Using BlenX for Systems Biology

Corrado Priami
CoSBi

Outline of the talk

1. Systems biology
2. BlenX genesis
3. Computational tools
4. A couple of examples

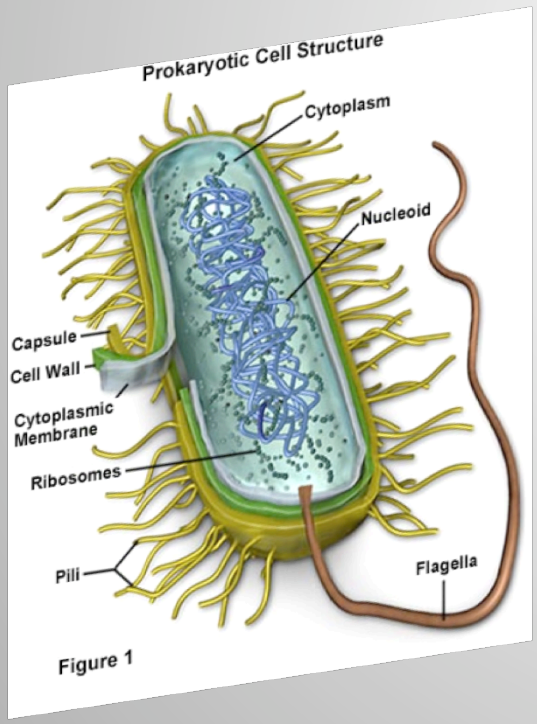
Systems Biology: some challenges

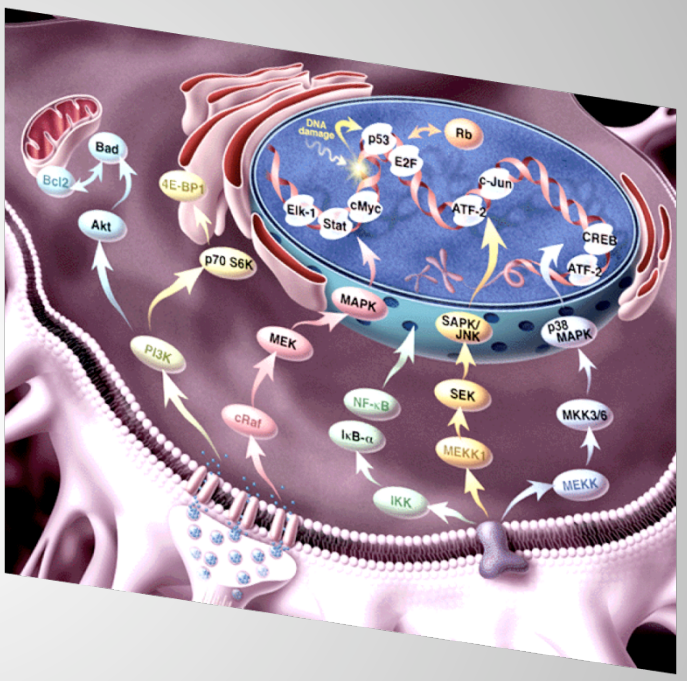
Interaction
Emergence
Multi-level, multi-scale
Partial knowledge
Ambiguous observations

