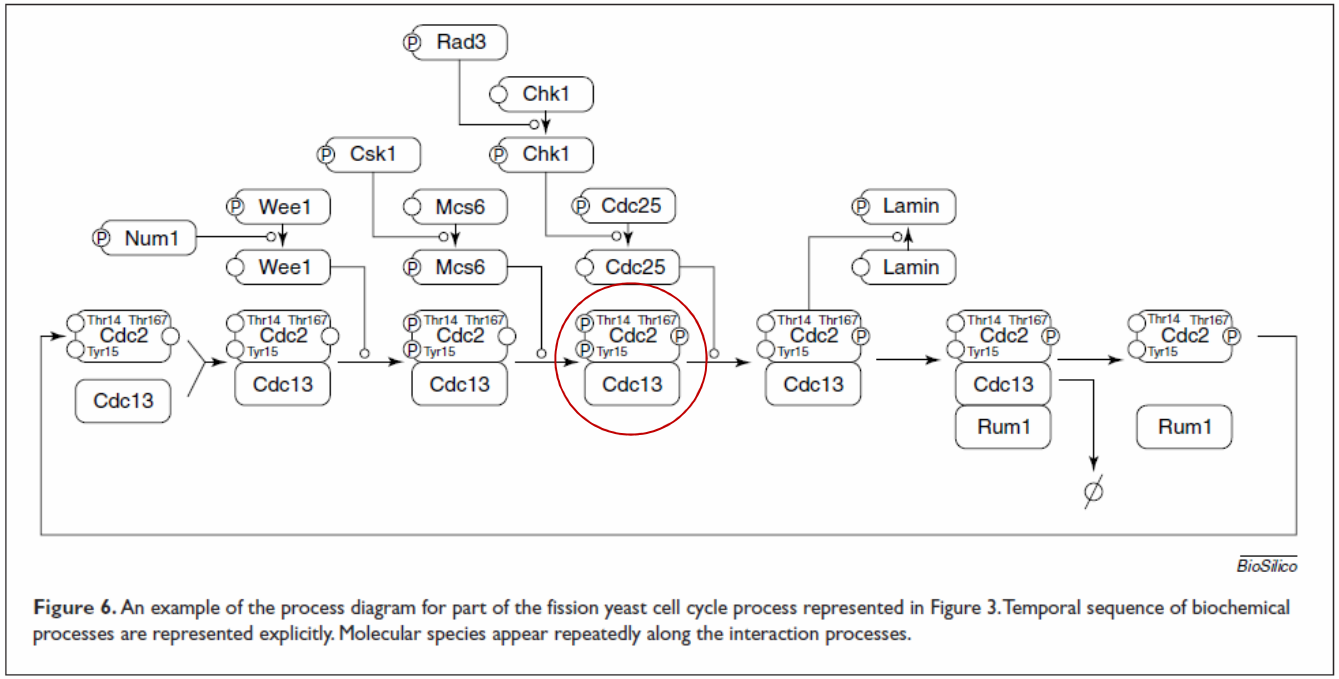


Figure 6. An example of the process diagram for part of the fission yeast cell cycle process represented in Figure 3. Temporal sequence of biochemical processes are represented explicitly. Molecular species appear repeatedly along the interaction processes.

BIOSILICO Vol. 1, No. 5 November 2003

A graphical notation for biochemical networks

Hiroaki Kitano

Problem: Polymers

- ‘Infinite’ explosion

Copolymer equation [edit]

An alternating copolymer has the formula: -A-B-A-B-A-B-A-B-A-B-, or $-(-A-B-)_n-$. The molar ratios of the monomer in the polymer is close to one, which happens when the reactivity ratios r_1 & r_2 are close to zero, as given by the Mayo-Lewis equation also called the **copolymerization equation**.[11]

$$\frac{d[M_1]}{d[M_2]} = \frac{[M_1](r_1[M_1] + [M_2])}{[M_2]([M_1] + r_2[M_2])}$$

where $r_1 = k_{11}/k_{12}$ & $r_2 = k_{22}/k_{21}$

WIKIPEDIA
The Free Encyclopedia

Solution: π-Calculus (or something comparable)

- A solution to combinatorial explosion
  - π-calculus does not have those problems, at least not when you are writing a model.
  - Models are more compact quadratically (for chemical reaction networks) or exponentially (for protein networks) or *infinitely* (for polymerization).
  - The combinatorial explosion still happens at execution (simulation time), but can be handled 'on demand'.
  - The state space is explored incrementally, and even if the state space is actually infinite (as with polymers) we can still simulate it with standard techniques, and (perhaps) analyze it.

π-Calculus for (Bio)Chemistry
(stochastic π-calculus with mass-action interaction law)

- To represent *soups* P we need:
  - Stochastic channels: $(\nu\ x_r)\ P$ r is the rate of an exponential distribution: the rate of communication on that channel
  - Composition: $P \mid P$ (with identity elem. 0)
  - Recursion: $*P$ (equal to P | *P)

- To represent *species* we need:
  - Collision: $?x_r;\ P$ (with no input variables)
  - Co-collision: $!x_r;\ P$ (with no output messages)
  - Delay: $\tau_r;\ P$ (= $(\nu\ x_r)\ ?x_r;P|!x_r;0$ for any x not in P)
  - Choice: $P \oplus P$ (with identity elem. 0)

How (any small) Process Algebra Helps: Abstracting Interfaces in Catalysis

- Two reactions, same catalyst C
  - The catalyst uses one channel for each reaction it catalyzes

a: $A + C \rightarrow^r C + B$

b: $D + C \rightarrow^r C + E$

$C = !a_r; C \oplus !b_r; C$

$A = ?a_r; B$

$D = ?b_r; E$

  - Modularizing: the catalyst has its own catalysis channel c, used for all the reactions it catalyzes:

(nothing comparable)

$C = !c_r; C$

$A = ?c_r; B$

$D = ?c_r; E$

How (full) π-Calculus helps: Representing Complexation

$$A + B \quad {}^{s}\!\leftrightarrow^{r} \quad A{:}B$$

There is no good notation for this reaction in chemistry: A:B is considered as a separate species (which leads to combinatorial explosion of models).

But there is a way to write this precisely in π-calculus. There is a single public *association* channel a_r at rate r, and many private *dissociations* channels d_s at rate s, one for each complexation event (created by ν):

$$A_{free} = (\nu\ d_s)\ !a_r(d_s);\ A_{bound}(d_s)$$
$$A_{bound}(d_s) = !d_s;\ A_{free}$$

$$B_{free} = ?a_r(d_s);\ B_{bound}(d_s)$$
$$B_{bound}(d_s) = ?d_s;\ B_{free}$$

Note that we are describing A *independently* of B: as in the catalysis example, A could form complexes with many different species over the a_r channel.

