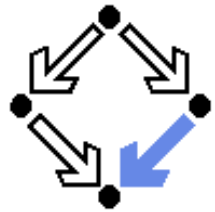
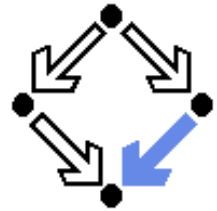
Biological Systems as Concurrent Processes
Part 1: Modeling and Simulating

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.uni-linz.ac.at



Computational Systems Biology

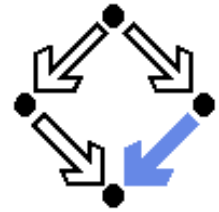


Representing the structure and function of biological systems via formal languages, for description, simulation, analysis and (eventually) compilation. (Luca Cardelli)

- New view: biological systems as discrete (computing) systems.
  Traditional view: continuous (physical) systems.
  - Discrete (non-linear) transitions.
  - Deep layering of abstractions.
  - Complex composition of simpler components.
  - Digital coding of information.
  - Reactive information-driven behavior.
  - Very high degree of concurrency.
- Survey (by Luca Cardelli):
  `http://lucacardelli.name/BioComputing.htm`

Apply techniques from computer science to the *modeling, simulating, specifying and verifying* of biological systems.

Modeling Systems



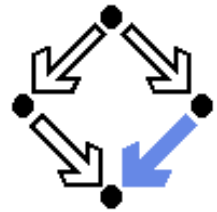
- Software/hardware systems may consist of multiple components.
  - Typical features:
    - Concurrency: Components execute simultaneously.
    - Interaction: Components may communicate with their neighbors.
    - Mobility: Components may move to another neighborhood.
  - May be formally modelled in some calculus.
    - E.g. the π-calculus (Milner, 1992).
- Biochemical systems may be viewed in a similar way.
  - Molecules as processes, molecule reactions as process interactions.
    - E.g. π-calculus model of a signal transduction pathway.
  - BioSPI project:

    `http://www.wisdom.weizmann.ac.il/~biospi`
  - SPiM simulator:

    `http://research.microsoft.com/~aphillip/spim`

Use of the (stochastic) π-calculus to model biochemical systems.

The π-Calculus: Syntax



- Actions: $\pi ::= x(y) \mid \overline{x}\langle y \rangle$

  $x(y)$ ... receive y along port x

  $\overline{x}\langle y \rangle$ ... send y along port x

- Processes: $P ::= \sum_{i \in I} \pi_i.P_i \mid P_1|P_2 \mid (\text{new } x)\ P \mid !P$

  $\sum_{i \in I} \pi_i.P_i$ ... summation (non-determinism between the P_i)

  $P_1|P_2$ ... composition (concurrent execution of P_1 and P_2)

  $(\text{new } x)\ P$ ... restriction (introduction of new port x in P)

  $!P$ ... replication (unbounded duplication of P)

- Syntactic sugar: (definable by above kinds of processes)

  0 ... terminating process

  $P + Q$ ... summation of P and Q

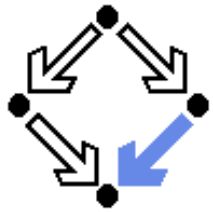
  $(\text{new } x_1, x_2, \ldots, x_n)\ P$ ... generalized restriction

  $\text{if } C \text{ then } P \text{ else } Q$ ... conditional process

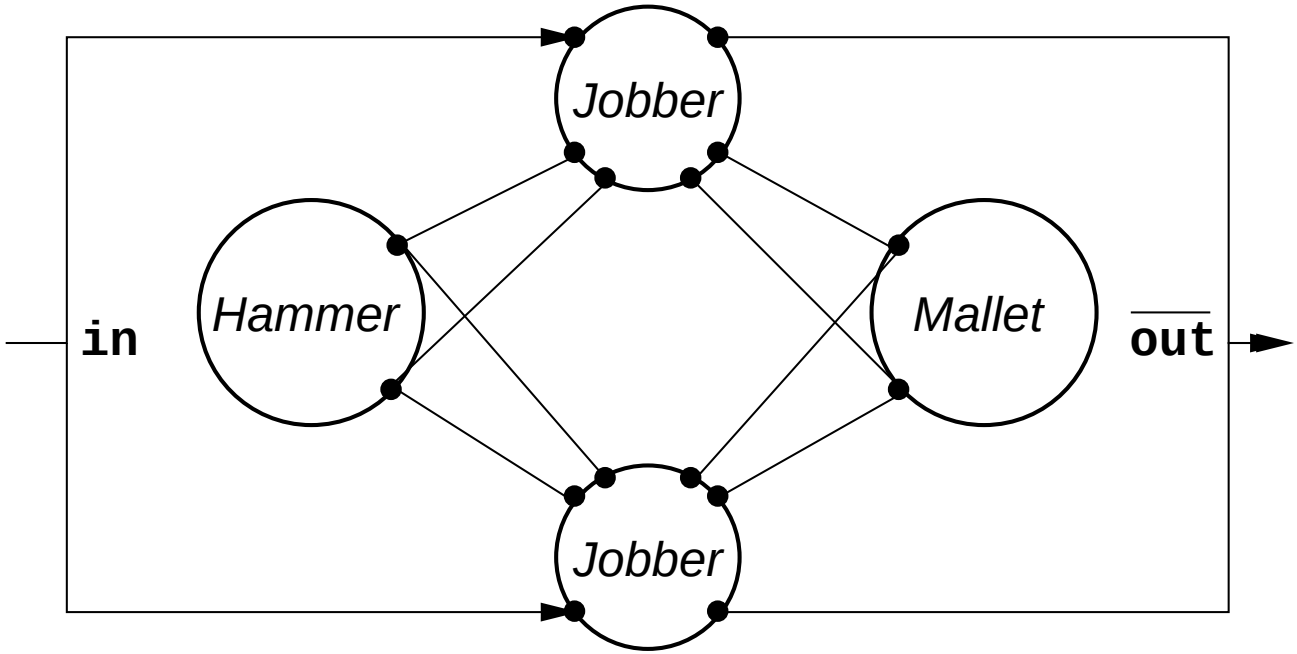
  $A\langle y \rangle$ ... invocation of process $A(x) := P$