Low-level local mechanisms affect high-level global behavior

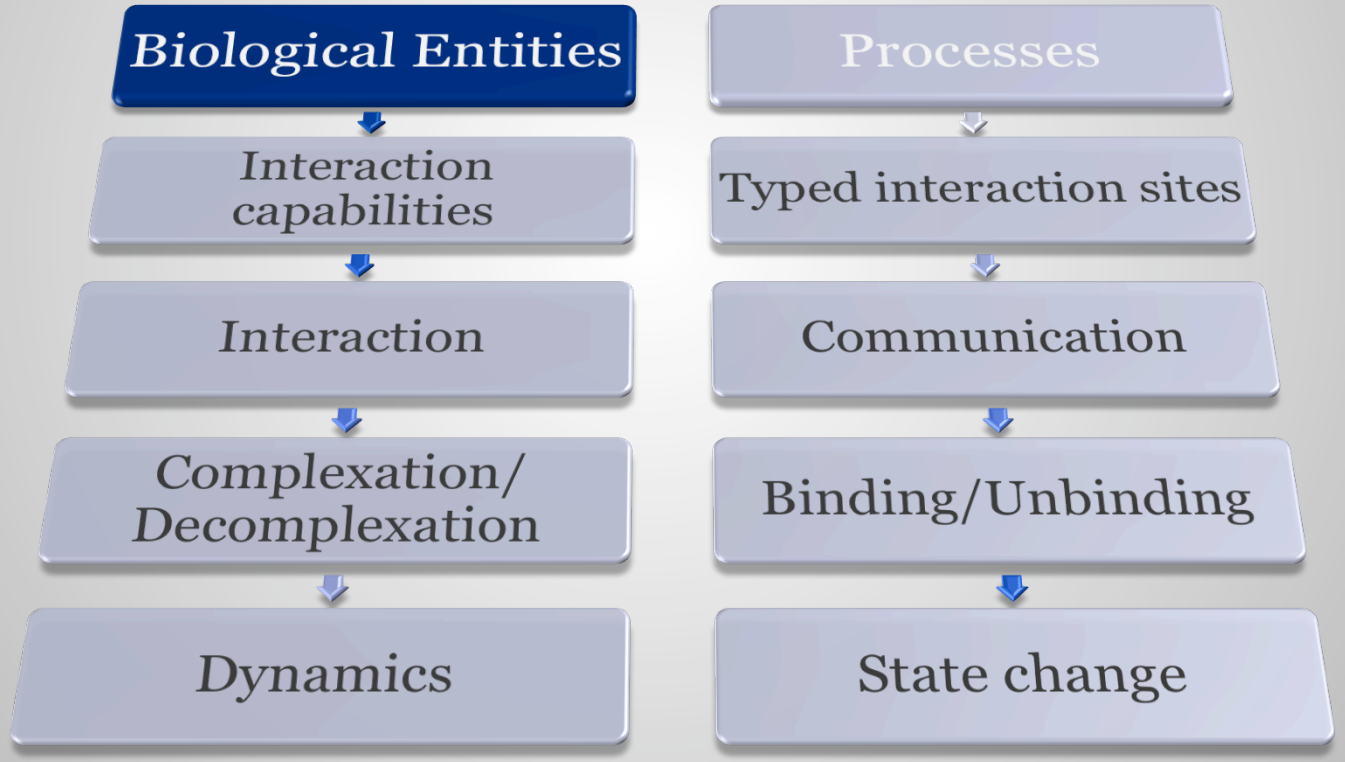
The conceptual framework

A molecular machinery

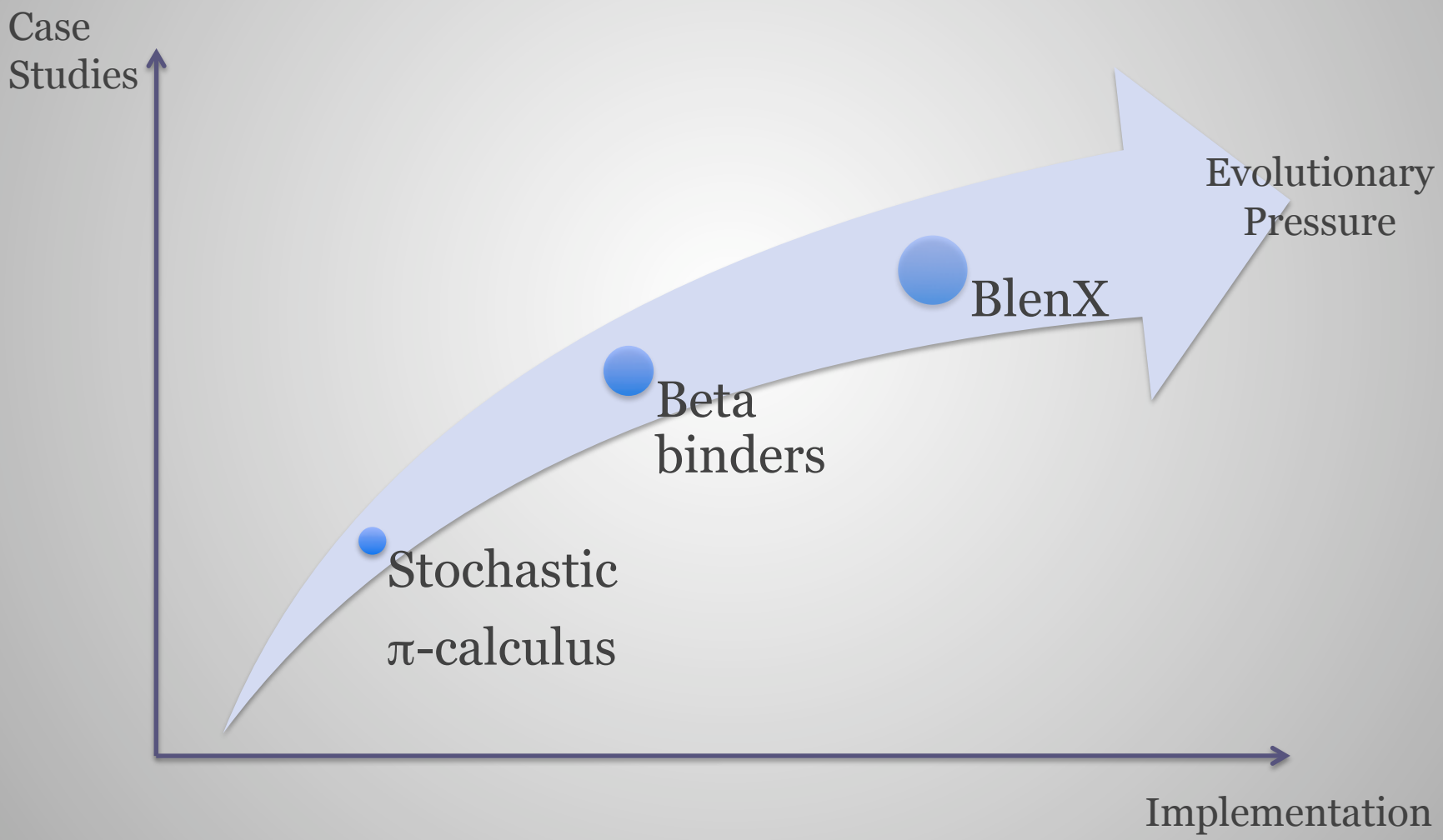




The metaphor



BlenX genesis



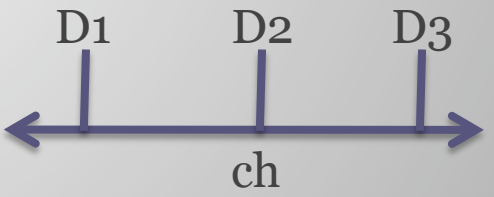
The origin: Stochastic π-calculus

act ::= (a!y, r) | (a?x, r) | (tau, r)

P ::= 0 | Σ_i $act_i.P_i$ | P1|P2 | (new x)P | [x=y]P |!P

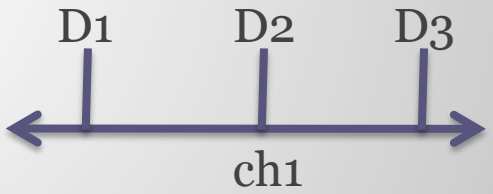
Stochastic π-calculus: proteins

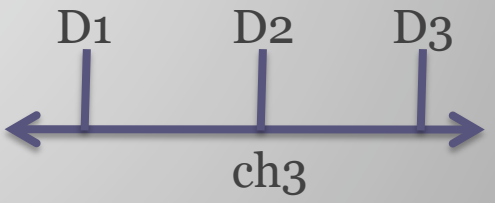
Prot A ::= (new ch)(D1 | D2 | D3)

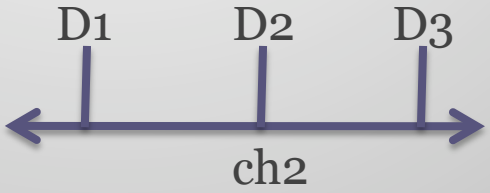


Stochastic π-calculus: concentration of proteins

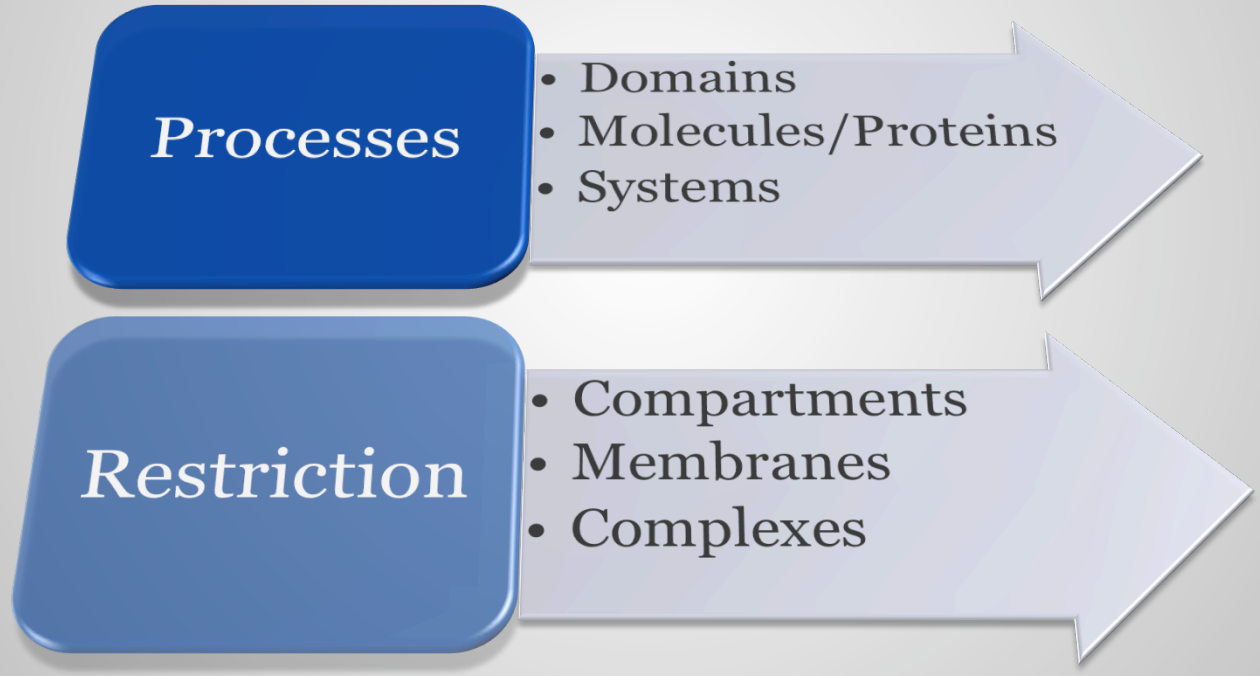
MultiProt A ::= Prot A | … | Prot A







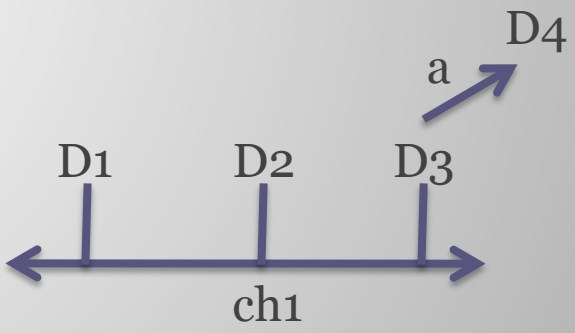
Stochastic π-calculus: 1st concern



Stochastic π-calculus: protein-protein interaction capabilities

Prot A ::= (new ch)(D1 | D2 | a!y.D3')
Prot B ::= a?x.D4'

System ::= Prot A | Prot B



Stochastic π-calculus: protein-protein interaction

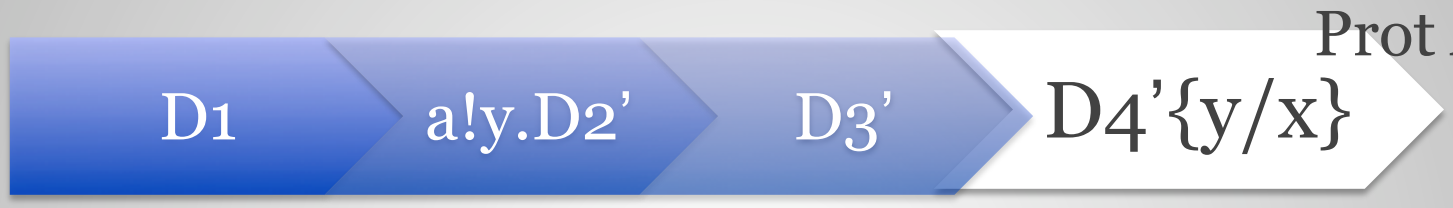
System ➔ (new ch)(D1 | D2 | D3') | D4' {y/x}

Stochastic π-calculus: protein-protein complexation

(new ch, **y**)(D1 | D2 | **a!y**.D3') | **a?x**.D4'