More compactly:

$A = (\nu\ d_s)\ !a_r(d_s);\ !d_s;\ A$
$B = ?a_r(d_s);\ ?d_s;\ B$

How π-Calculus helps (*infinitely*): Representing Polymerization

- Polymerization is iterated complexation
  - It can be represente in π-calculus *finitely*, with one process (definition) for each monomer.
  - Note that polymerization cannot be described *finitely* in chemistry (or ODEs) because there it needs one reaction for each *length* of polymer.
  - The reason it works in π-calculus is because of the ν operator. It enables the finite representation of systems of potentially unbounded complexity.
  - Like in the genome: the structure of each monomer is coded in a finite description, and yet unbounded-length polymers happen. *Otherwise, there would be no space in the genome to code all those reactions!*

Recent Progress:
A Host of New Molecular Process Algebras

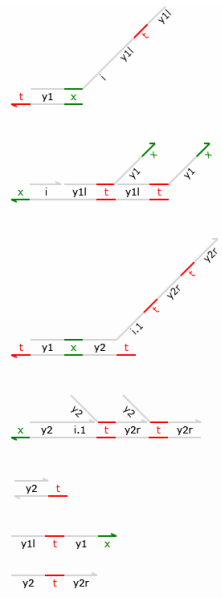
- Activities have since moved away from classical "raw" process algebras, for more domain-specificity
  - Reaction-Based (A + B $\rightarrow$ C + D) (Chemistry)
    - Limited to finite set of species (no polymerization)
    - Practically limited to small number of species (no run-away complexation)
  - Interaction-Based (A = !c. B) (Specialized Process Algebra)
    - Reduces combinatorial complexity of models by combining independent submodels connected by interactions.
  - Rule-Based (A{-}:B{p} $\rightarrow$ A{p}:B{-}) (Logic, Graph Rewriting)
    - Further reduces model complexity by describing molecular state, and by allowing one to 'ignore the context': a *rule* is a reaction in an unspecified (complexation/phosphorylatio) context.
    - Similar to informal descriptions of biochemical events ("narratives").
  - Formal connections
    - The latter two can be translated (to each other and) to the first, but doing so may introduce an infinite, or anyway *extremely large*, number of species.
- But these are still process algebras (interaction/composition based) with all the basic semantic requirements of classical process algebras, and in addition quantitative requirements, and requirement of flexible high-level modeling in certain domains.

Applications in Synthetic Biology

DNA Strand Displacement (DSD)

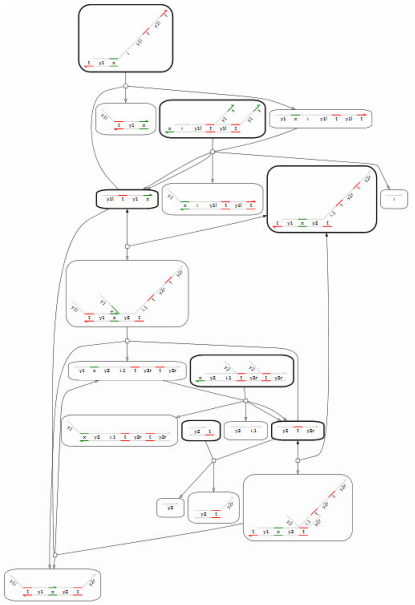
Designing *DNA Hardware*: computers inside cells

Step 1: Program circuit design

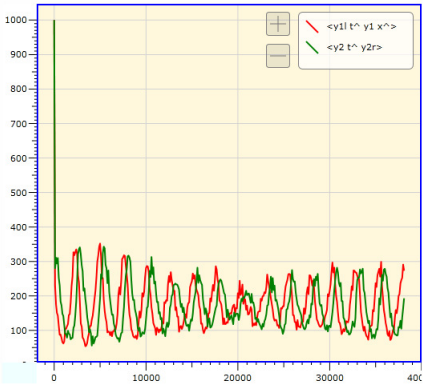


```
directive sample 200000.0 2000
directive plot <y1l t^ y1 x^>; <y2 t^ y2r>
directive leak 1.0E-10 (*1.0E-12*)

def Scale = 1
def Excess = 1000
def bind = 0.00001
def unbind = 0.1

new x@ bind,unbind
new t@ bind,unbind

def SpeciesL(N,al,a) = N * Scale * <al t^ a x^>
def SpeciesR(N,a,ar) = N * Scale * <a t^ ar>
def BinaryLRxRR(N,al,a,b,br,c,cr,d,dr) =
  new i
  ( constant N * Scale * t^:[a x^ b]<i t^ cr t^ dr>:t^
  | constant N * Excess * x^:[b i]:<c>[t^ cr]:<d>[t^ dr]
  )
def UnaryLxLL(N,al,a,cl,c,dl,d) =
  new i
  ( constant N * Scale * t^:[a x^]<i cl t^ dl t^>
  | constant N * Excess * x^:[i]:[cl t^]<c x^>:[dl t^]<d x^>
  )
def UnaryRx(N,a,ar) =
  constant N * Scale * [a]:t^

( UnaryLxLL(1000,y1l,y1,y1l,y1,y1l,y1)
| BinaryLRxRR(30000,y1l,y1,y2,y2r,y2,y2r,y2,y2r)
| UnaryRx(1000,y2,y2r)
| SpeciesL(1000,y1l,y1)
| SpeciesR(1000,y2,y2r)
)
```

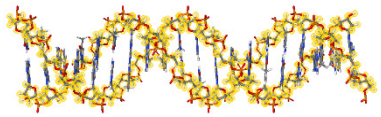
Step 2: Compile circuit behaviour



Step 3: Simulate circuit



Step 4: Compile program to DNA



Interface focus | IET | FirstCite e-publishing

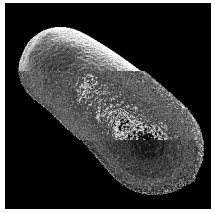
J. R. Soc. Interface
doi:10.1098/rsif.2009.0072.focus
Published online

A programming language for composable DNA circuits

Andrew Phillips* and Luca Cardelli

Microsoft Research, Cambridge CB3 0FB, UK

Step 5: Insert DNA into cells



Formal Syntax and Stochastic Semantics

D	syntax	description
M	N	Long Domain
	N~	Short domain
S	M	Domain
	M*	Complement Domain
	S1 S2	Concatenation of S1 and S2
L,R	_	Empty Concatenation
	S	Domain Concatenation

	syntax	description
A	<S> S	Upper strand with domain concatenation S
	{S} S	Lower strand with domain concatenation S
G	{L'}<L>[S]<R>{R'}	Double stranded complex [S] with overhanging single strands <L>, <R> and {L'}, {R'}
	G1:G2	Gates joined along a lower strand
	G1::G2	Gates joined along an upper strand
D	A	Strand A
	G	Gate G
	D1 | D2	Parallel systems D1, D2
	new N D	System D with private domain N
	X(ñ)	Module X with parameters ñ