A language of process descriptions.

Example: A Job Shop

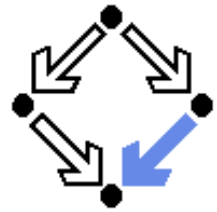


Two jobbers with a hammer and a mallet.



A system with four processes.

Example: A Job Shop



Formal process descriptions:

$$Jobshop := (\text{new } getH, putH, getM, putM)\ (Jobber|Jobber|Hammer|Mallet)$$

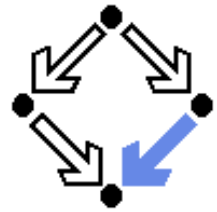
$$Hammer := getH.\overline{putH}.Hammer$$
$$Mallet := getM.\overline{putM}.Mallet$$

$$Jobber := inE.Easy + inN.Normal + inD.Difficult$$

$$Easy := Done$$
$$Normal := \overline{getH}.putH.Done + \overline{getM}.putM.Done$$
$$Difficult := \overline{getH}.putH.Done$$
$$Done := \overline{out}.Jobber$$

A model of the system in the π-calculus.

The π-Calculus: Structural Congruence

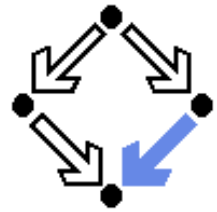


Congruence relation $\equiv$ defined by following equations:

$$
\begin{array}{rcll}
x(y).P & \equiv & x(z).\{z/y\}P & \text{if } z \text{ not free in } P \\
(\text{new } y)\ P & \equiv & (\text{new } z)\ \{z/y\}P & \text{if } z \text{ not free in } P \\
\\
P + Q & \equiv & Q + P & \\
P + (Q + R) & \equiv & (P + Q) + R & \\
\\
P|0 & \equiv & P & \\
P|Q & \equiv & Q|P & \\
P|(Q|R) & \equiv & (P|Q)|R & \\
\\
(\text{new } x)\ (P|Q) & \equiv & P \mid (\text{new } x)\ Q & \text{if } x \text{ not free in } P \\
(\text{new } x)\ 0 & \equiv & 0 & \\
(\text{new } x)(\text{new } y)\ P & \equiv & (\text{new } y)(\text{new } x)\ P & \\
\\
!P & \equiv & P \mid !P &
\end{array}
$$

Processes described by congruent terms are considered indistinguishable.

The π-Calculus: Reactions



Reaction relation $\rightarrow$ that can be derived by the following rules:

COMM $(\overline{x}\langle z\rangle.P + M)|(x(y).Q + N) \rightarrow P|\{z/y\}Q$

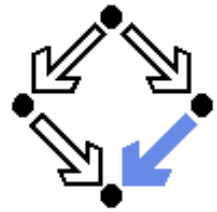
PAR $$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$$

RES $$\frac{P \rightarrow P'}{\text{new } a\ P \rightarrow \text{new } a\ P'}$$

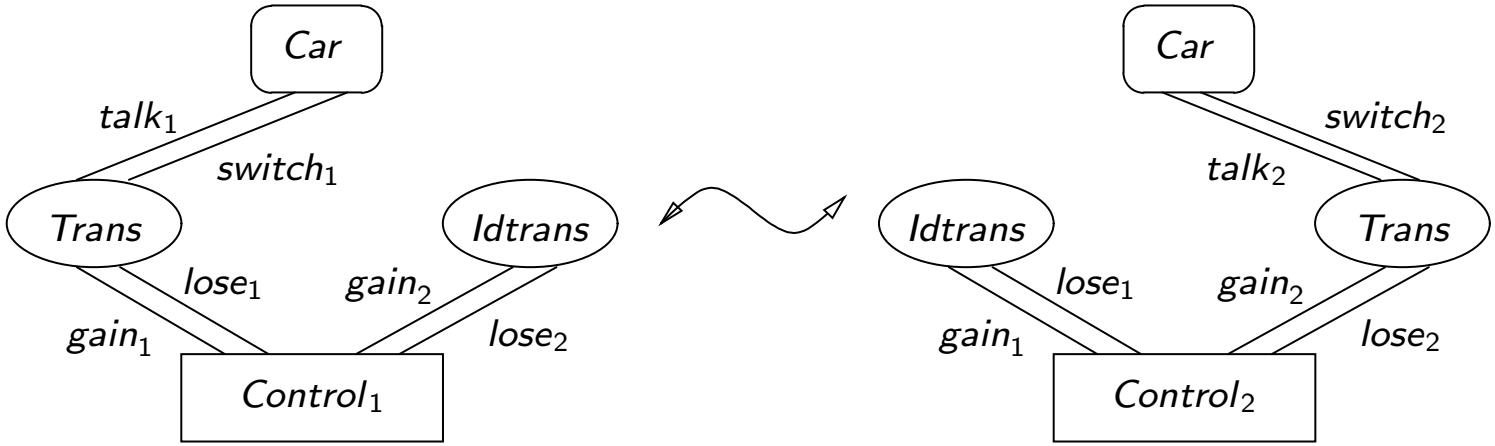
STRUCT $$\frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$$