➔ **(new y)[**(new ch)(D1 | D2 | D3') | D4' {y/x}**]**

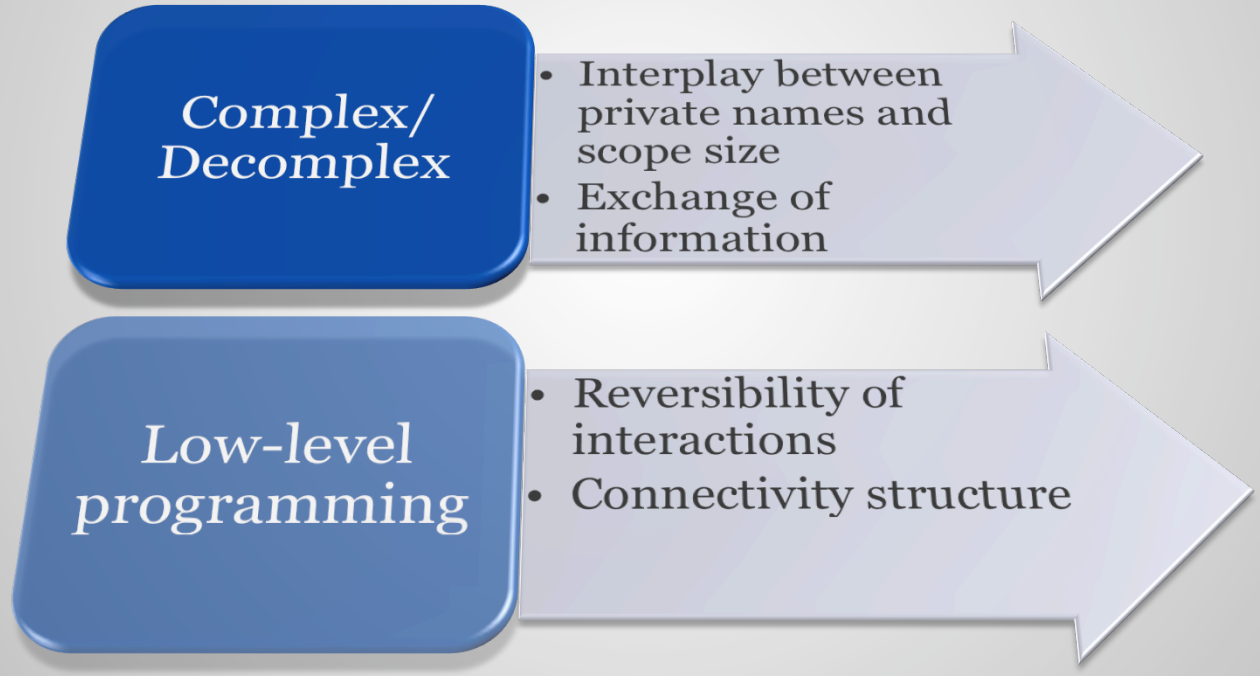
Stochastic π-calculus: protein-protein decomplexation



(new y)[(new ch)(D1 | **a!y**.D2' | D3') | D4' {y/x}]

➔ (new ch)(D1 | D2' | D3') | D4' {y/x}

Stochastic π-calculus: 2nd concern

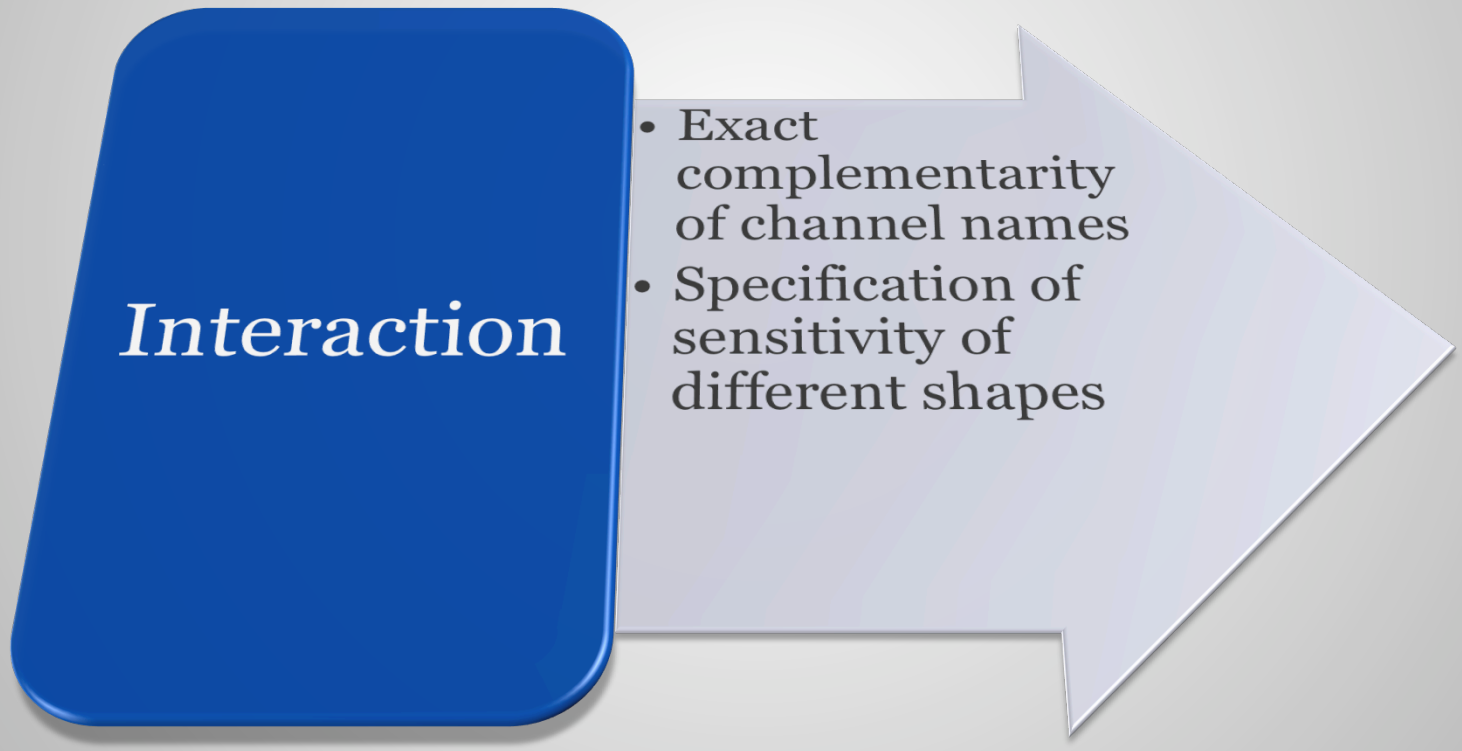


Stochastic π-calculus: non key-lock interaction

(new ch)(D1 | D2 | **a_1!y.D3'+…+a_n!y.D3'**) | a_i?x.D5'