before	rule	after
{L' N~* R'} | <L N~ R> L N R L' N* R'	RB_{N+}	{L'}<L>[N~]<R>{R'} L R N N*
{L'}<L>[N~]<R>{R'} L R N N* L' R'	RU_{N-}	{L' N~* R'} | <L N~ R> L N R L' N* R'
{L'}<L>[S]<N~ R>{N~* R'} L R S N+ S* N- L' R'	RC_{N^-}	{L'}<L>[S N~]<R>{R'} L R S N S* N~ L' R'
{L'}<L>[S1]<S R2>: <L1>[S S2]<R>{R'} L R L1 S R2 S1 S S2 S1* S* S2* L' R'	RM_{S^-}	{L'}<L>[S1 S]<R2>: <L1 S>[S2]<R>{R'} L R L1 S R2 S1 S S2 S1* S* S2* L' R'
{L'}<L>[S1]<S R>: <L2>[S]<R2>{R'} L R L2 S R2 S1 S S1* S* L' R'	RD_{S^-}	<L2 S R2> | {L'}<L>[S1 S]<R>{R'} L2 S R2 L R S1 S S1* S* L' R'

rule	condition	before	reduce	after
RGA1	{S1} | <S2> $\xrightarrow{R,r}$ G	<L>{S1}[S]{R'}<R> | <S2>	$\xrightarrow{R,r}$	G:<L>[S]{R'}<R>
RGA2	G $\xrightarrow{R,r}$ {S1} | <S2>	G:<L>[S]{R'}<R>	$\xrightarrow{R,r}$	<L>{S1}[S]{R'}<R> | <S2>
RGB	G | A $\xrightarrow{R,r}$ G'	U1:G:U2 | A	$\xrightarrow{R,r}$	U1:G':U2
RGU	G $\xrightarrow{R,r}$ G' | A	U1:G:U2	$\xrightarrow{R,r}$	U1:G':U2 | A
RGL	G | A $\xrightarrow{R,r}$ G' | A'	U1:G:U2 | A	$\xrightarrow{R,r}$	U1:G':U2 | A'
RG	G $\xrightarrow{R,r}$ G'	U1:G:U2	$\xrightarrow{R,r}$	U1:G':U2
RV	D $\xrightarrow{R,r}$ D'	rev(D)	$\xrightarrow{R,r}$	rev(D')
RC	D $\xrightarrow{R,r}$ D'	com(D)	$\xrightarrow{R,r}$	com(D')
RE	D1$\equiv_\sigma$D2$\xrightarrow{R,r}$D2'$\equiv_\sigma$D1'	D1	$\xrightarrow{R,r}$	D1'

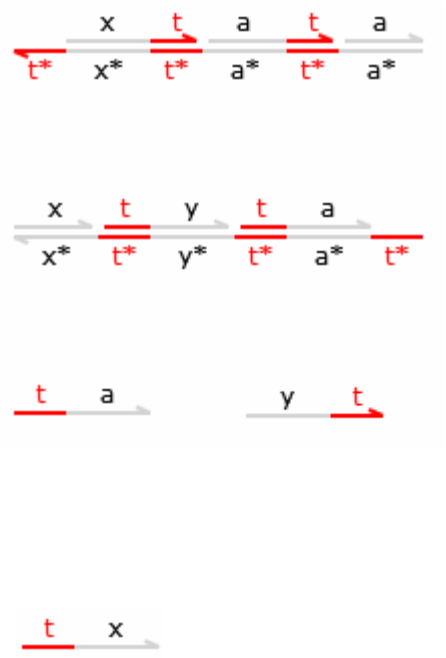
rule	condition	before	equal	after
EC		D1 | D2	$\equiv_\sigma$	D2 | D1
EA		D1 | (D2 | D3)	$\equiv_\sigma$	(D1 | D2) | D3
ED	X(m) = D	X(n)	$\equiv_\sigma$	D{m:=n}
ENP	N $\notin$ fn(D2)	(new N D1) | D2	$\equiv_\sigma$	new N (D1 | D2)
ENN		new N1 new N2 D	$\equiv_\sigma$	new N2 new N1 D
END	N $\notin$ fn(D)	new N D	$\equiv_\sigma$	D
EP	D1 $\equiv_\sigma$ D1'	D1 | D2	$\equiv_\sigma$	D1' | D2
EN	D $\equiv_\sigma$ D'	new N D	$\equiv_\sigma$	new N D'
EL	G $\equiv_\sigma$ G'	G1:G	$\equiv_\sigma$	G1:G'
ER	G $\equiv_\sigma$ G'	G:G2	$\equiv_\sigma$	G':G2
EROTG		G	$\equiv_\sigma$	rotate(G)
EROTA		A	$\equiv_\sigma$	rotate(A)
ESL		{L1'}<L1>[S1]<R1>{R1' S}	$\equiv_\sigma$	{L1'}<L1>[S1]<R1>{R1'}
		:{L2'}<L2>[S2]<R2>{R2'}		:{S L2'}<L2>[S2]<R2>{R2'}
ESU		{L1'}<L1>[S1]<R1 S>{R1'}	$\equiv_\sigma$	{L1'}<L1>[S1]<R1>{R1'}
		::{L2'}<L2>[S2]<R2>{R2'}		::{L2'}<S L2>[S2]<R2>{R2'}