The reaction relation describes how the individual processes in a system may react with each other yielding a new state of the system.

Example: A Mobile System



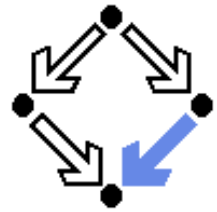
System with one car and two transmitters.



$$
\begin{aligned}
&\mathit{System} := \\
&\quad (\text{new } \mathit{talk}_1, \mathit{switch}_1, \mathit{gain}_1, \mathit{lose}_1, \mathit{talk}_2, \mathit{switch}_2, \mathit{gain}_2, \mathit{lose}_2) \\
&\qquad (\mathit{Car}\langle \mathit{talk}_1, \mathit{switch}_1\rangle | \mathit{Trans}\langle \mathit{talk}_1, \mathit{switch}_1, \mathit{gain}_1, \mathit{lose}_1\rangle | \\
&\qquad \mathit{Idtrans}\langle \mathit{gain}_2, \mathit{lose}_2\rangle | \mathit{Control}_1).
\end{aligned}
$$

Descriptions of car and transmitters parameterized over current links.

Mobility (Contd)



$$Car(talk, switch) := \overline{talk}.Car\langle talk, switch\rangle + switch(t, s).Car\langle t, s\rangle.$$

$$\begin{aligned} Trans(talk, switch, gain, lose) := \\ talk.Trans\langle talk, switch, gain, lose\rangle + \\ lose(t, s).\overline{switch}\langle t, s\rangle.Idtrans\langle gain, lose\rangle. \end{aligned}$$

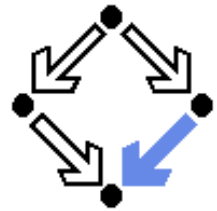
$$Idtrans(gain, lose) := gain(t, s).Trans\langle t, s, gain, lose\rangle.$$

$$Control_1 := \overline{lose_1}\langle talk_2, switch_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle.Control_2.$$

$$Control_2 := \overline{lose_2}\langle talk_1, switch_1\rangle.\overline{gain_1}\langle talk_1, switch_1\rangle.Control_1.$$

Link names ("ports") may be transmitted as messages; received link names may be used for sending messages.

Biological Systems



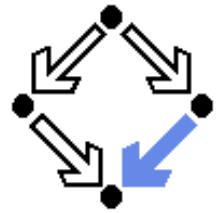
BioSPI project:

We employ 5 major principles in modeling biochemical processes as concurrent systems:

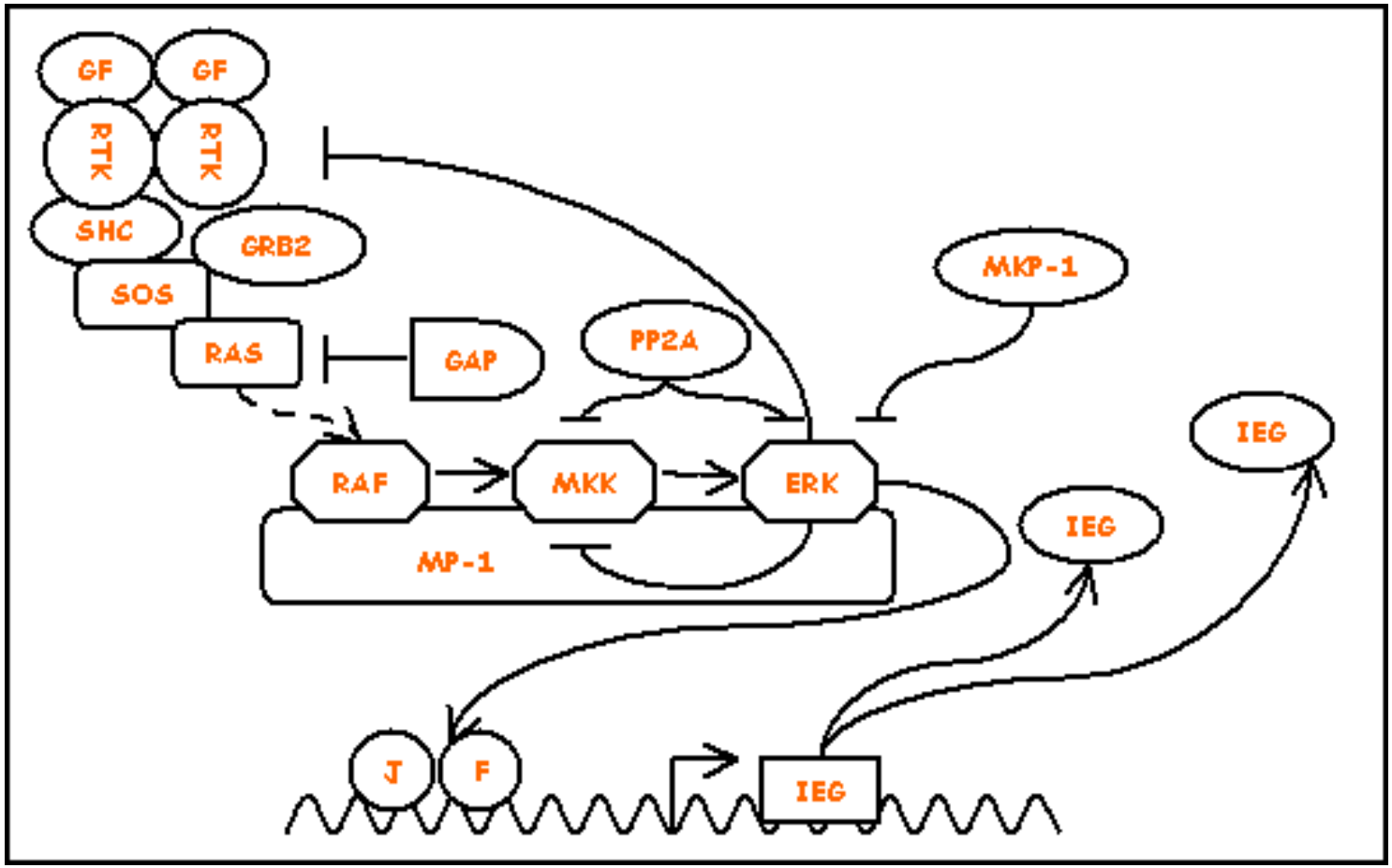
- *Pathways, molecules and molecular domains as computational processes.*
- *Complementary molecular determinants as communication channels.*
- *Molecular interaction and modification as communication and change of channel names.*
- *The integrity of molecules, complexes and compartment as channels with restricted scope.*
- *The formation of complexes and translocation of molecule as extrusion of restricted channels.*

Based on this strong correspondence between the calculus and biochemical networks, we can incrementally represent detailed information on biochemical systems in a structured, biologically faithful fashion. The resulting representations can be used in simulation, analysis and verification.

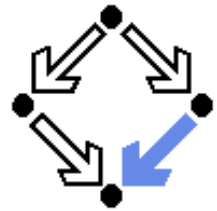
Modeling Signal Transduction Pathways



Regev et al: "Representation and Simulation of Biochemical Processes using the π-calculus Process Algebra", 2001.



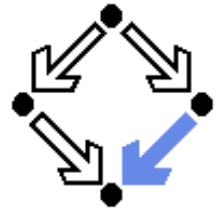
Modeling Biochemical Pathways



A protein ligand molecule (GF), with two identical domains, binds two receptor tyrosine kinase (RTK) molecules on their extracellular part. The bound receptors form a dimeric complex, and cross-phsphorylate and activate the protein tyorisine kinase in their intracellaluar part. . . . Activated ERK1 translocates to the nucleus, where it phosphorylates and activates transcription factors, leading to de novo gene expression.

- RTK-MAPK signal transduction pathway:
  - 14 kinds of proteins.
  - Bind and form complexes, modify certain residues on their counterparts, change their confirmation and activity, and translocate between different cellular compartments.
  - Interactions result in a change in gene expression patterns.
- BioSPI model:
  - 5 molecular processes with 24 different domains and 15 sub-domains.
  - Four compartments (extracellular, membrane, cytoplasm and nucleus).
  - Incremental buildup of concise (250 lines) formal representation.

Sketch of BioSPI Model

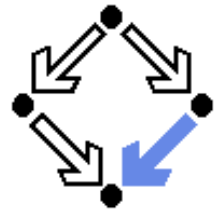


- Pathway is defined as collection of concurrently operating molecules:
  $RtkMapkPathway ::= FreeLigand | RTK | \ldots$
- A protein molecule is composed of several binding domains:
  $FreeLigand := LigandDomain | LigandDomain$
- Interaction may occur between molecules/domains:
  $LigandDomain := \overline{ligandBinding}. \ldots$
  $ExtraDomain := ligandBinding.rtkBinding. \ldots$
- Pathway is composed of *compartments*; only domains in same compartment (e.g. sharing a molecule backbone) may interact:
  $RTK := (\text{new } bb)(ExtraDomain | TransDomain | IntraDomain)$
- Compartments may change during complex formation:
  $Complex := FreeLigand | ExtraDomain | ExtraDomain$
  $FreeLigand := (\text{new } bb)\ (BindingDomain | BindingDomain)$
  $BindingDomain := \overline{ligandBinding}\langle bb \rangle.BoundLigand$
  $ExtraDomain := ligandBinding(cb).BoundExtra$

  FreeLigand is linked with two instances of *ExtraDomain* by *bb*.