➔ (new ch)(D1 | D2 | D3') | D5' {y/x}

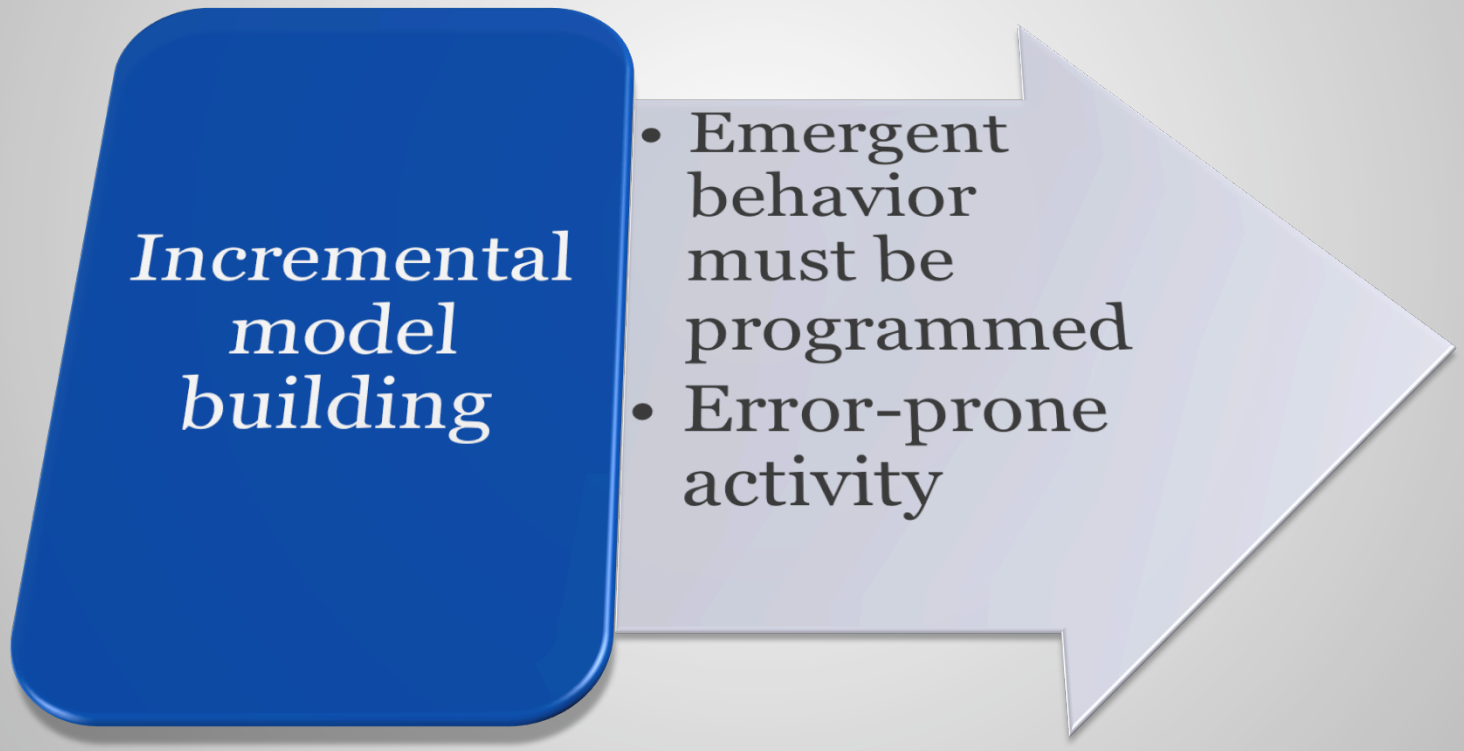
Stochastic π-calculus: 3rd concern



Stochastic π-calculus: partial knowledge

New knowledge : new programming

The origin: 4th concern



?

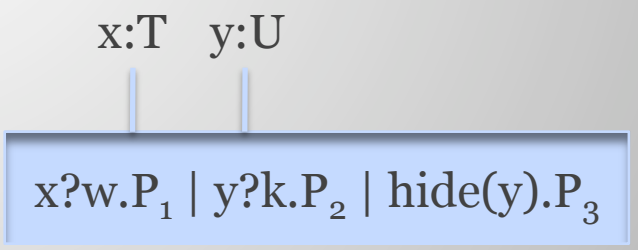
The evolution: Beta-binders

act ::= (a!y, r) | (a?x, r) | (tau, r) | (expose(x, T), r) | (hide(x), r) | (unhide(x), r)

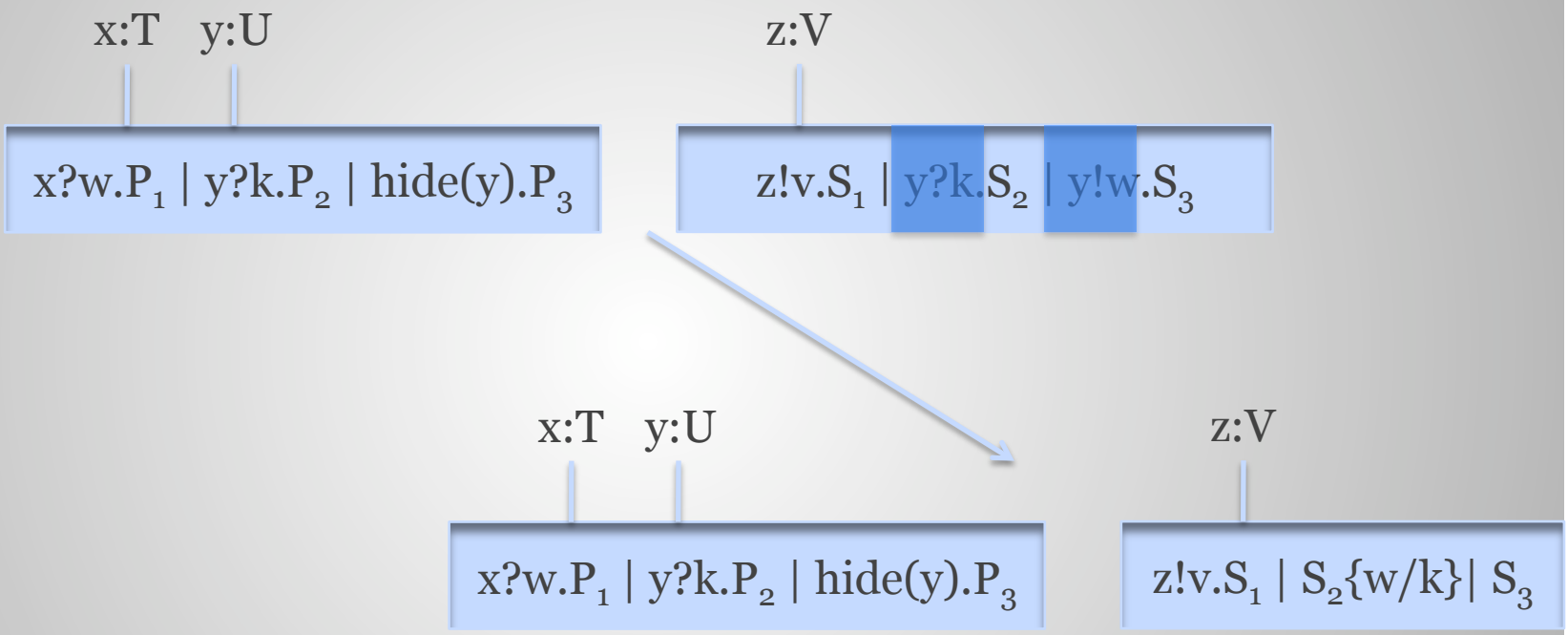
$\mathcal{B}$ ::= $\beta(x:T) \mid \beta^h(x:T) \mid \beta(x:T)\mathcal{B} \mid \beta^h(x:T)\mathcal{B}$