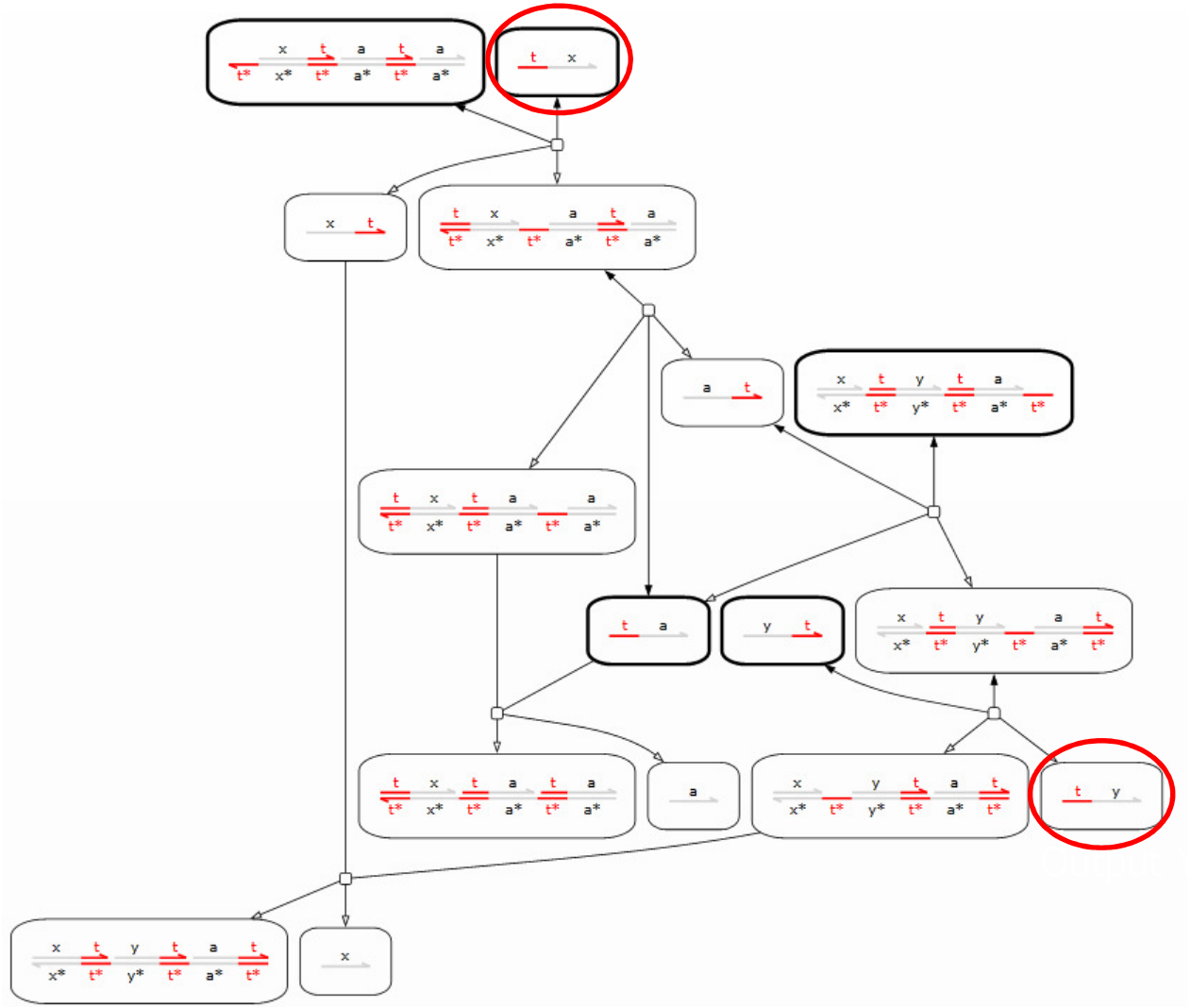
Based on the semantics:

Stochastic or Deterministic Simulation

Generating the associated chemical reaction network for simulation

X→Y transducer

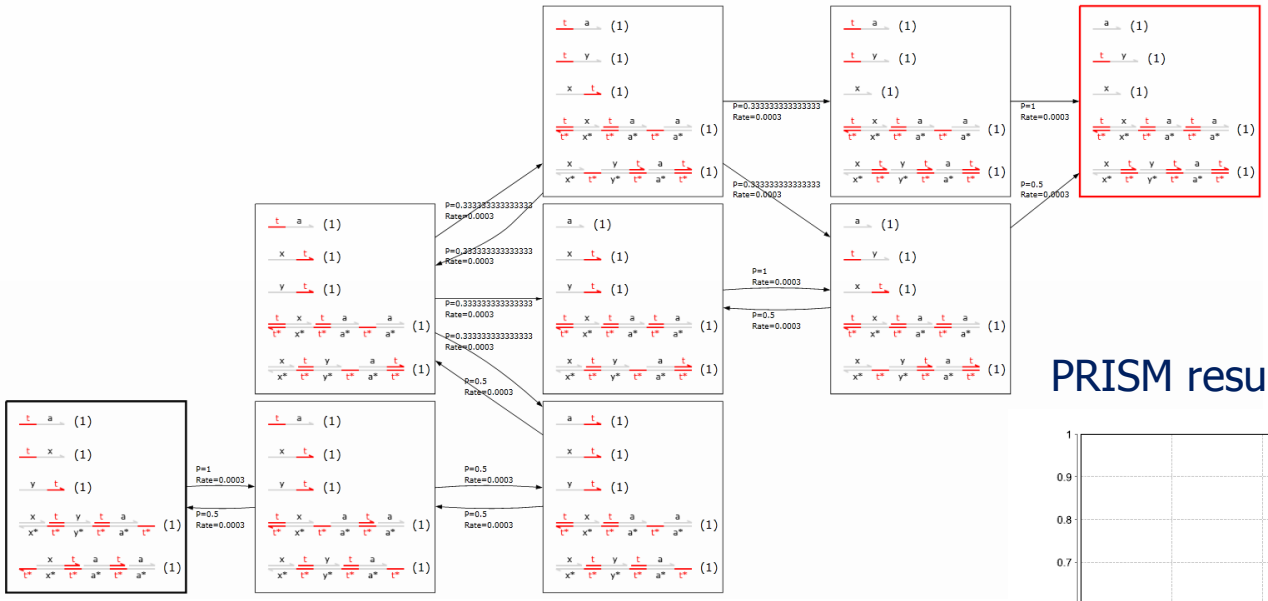




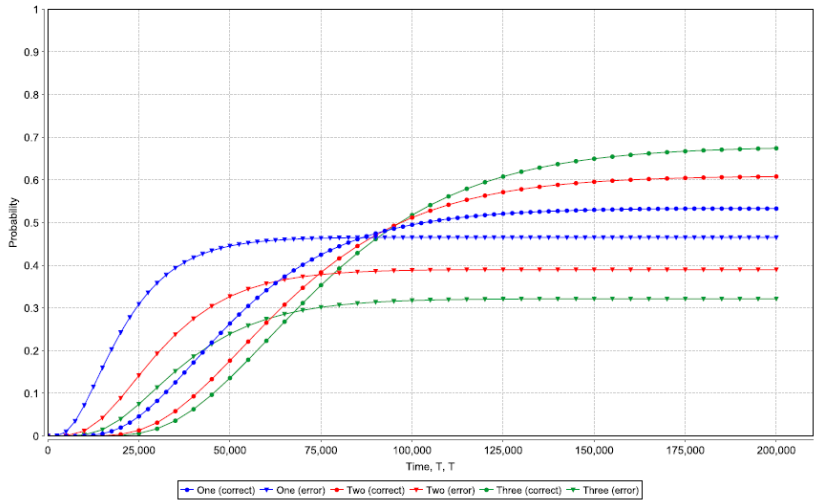
Based on the semantics:

Stochastic Modelchecking

Generating the associated CTMC



PRISM results for sequential transducers



Based on the semantics:

Theorem Proving

Checking invariants in Z3

In progress ... (Biological Computation group & Constraint Reasoning group, MSRC)

We have integrated the Z3 theorem prover in our software to debug the circuit designs. Properties we have checked include whether circuits perform garbage-collection of DNA strands, which would otherwise interfere with the functioning of the circuit, and whether a well-formed terminal state is always reached. When a design error is detected, Z3 generates a counter example showing a path leading to the error.

A More Abstract View

- The semantics of DSD systems is very detailed, and talks about the physical structure of the components, so I will not present it here.