π-Calculus used to model formation and modification of compartments.

Simulation of Transduction Pathway

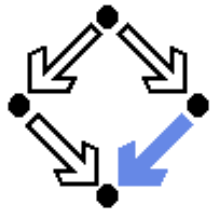


Execute π-calculus model in BioSPI simulator and observe effects of certain perturbations on strength of signal (rate of gene expressions).

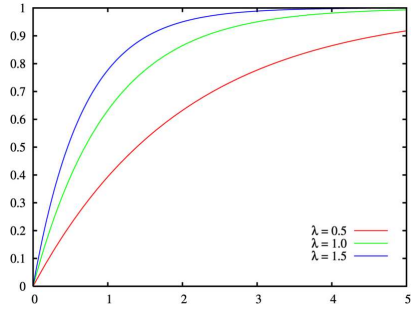
- Modify the quantities of certain molecules:
  - RAF increase $\Rightarrow$ signal increase.
  - MP1 increase $\Rightarrow$ signal decrease.
  - MP1 and MEK increase $\Rightarrow$ signal increase.
  - ...
- Mutate molecules (e.g. insert/delete domains):
  - $RTK := (\text{new } bb)(ExtraDomain | TransDomain) \Rightarrow$ signal decrease.
  - ...

Replace experiments in the laboratory by simulations on the computer.

The Stochastic π-Calculus



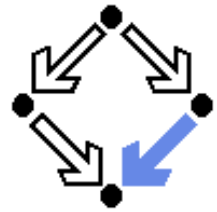
The basic π-calculus is not yet adequate to model the quantitative behavior of biochemical systems (e.g. reaction rates).



- Event rate $r > 0$:
  - The number of events expected per time unit.
  - The parameter of an exponential distribution.
    - Probability that an event occurs within x time units is $1 - e^{-rx}$.
    - Distribution is memoryless: in every fixed time interval of same size, the probability that an event occurs is the same (i.e. independent how often the event has occured in the past).
- Variants of stochastic π-calculus:
  - Original: actions $(x(y), r)$ and $(\overline{x}\langle y\rangle, r)$ with rate r.
  - BioSpi and SPiM: channel x may have a rate r associated.
  - SPiM: also special delay action τ_r with rate r.

Different reactions may "run with different speed".

Example: A Chemical Reaction



- A reaction equation: $\mathrm{Na} + \mathrm{Cl} \leftrightarrow \mathrm{Na}^{+} + \mathrm{Cl}^{-}$

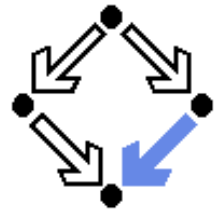
  A salt (NaCL) molecule consists of a Na atom and a Cl atom shared by a "ionic bond": the Na atom gives an electron to the Cl atom such that the resulting ions are coupled by electrostatic attraction; to separate the atoms, the process has to be reversed.

- A basic *pi*-calculus model with four atoms:

  $System := \text{new } e_1, e_2\ (Na \mid Na \mid Cl \mid Cl).$
  $Na := \overline{e_1}\langle\rangle.NaPlus.$
  $NaPlus := e_2().Na.$
  $Cl := e_1().ClMinus.$
  $ClMinus := \overline{e_2}\langle\rangle.Cl.$

Molecules as concurrent systems of atoms that emit and absorb electrons.