P ::= 0 | P1|P2 | (new x)P |!P

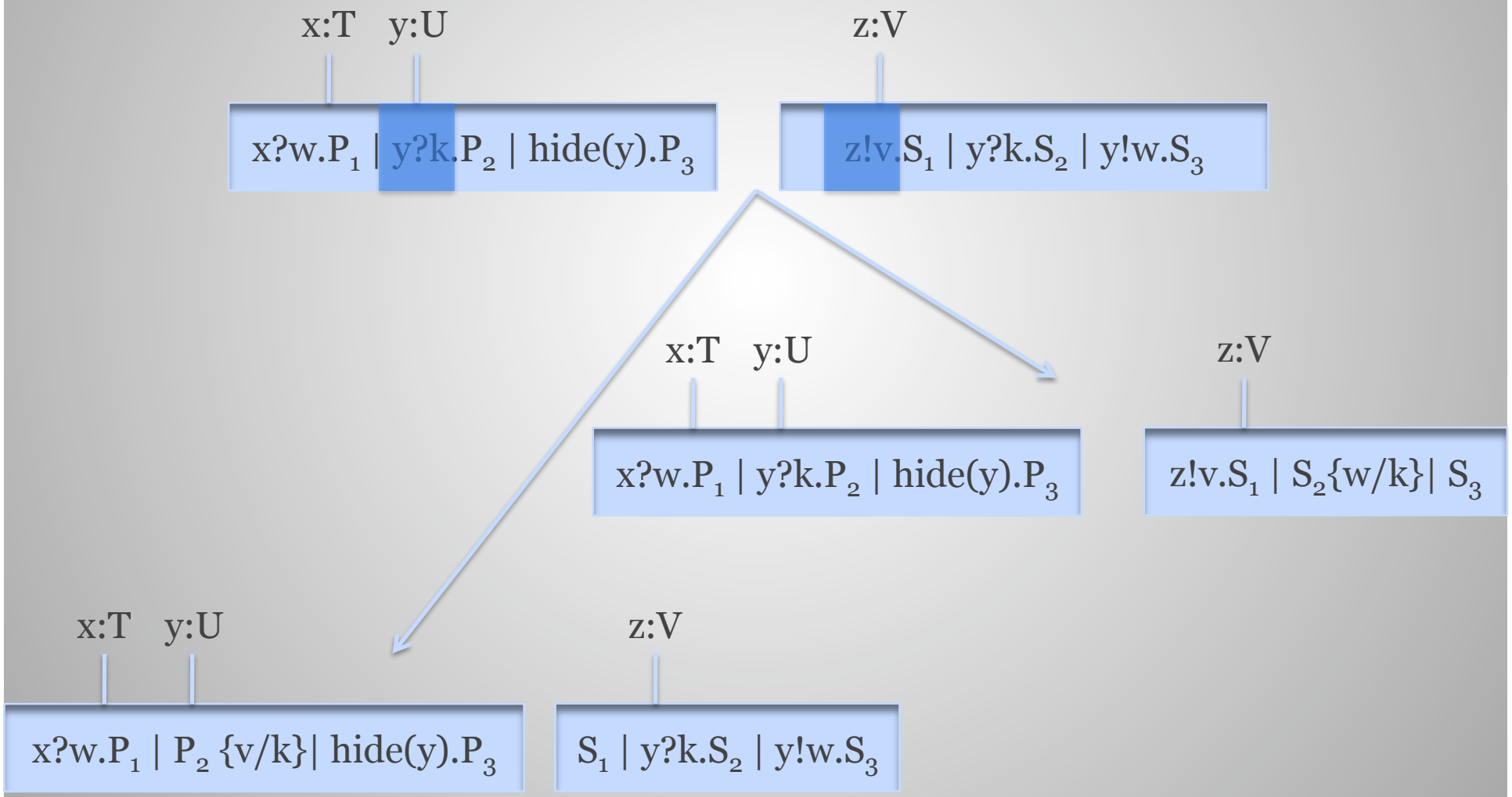
B ::= Nil | $\mathcal{B}[P]$| B|B



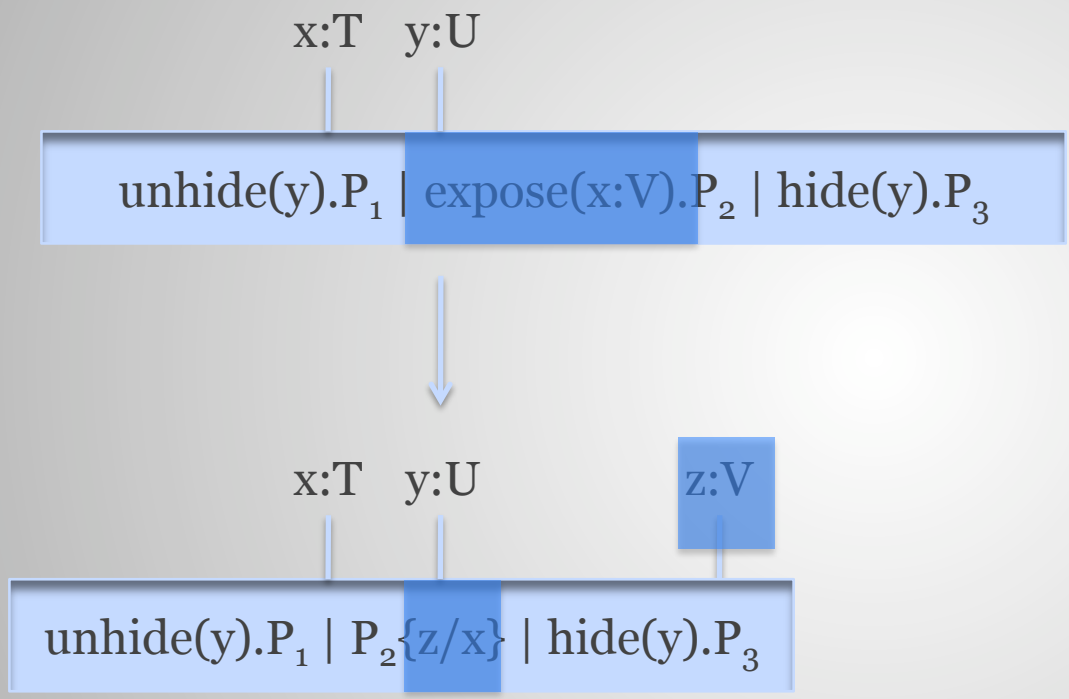
Beta-binders: interaction



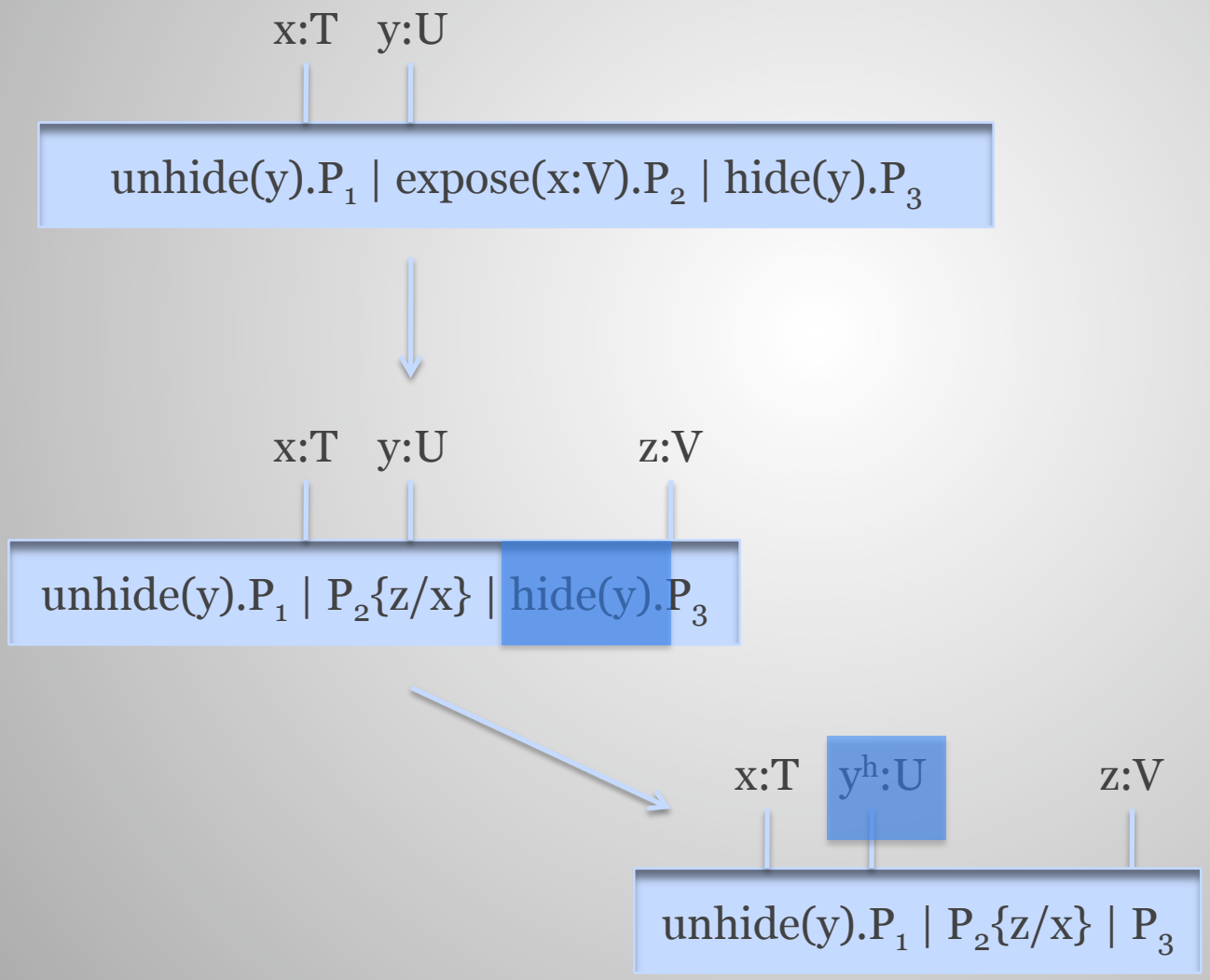
Beta-binders: interaction



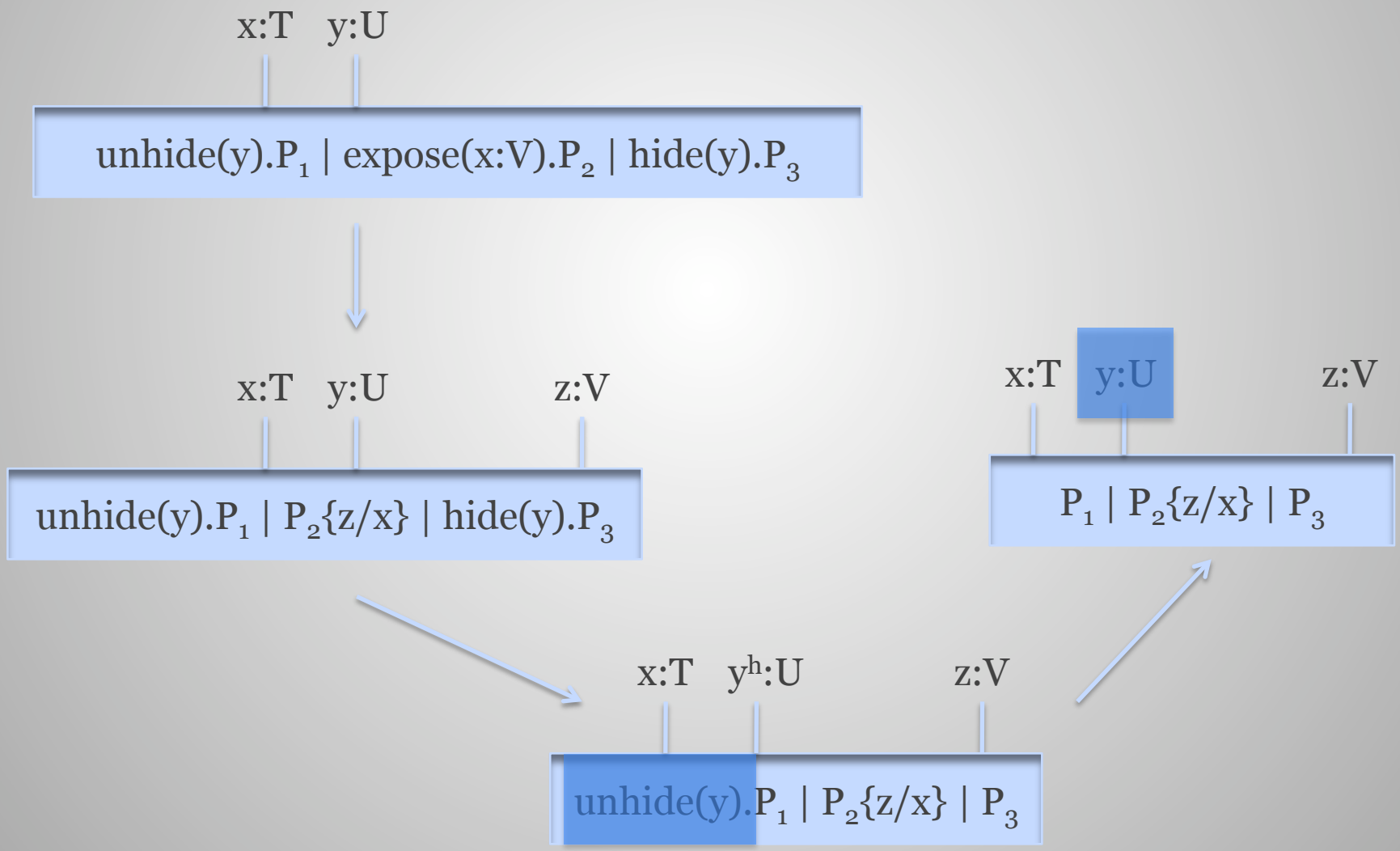
Beta-binders: binders manipulation



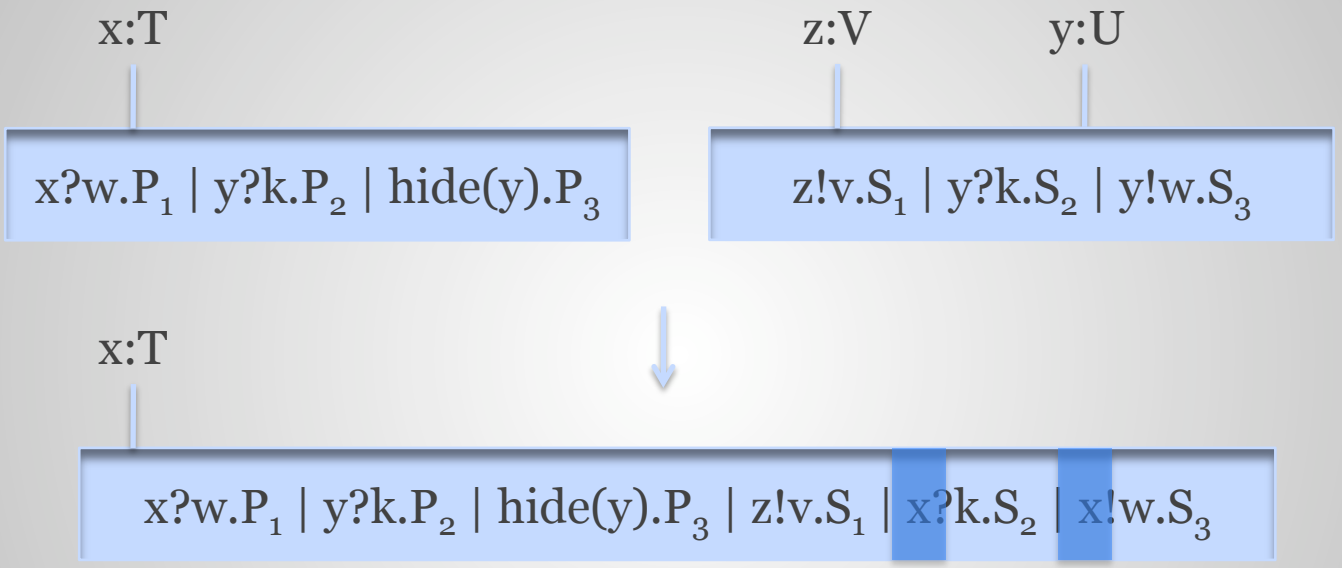