- But basically:

  - There are signals (single DNA strands)
  - There are gates that transduce signals (double DNA strands)
  - There are populations of those (consumed during computation)

Strand Algebra

n x m gates

$$P ::= x \mid [x_1,..,x_n].[y_1,..,y_m] \mid 0 \mid P|P \mid P^* \qquad n \geq 1,\ m \geq 0$$

x	is a *signal*
$[x_1,..,x_n].[y_1,..,y_m]$	is a *gate*
0	is an *inert solution*
$P\|P$	is *parallel composition* of signals and gates
P^*	is a *population* (multiset) of signals and gates

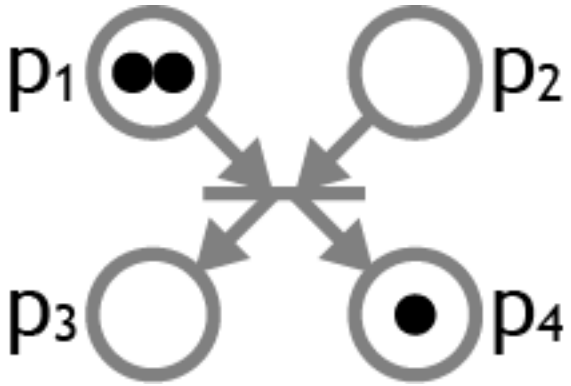
Reaction Rule

$$x_1 \mid .. \mid x_n \mid [x_1,..,x_n].[y_1,..,y_m] \rightarrow y_1 \mid .. \mid y_m$$

Equivalent to place-transition Petri Nets.
(Stochastic ones if adding rates)

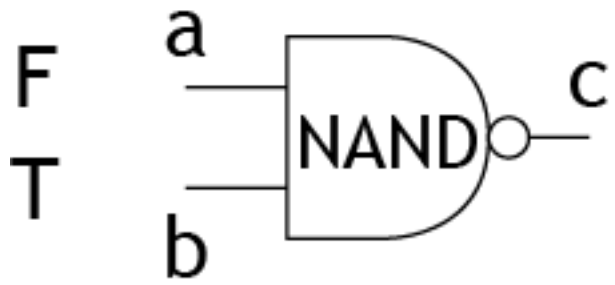
Representing Petri Nets

Transitions as Gates
Place markings as Signals



$([p_1, p_2] . [p_3, p_4])^* \mid$
$p_1 \mid p_1 \mid p_4$

Representing Boolean Networks



$$([a_F, b_F].c_T)^* \mid$$
$$([a_F, b_T].c_T)^* \mid$$
$$([a_T, b_F].c_T)^* \mid$$
$$([a_T, b_T].c_F)^* \mid$$
$$a_F \mid b_T$$

This encoding is *compositional*, and can encode *any* Boolean network:
- multi-stage networks can be assembled (combinatorial logic)
- network loops are allowed (sequential logic)

Representing Chemistry

Translate reaction by reaction, and put everything in parallel with the initial molecules. (In a stochastic version, one must turn P* into a k-weighted $P^{=k}$).

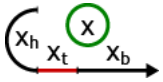
Chemistry	Strand Algebra
$A \rightarrow C_1 + \ldots + C_n$	$(A.[C_1, \ldots, C_n])^*$
$A + B \rightarrow C_1 + \ldots + C_n$	$([A,B].[C_1, \ldots, C_n])^*$
$A_1 + \ldots + A_n$	$A_1 \mid \ldots \mid A_n$ Initial solution

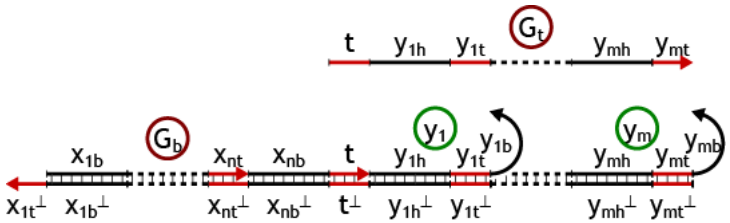
Almost trivial at this level, but by doing so,
we transform chemistry into an executable programming language!

D. Soloveichik, G. Seelig, E. Winfree, "DNA as a Universal Substrate for Chemical Kinetics". PNAS 107 (12): 5393-5398, 2010

Compiling Strand Algebra to DNA

$P ::= x \mid [x_1,..,x_n].[y_1,..,y_m] \mid 0 \mid P|P \mid P^* \qquad n \geq 1, m \geq 0$