Stochastic Models

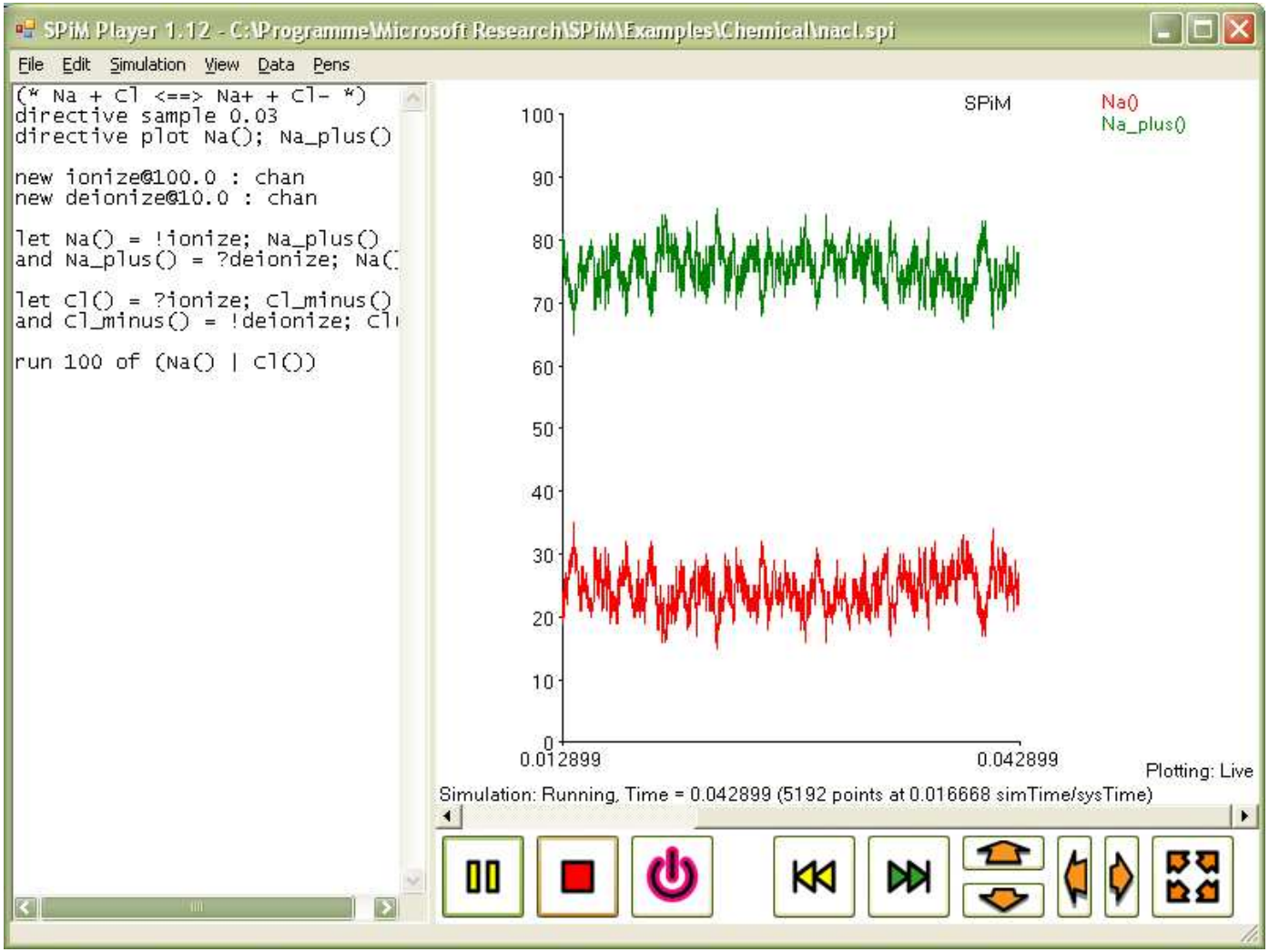


BioSPI model versus SPIM model.

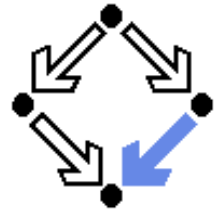
```
% Na + Cl <--> Na+ + Cl-
-language(psifcp).
global(e1(100),e2(10)).

System::= Na | Cl .

Na::= e1 ! [] , Na_plus .
Na_plus::= e2 ? [] , Na .

Cl::= e1 ? [] , Cl_minus .
Cl_minus::= e2 ! [] , Cl .
```

```
(* Na + Cl <==> Na+ + Cl- *)
directive sample 0.03
directive plot Na(); Na_plus()

new ionize@100.0 : chan
new deionize@10.0 : chan

let Na() = !ionize; Na_plus()
and Na_plus() = ?deionize; Na()

let Cl() = ?ionize; Cl_minus()
and Cl_minus() = !deionize; Cl()

run 100 of (Na() | Cl())
```

Simulation in SPiM



Example: Circadian Clock



- Circadian rhythm: (from Wikipedia)