Beta-binders: binders manipulation



Beta-binders: binders manipulation



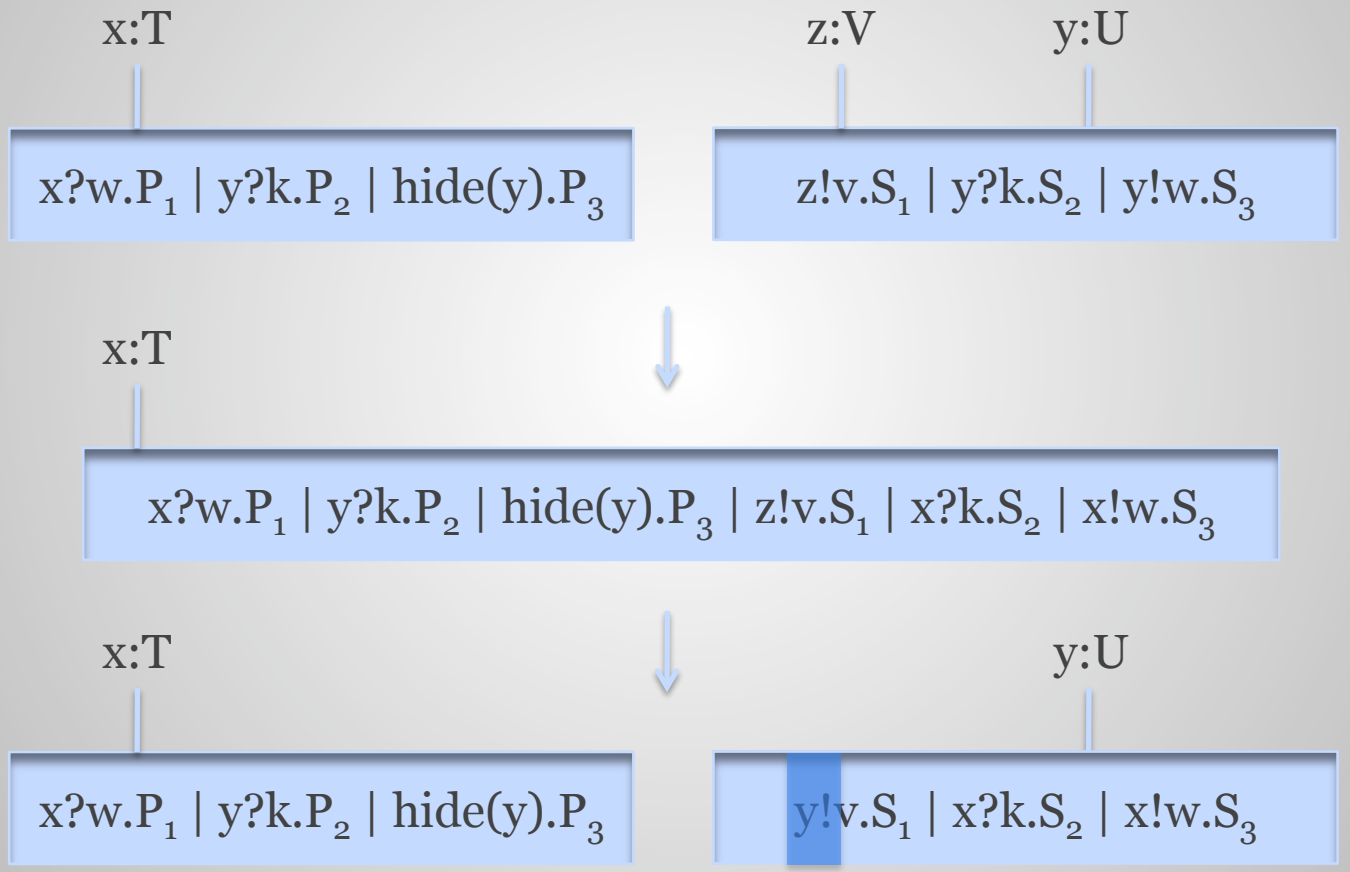
Beta-binders: boxes manipulation



$f_{join} = \lambda \mathcal{B}_1 \mathcal{B}_2 Q_1 Q_2$. **if** $[\mathcal{B}_1 = \beta(x{:}T)\mathcal{B}_1^*$ and $\mathcal{B}_2 = \beta(y{:}U)\mathcal{B}_2^*$ **and** *comp*$(T,U)]$