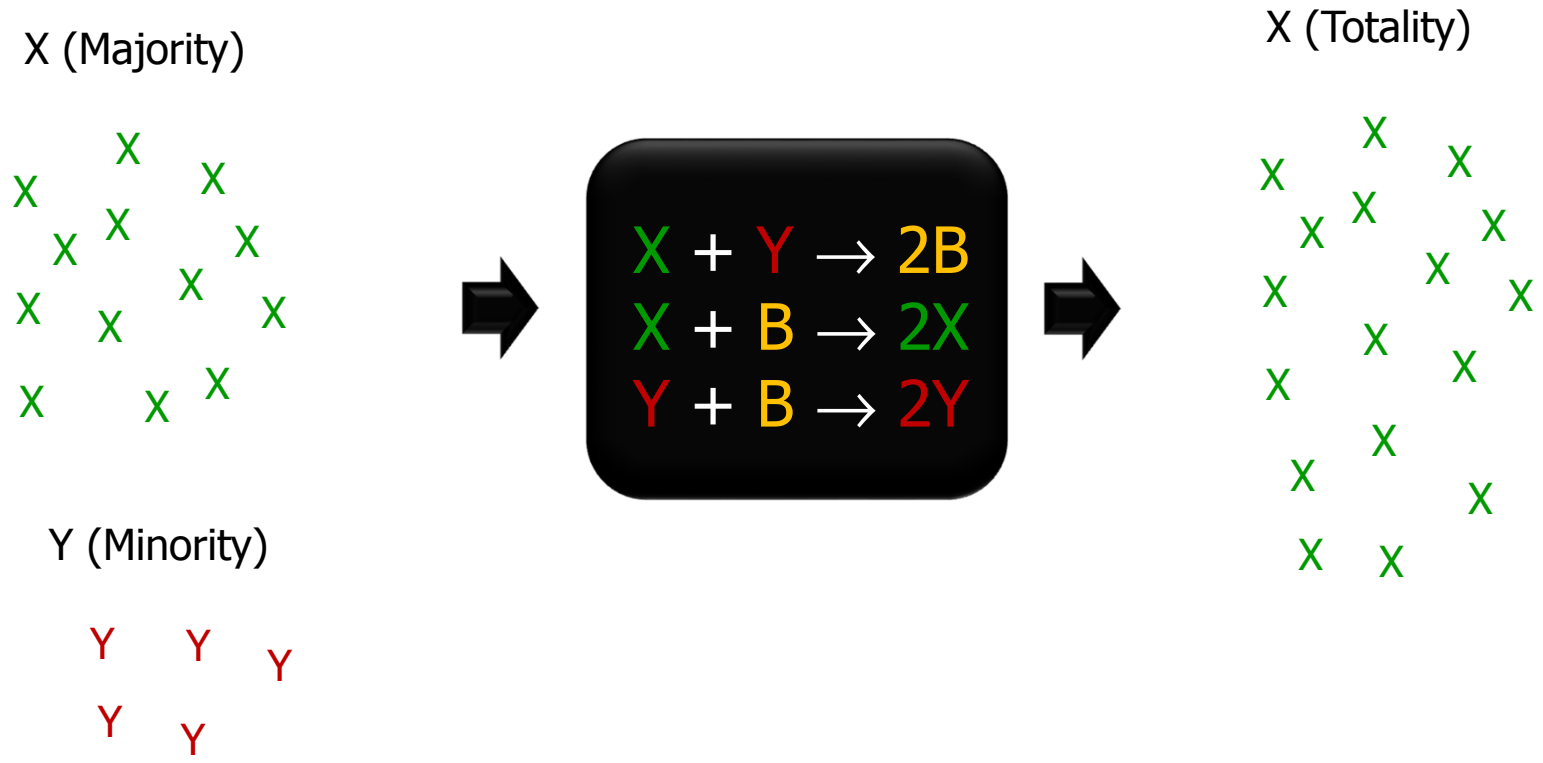
- compile(x) = 

- compile($[x_1,..,x_n].[y_1,..,y_m]$) = 

- compile(0) = empty solution
- compile(P | P') = mix(compile(P), compile(P'))
- compile(P*) = population(compile(P))

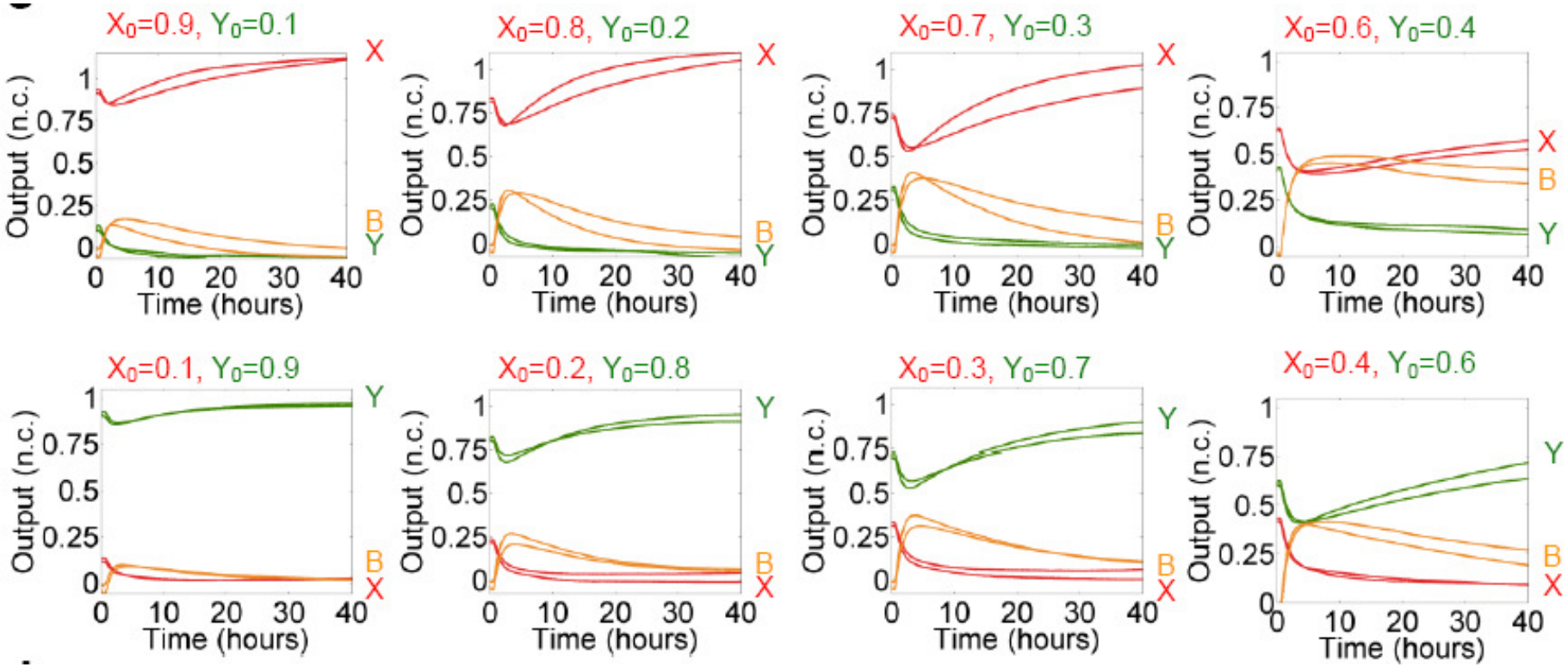
“Executable Chemical Algorithms”

A Consensus Algorithm (Approximate Majority)



Chemical Program => Strand Algebra => DSD structure => (real) DNA

Approximate Majority experiments (Seelig Lab, U.W.)



Correctness of Compilation

- The spec of a transducer:

$$x.y \mid x \rightarrow y$$

  - Is it true at all?
  - Is it true *possibly, necessarily,* or *probabilistically ?*
  - Is it true in the context of a *population of identical transducers?*
  - Is it true *in all possible contexts?*
  - Is it true (only) for *infinite populations*?

Stochastic Operational Semantics

Process Algebras in the Wet Lab

- Common to Systems and Syntetic Biology applications:
  - Need to describe real-time evolution of complex systems
    - both stochastic and deterministic
  - Need to relate our models to standard ones from the literature:
    - chemical reaction networks
    - ordinary differential equations (deterministic)
    - chemical master equations (stochastic)
  - Need to verify quantitiative properties
    - In experimental design (correctness)
    - In experimental validation (parameter inference)
  - Need to relate process algebras to "chemical semantics": stochastic and deterministic kinetics

The Fouth Era: Quantitative Semantics

In Computer Science we now have only a few mainstream programming languages

- (Fortunately still a wealth of specification languages...)

But in Computationa Biology, complex networks are represented by a variety of

- modelling languages for systems biology
- programming languages for synthetic biology
- both areas are vast in scope and requirements, and in their infancy
- leading to an explosion in new laguages and creativity

So an old problem resurfaces: how to give precise *and* flexible meaning to all these (now stochastic) languages? Our approach:

- start with a warm-up exercise: no-value-passing no-recursion CCS [QEST'10]
- proceed directly to the "worst case scenario": π-calculus [ICTAC'13]
- Based on *much* prior work in theory of Markov processes, measure theory, automata theory, performance evaluation, and process algebra

Stochastic Process Algebras

- The Labeled Transition Systems of standard SOS are replaced by (labelled) Markov processes (e.g., CTMCs)
  - nondeterministic a-transition:

    $P \xrightarrow{a} Q$
  - stochastic (Markovian) a-transition:

    $P \xrightarrow{a,r} Q$

    $r \in [0,+\infty)$ is the rate of an exponentially distributed random variable that characterises the a-transitions from P to Q.