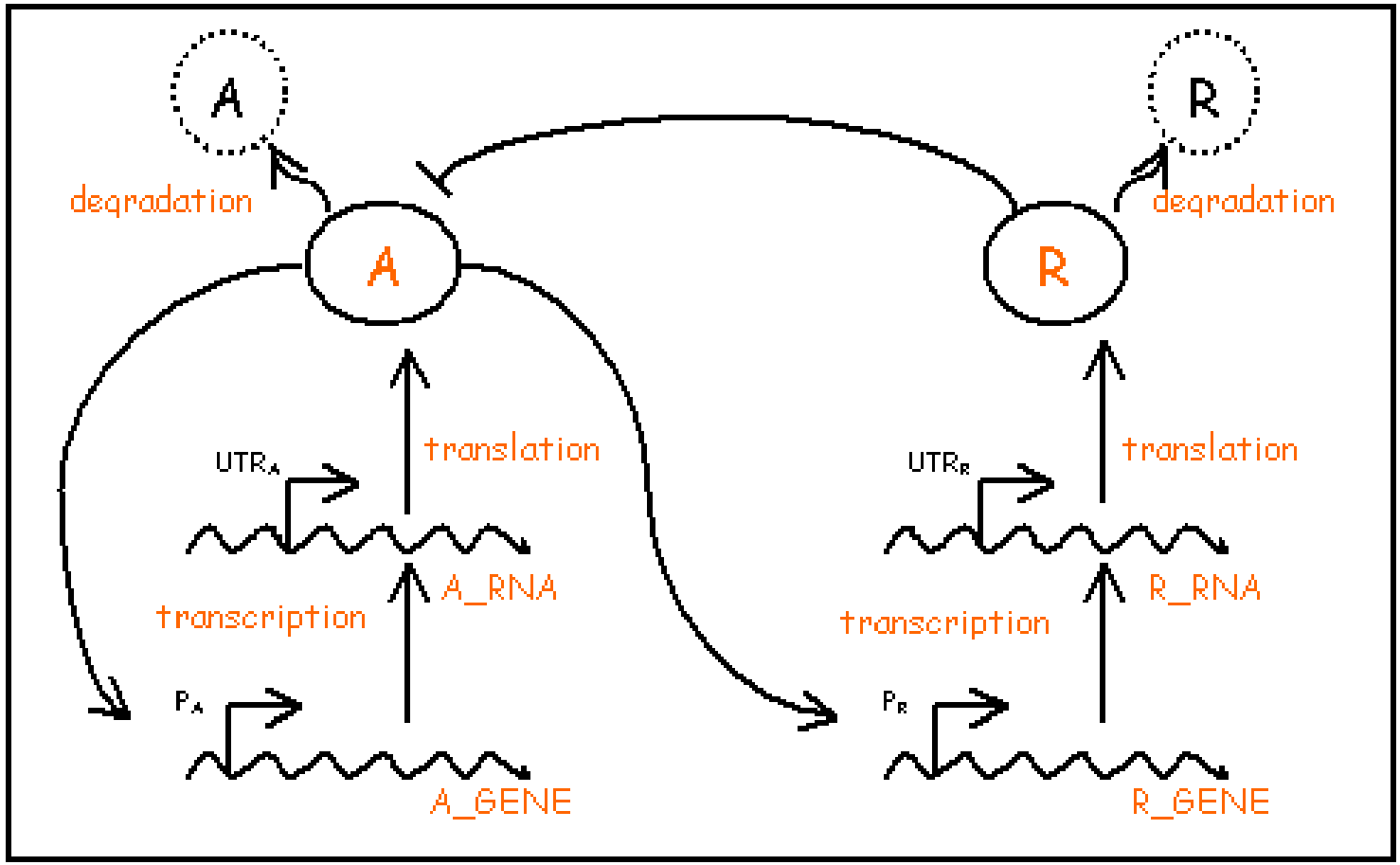
  A circadian rhythm is a roughly-24-hour cycle in the physiological processes of living beings . . . The term "circadian" . . . comes from the Latin circa, "around", and dies, "day", meaning literally "about a day.". In a strict sense, circadian rhythms are endogenously generated . . . The first endogenous circadian oscillation was observed in the 1700s by the French scientist Jean-Jacques d'Ortous de Mairan who noticed that 24-hour patterns in the movement of plant leaves continued even when the plants were isolated from external stimuli.

- Signal transduction pathway: (from BioSPI)

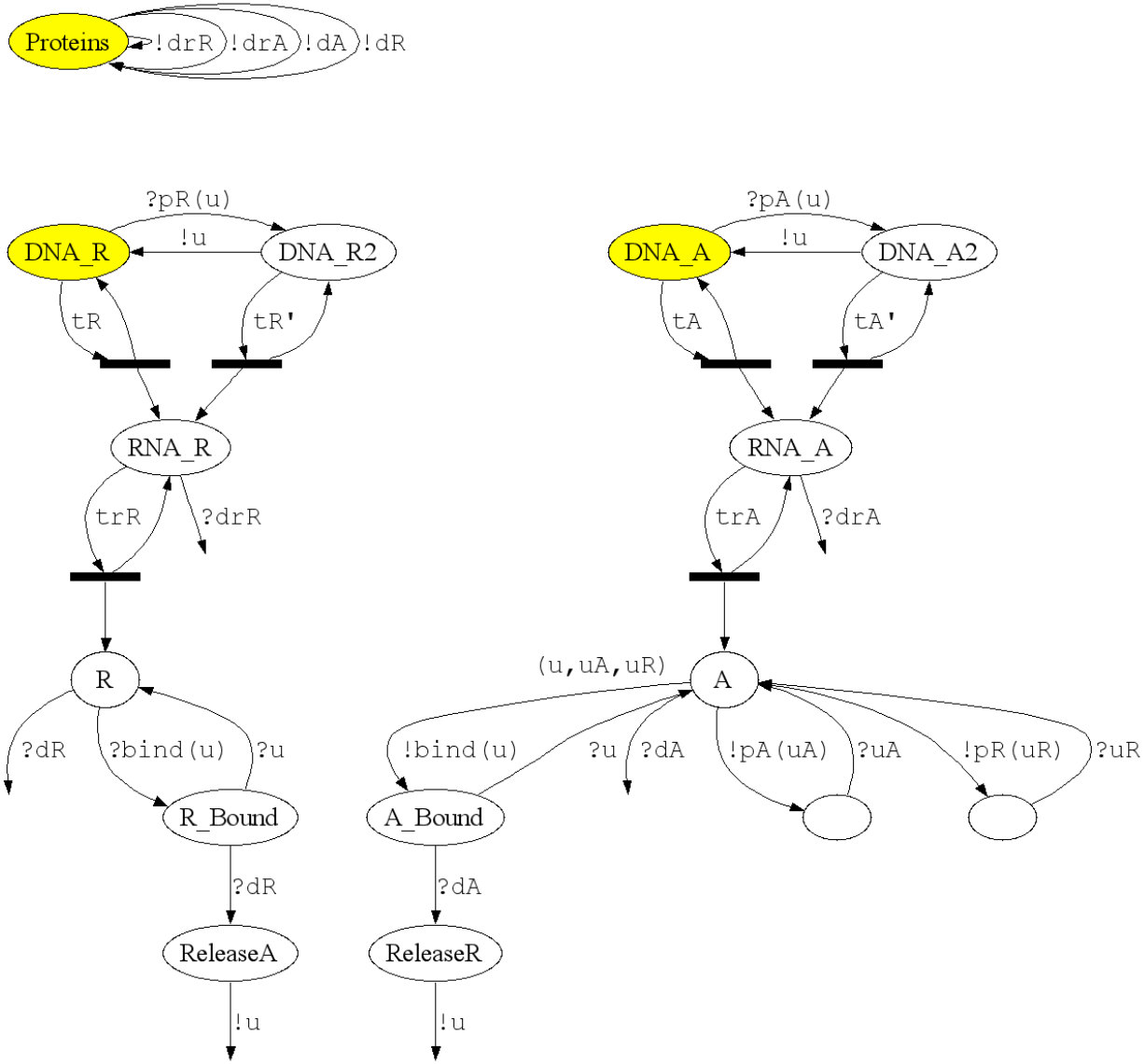
  A positive element, protein A, increases its own expression and that of a negative element, protein R. Strong binding of R to A inhibits A's activity and so represses the expression of both elements by binding to the promoters PA and PR. The autoactivation of A results in a hysteresis: a bi-stable dependence of A concentration on R. Slow kinetics of R then lead to oscillations, which can be described as successive transitions between 'induced' and 'repressed' states.