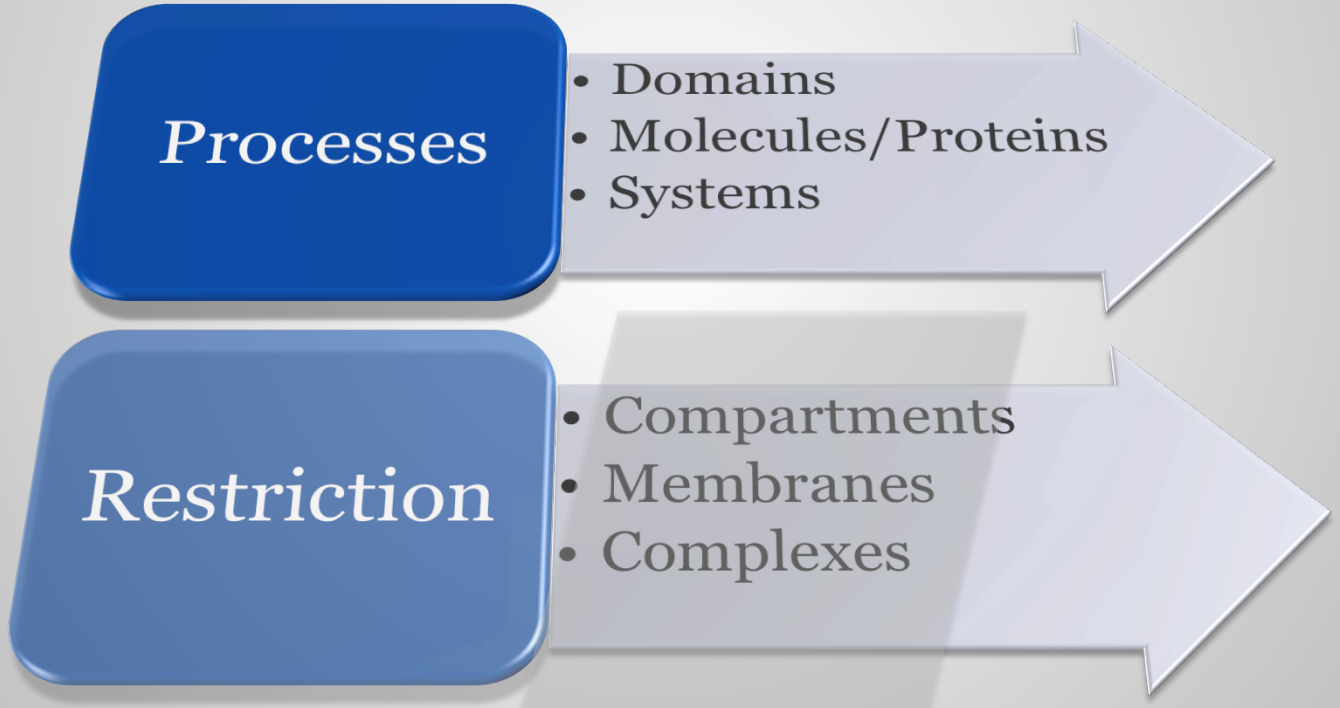
then $(\mathcal{B}_1, \sigma_{id}, \{x/y\})$ **else** nothing

Beta-binders: boxes manipulation

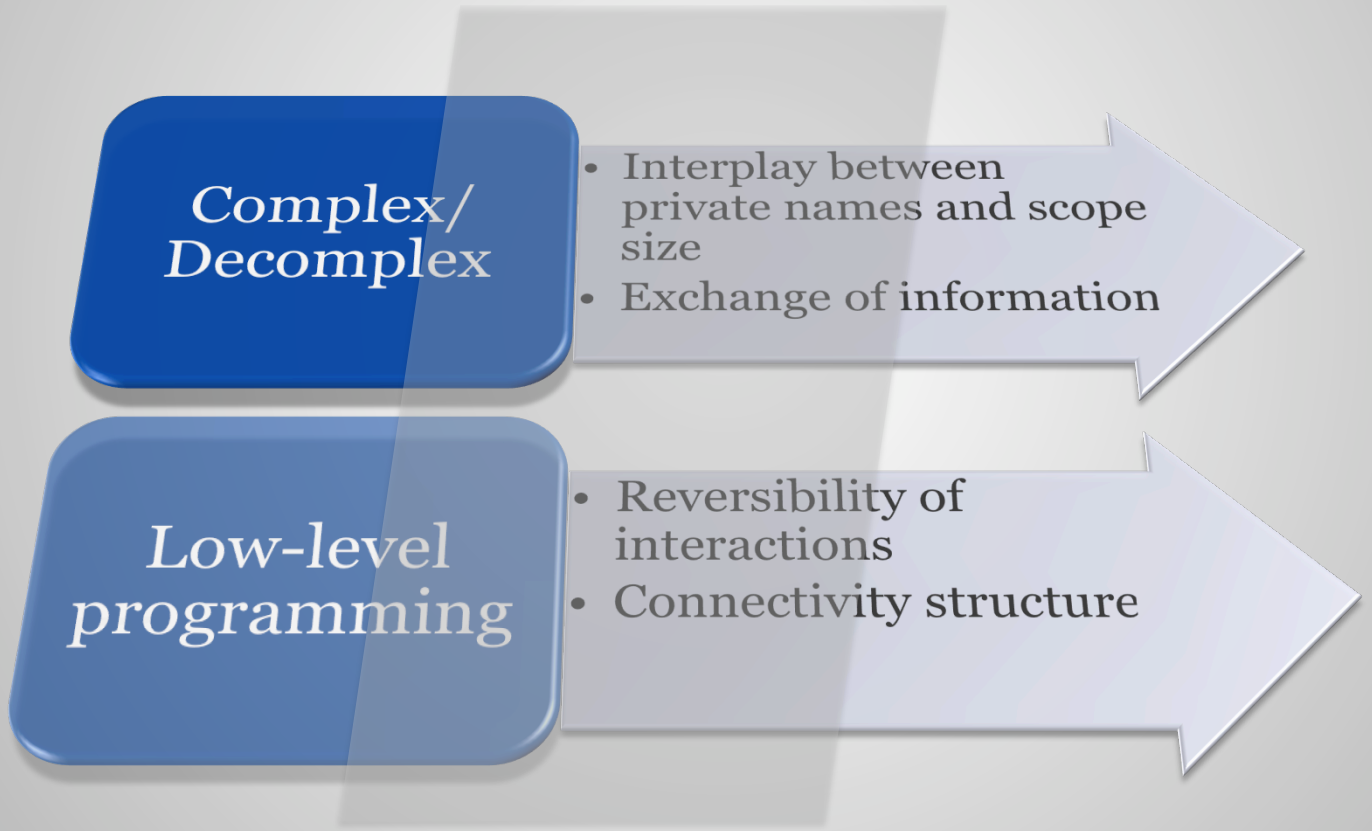


$f_{split} = \lambda \mathcal{B}_1 Q_1 Q_2$. **if** $\mathcal{B}_1 = \beta(x{:}T)\mathcal{B}_1^*$ **then** $(\mathcal{B}_1, \beta(y{:}U), \sigma_{id}, \{y/z\})$ **else** nothing