- In recent decades a plethora of SPAs appeared, such as
  - TIPP (Gotz, Herzog, Rettelbach)
  - PEPA (Hillston)
  - EMPA (Bernardo, Gorrieri)
  - Stochastic pi-calculus (Priami, Degano)
  - StoKlaim (De Nicola, Katoen, Latella, Loreti, Massink)

  etc.

The challenge of stochastic processes

- "Pointwise" semantics, similar to nondeterministic PAs, faces *counting problems*, and the known SPAs solve them using rather complex solutions such as the multi-transition system (PEPA) or the proved SOS (stochastic pi-calculus).

Problems Arise:

(B. Klin, V. Sassone, *Structural Operational Semantics for Stochastic Process Calculi*, FOSSACS'08)

- These SOS formalisms are difficult to extend to a general format for well-behaved stochastic specifications;
- In stochastic π-calculus (with proved SOS) parallel composition is not associative up to bisimulation;
- In PEPA, if arbitrary relations between transition rates and the rates of subprocesses are allowed, stochastic bisimulation is not a congruence;

A possible explanation (ibid.): in a well-behaved SOS framework the labels of transitions should only carry as much data as required for the derivation of the intended semantics;

Both the proofs and the transition multiplicities contain superfluous data.

The challenge of stochastic processes

A solution: return to the simplicity and elegance of nondeterministic PAs:

instead the pointiwise semantics, use a semantics based on measures.

$P \xrightarrow{a,r} Q \quad \Rightarrow \quad P \longrightarrow \mu, \quad \mu(a)(\{Q\})=r$

where μ is a measure (indexed by actions) on the measurable space of processes.

Similar approaches

R. Segala, N. Lynch, *Probabilistic Simulations for Probabilistic processes*, 1995.

M. Kwiatkowska, G. Norman, R. Segala, J. Sproston, *Automatic Verification of Real-Time Systems with Discrete Probability Distributions*, 1999.

E. P. de Vink, J. Rutten, *Bisimulation for probabilistic transition systems: A coalgebraic approach*, 1999.

J. Rutten, *Universal Coalgebra: a theory of systems*, 2000.

F. Bartels, *On Generalised Coinduction and Probabilistic Specification Formats*, 2004.

M. Bravetti, H. Hermanns, J.-P. Katoen, *YMCA: Why Markov Chain Algebra?*, 2006.

B. Klin, V. Sassone, *Structural Operational Semantics for Stochastic Process Calculi*, 2008.

R. De Nicola, D. Latella, M. Loreti, M. Massink, *Rate-based Transition Systems for Stochastic Process Calculi*, 2009.

Solving The Counting Problem

- We expect no difference in the behavior of, e.g., Q|R, R|Q, R|Q|0, etc. These trivial equivalences, reflected in the rules of structural congruence, embody the assumption of "well mixed chemical solutions": namely that the probability of interaction is *independent* of the initial location (physically, or here syntactically) of the components.

- Our approach: Use structural congruence to organizes a measurable space of processes; instead of $2^{\mathbf{P}}$ (every syntactic form is measurable) we use the sigma algebra Π generated by $\mathbf{P}^{\equiv}$: only congruence-closed classes are measurable.

- Transitions relates an (initial) process to a (final) *measurable set*: a congruence-closed set of processes. This way if P performs an action $P \xrightarrow{a,r} (Q|R)^{\equiv}$ we can also derive $P \xrightarrow{a,r} (R|Q)^{\equiv}$ without overcounting the rates of the transitions.

- The alternative is to consider *any* set of processes as measurable. Then the rate of an a-transition from P to the set {Q|R, R|Q, R|(Q|0))} obtains the undesired result $P \xrightarrow{a,3r} \{Q|R, R|Q, R|(Q|0)\}$.

- To avoid such problems, in the literature we find complicated variants of SOS that make the theory heavy and often problematic.

Stochastic CCS: The syntax

Let **A** be a denumerable set of action names endowed with

- an involution $*:\mathbf{A} \longrightarrow \mathbf{A}$, $a^* \neq a$, $a^{**}=a$
- a weight function $\iota:\mathbf{A} \longrightarrow \mathbb{Q}^+$, $\iota(a)= \iota(a^*)$ for all $a \in \mathbf{A}$ (*rate* of a)

Let $\mathcal{C} \notin \mathbf{A}$ and $\mathbf{A}^+=\mathbf{A} \cup \{\mathcal{C}\}$

The set **P** of processes are defined by the following grammar, for arbitrary $r \in \mathbb{Q}^+$.

$$P := 0 \mid \varepsilon.P \mid P|P \mid P+P$$

$$\varepsilon := a \in \mathbf{A} \mid \mathcal{C}(r)$$

We extend the weight function ι by $\iota(\mathcal{C}(r))=r$.

Structural Congruence "$\equiv$" is the smallest equivalence relation on **P** that satisfies

I. 1. $P|Q \equiv Q|P$; 2. $(P|Q)|R \equiv P|(Q|R)$; 3. $P|0 \equiv P$.

II. 1. $P+Q \equiv Q+P$; 2. $(P+Q)+R \equiv P+(Q+R)$; 3. $P+0 \equiv P$.

III. if $P \equiv Q$, then for any ε and any $R \in \mathbf{P}$,

1. $P|R \equiv Q|R$; 2. $P+R \equiv Q+R$; 3. $\varepsilon.P \equiv \varepsilon.Q$.

The measurable space

For arbitrary P∈**P**, let $P^{\equiv}$ be the ≡-equivalence class of P and $\mathbf{P}^{\equiv}$ the set of ≡-equivalence classes of processes.

Let ∏ be the sigma-algebra generated by $\mathbf{P}^{\equiv}$ over **P.**
(∏ is a set of subsets of **P**: the closure of $\mathbf{P}^{\equiv}$ under complement and countable union, and contains **P**)

(**P**,∏) is a measurable space. (A set with a sigma-algebra over it.)
The measurable sets are the members of ∏.

Let Δ(**P**,∏) denote the set of measurable functions on (**P**,∏).
(functions f: ∏ $\longrightarrow$ $\mathbb{Q}^+$ such that f(∅) = 0, and f over a union of sets with pairwise disjoint elements is the sum of the f's of the sets)
The null measure is ω(S)=0, for any $S \in \prod$.

For $S, T \in \prod$, let $S \mid T = \bigcup_{P \in S,\, Q \in T} (P|Q)^{\equiv}$ and $S_T = \bigcup_{P|R \in S,\, P \in T} R^{\equiv}$

Lemma: If $S, T \in \prod$ and P∈**P**, then $S \mid T$ and S_T are measurable sets.