Physiological rhythms created from interleaving positive and negative feedback loops in the same pathway.

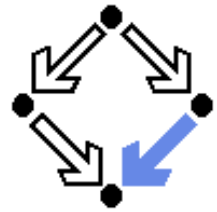
Signal Transduction Pathway



SPiM Model


Proteins
!drR
!drA
!dA
!dR
?pR(u)
DNA_R
!u
DNA_R2
tR
tR'
RNA_R
trR
?drR
R
?dR
?bind(u)
?u
R_Bound
?dR
ReleaseA
!u
?pA(u)
DNA_A
!u
DNA_A2
tA
tA'
RNA_A
trA
?drA
(u,uA,uR)
A
!bind(u)
?u
?dA
!pA(uA)
?uA
!pR(uR)
?uR
A_Bound
?dA
ReleaseR
!u

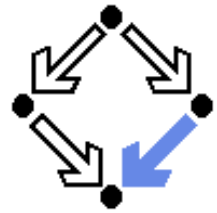
SPiM Model



```
(* Circadian Clock *)
directive sample 800.0 1000
directive plot
?drA as "RNA A"; ?drR as "RNA R";
?dA as "A"; ?dR as "R"

(* Binding Sites *)
new bind@100.0:chan(chan)
new pA@10.0:chan(chan)
new pR@10.0:chan(chan)
new drA@1.0:chan
new dA@0.1:chan
new drR@0.02:chan
new dR@0.01:chan
```

```
let Proteins() =
  do !dA; Proteins()
  or !dR; Proteins()
  or !drA; Proteins()
  or !drR; Proteins()

let DNA_A() = ...
let DNA_R() = ...

run (Proteins() | DNA_A() | DNA_R())
```

SPiM Model



```
val tA = 4.0
val tA' = 40.0
val trA = 1.0
let DNA_A() =
  do delay@tA; (RNA_A() | DNA_A())
  or ?pA(u); DNA_A2(u)
and DNA_A2(u:chan) =
  do delay@tA'; (RNA_A() | DNA_A2(u))
  or !u; DNA_A()
and RNA_A() =
  do delay@trA; (A() | RNA_A())
  or ?drA
and A() = (new u:chan
  new uA@10.0:chan
  new uR@100.0:chan
  do !pA(uA); ?uA; A()
  or !pR(uR); ?uR; A()
  or ?dA
  or !bind(u); A_Bound(u))
and A_Bound(u:chan) =
    do ?dA; ReleaseR(u)
    or ?u; A()
and ReleaseR(u:chan) = !u
```

```
val tR = 0.001
val tR' = 2.0
val trR = 0.1
let DNA_R() =
  do delay@tR; (RNA_R() | DNA_R())
  or ?pR(u); DNA_R2(u)
and DNA_R2(u:chan) =
  do delay@tR'; (RNA_R() | DNA_R2(u))
  or !u; DNA_R()
and RNA_R() =
  do delay@trR; (R() | RNA_R())
  or ?drR
and R() =
  do ?dR
  or ?bind(u); R_Bound(u)
and R_Bound(u:chan) =
    do ?u; R()
    or ?dR; ReleaseA(u)
and ReleaseA(u:chan) = !u
```

SPiM Simulation