Structural Operational Semantics

Has the form:

$$P \longrightarrow \mu,$$

Where, μ gives for any action x and for any measurable set $S \in \Pi$ (closed under structural congruence) the overall transition rate from P through x to (some element of) S.

where $\mu: \mathbf{A}^+ \longrightarrow \Delta(\mathbf{P},\Pi)$ is such that

for each $x \in \mathbf{A}^+$, $\mu(x) \in \Delta(\mathbf{P},\Pi)$ is a measure and

for each $S \in \Pi$, $\mu(x)(S)=r \in \mathbb{Q}^+$,

r is the rate of the x-transition from P to (elements of) S .

Notice that S is not just any set, but a measurable set, e.g. μ(x)({Q}) is undefined.

In simple cases, we can still write, pointwise:

$P \xrightarrow{x,r} Q$ for $P \longrightarrow \mu$ with $\mu(x)(Q^{\equiv}) = r$

Structural Operational Semantics

(Null) $\dfrac{}{0 \longrightarrow \varpi}$ $\qquad \varpi(x)=\omega$ for any $x \in \mathbf{A}^+$,

(Guard) $\dfrac{}{\varepsilon.P \longrightarrow \left[\begin{smallmatrix}\varepsilon\\ P^{\equiv}\end{smallmatrix}\right]}$ $\qquad \left[\begin{smallmatrix}\varepsilon\\ P^{\equiv}\end{smallmatrix}\right](a)=\begin{cases} D(\iota(\varepsilon),P^{\equiv}), & a=\varepsilon \\ \omega, & a\neq\varepsilon \end{cases}$

(Sum) $\dfrac{P \longrightarrow \mu \quad Q \longrightarrow \mu'}{P+Q \longrightarrow \mu \oplus \mu'}$ $\qquad (\mu \oplus \mu')(x)(S)=\mu(x)(S)+\mu'(x)(S)$

(Par) $\dfrac{P \longrightarrow \mu \quad Q \longrightarrow \mu'}{P|Q \longrightarrow \mu\ {}_{P^{\equiv}}\otimes_{Q^{\equiv}}\ \mu'}$ $\qquad$ ... the tricky one ...

Lemma: For any $P \in \mathbf{P}$, there exists a unique $\mu \in \Delta(\mathbf{P},\Pi)^{\mathbf{A}^+}$ such that $P \longrightarrow \mu$.

Notice that we have no rule that guarantees that structural congruent processes have identical behaviour. But we can prove this.

Theorem: If $P \equiv Q$ and $P \longrightarrow \mu$, then $Q \longrightarrow \mu$.

Structural Operational Semantics

The parallel composition "|"

For any $P,Q \in \Pi$, let $\ {}_P\otimes_Q : \Delta(\mathbf{P},\Pi)^{\mathbf{A}^+} \times \Delta(\mathbf{P},\Pi)^{\mathbf{A}^+} \longrightarrow \Delta(\mathbf{P},\Pi)^{\mathbf{A}^+}$
such that for any $\mu,\mu' \in \Delta(\mathbf{P},\Pi)^{\mathbf{A}^+}$, any $S \in \Pi$,

for atomic actions $a \in \mathbf{A}$:
the sum of the two measures "acting independently" through a on parts of *S*.

$$(\mu\ {}_P\otimes_Q\ \mu')(a)(S) = \mu(a)(S_Q) + \mu'(a)(S_P)$$

for ϛ:

$$(\mu\ {}_P\otimes_Q\ \mu')(\varsigma)(S) = \mu(\varsigma)(S_Q) + \mu'(\varsigma)(S_P) + \sum_{a\in\mathbf{A},\ \iota(a)>0}^{T'|T''=S} \frac{\mu(a)(T') \times \mu'(a^*)(T'')}{2\iota(a)}$$

again the sum of the two measures "acting independently" through ϛ on parts of *S* plus the ϛ resulting from interactions of complementary actions a for all possible parallel decompositions of *S*.

Here is where the law of mass action is embodied: in the case of interaction we take the *product* of the rates of the interacting *populations*. (This leads to multiply $\iota(a)\times\iota(a^*)$ and to further count them twice because of symmetry of |, so we divide again by $2\iota(a)$ to get the proper overall rate for ϛ.)

The algebra of measures

We have defined an algebraic structure $(\Delta(\mathbf{P},\Pi)^{\mathbf{A}^+}, \varpi, [\,{}^{\varepsilon}_{P}\,], \oplus, {}_{P}\otimes_{Q})$ with operators defined for arbitrary ε, P and Q.

Lemma:

I. (1) $\mu \oplus \mu' = \mu' \oplus \mu$,
(2) $(\mu \oplus \mu') \oplus \mu'' = \mu \oplus (\mu' \oplus \mu'')$,
(3) $\mu \oplus \varpi = \mu$.

II. (1) $\mu \; {}_{P}\otimes_{Q} \mu' = \mu' \; {}_{Q}\otimes_{P} \mu$,
(2) $(\mu \; {}_{P}\otimes_{Q} \mu') \; {}_{P|Q}\otimes_{R} \mu'' = \mu \; {}_{P}\otimes_{Q|R} (\mu' \; {}_{Q}\otimes_{R} \mu'')$,
(3) $\mu \; {}_{P}\otimes_{0} \varpi = \mu$.

Theorem:

Stochastic bisimulation is a congruence, i.e.,

1. if $P \backsim P'$, then for arbitrary ε, $\varepsilon.P \backsim \varepsilon.P'$;
2. if $P \backsim P'$ and $Q \backsim Q'$, then $P+P' \backsim Q+Q'$;
3. if $P \backsim P'$ and $Q \backsim Q'$, then $P|P' \backsim Q|Q'$

Stochastic π-Calculus Revisited

- In the same style of measure-based operational semantics:
  - Add name restriction (ν a@r) P and name-passing input/output. This further complicates the operator for |, but also simplifies the treatment of bound output w.r.t. ordinary π-calculus: no need for higher-order abstraction-concretions here.
  - Remove ↻ because it is now definable.
  - Add recursion !P. This can no longer be included in structural congruence because otherwise rates become infinite. It is instead handled through a (simple) rule. This is the main way stochastic π-calculus deviates from the original π-calculus, which otherwise could be seen as stochastic-π with all the rates = 1.
- Paper in the proceedings.

Conclusions

- We took the challenge of reconsidering Stochastic Process Algebras from a foundational perspective, to facilitate the application of its basic principles to new domains.

- The goals:
  - understanding if the "counting " approaches can be avoided
  - providing well-behaved SOS formats similar to the formats of nondeterministic PAs

- The way to do it:
  - center the work on the equational theory of structural congruence
  - lift the algebraic structure from the space of processes to the space of measures

- Advantages:
  - an elegant and compact SOS
  - well-behaved SOS: bisimulation is a congruence that extends structural congruence
  - a simple extension to metric semantics
  - simple solutions to the problems related to recursion and bound output