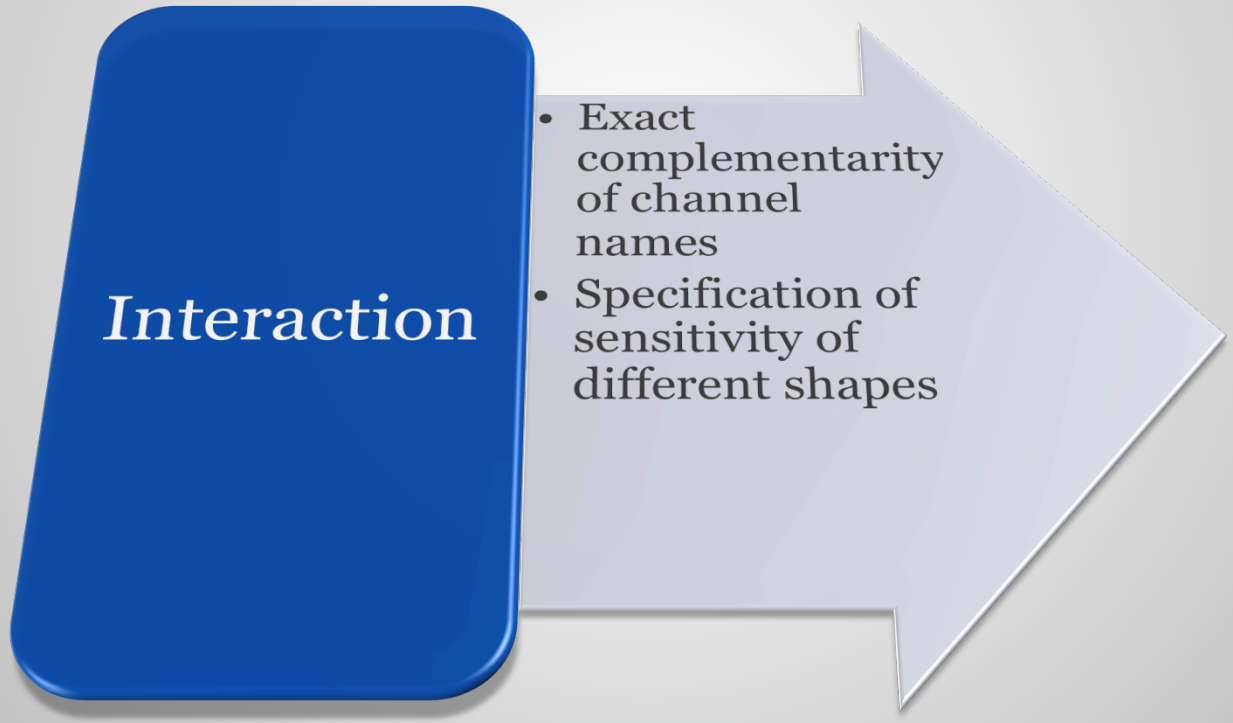
Beta-binders: 1st concern



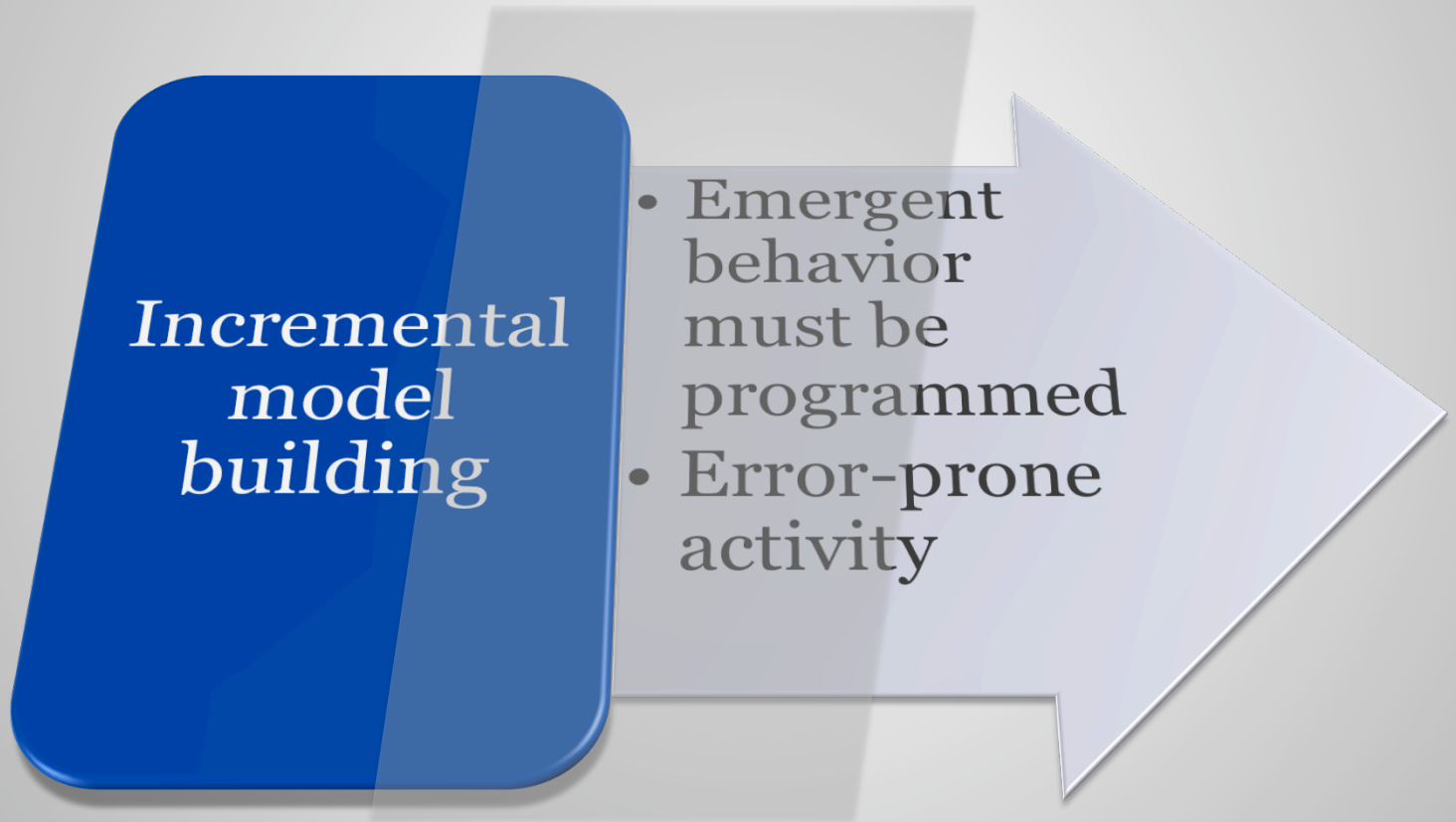
Beta-binders: 2nd concern



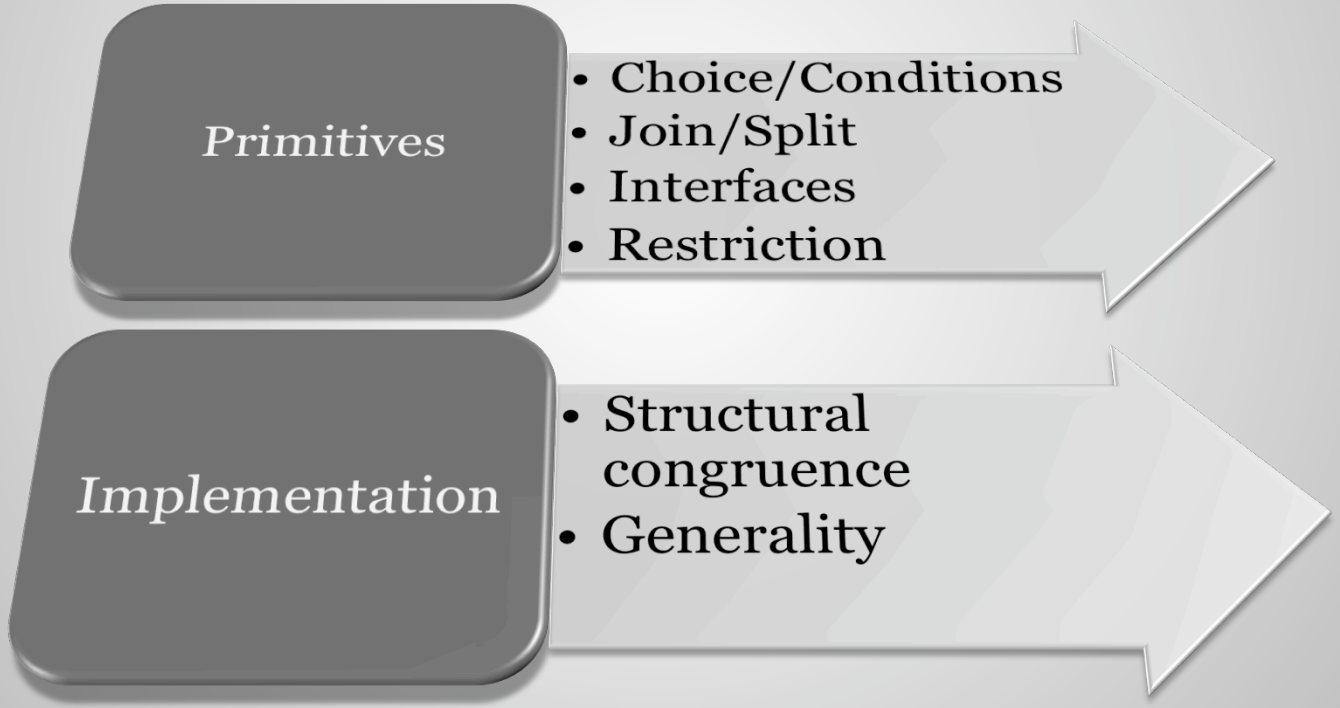
Beta-binders: 3rd concern



Beta-binders: 4th concern



Beta-binders: further concerns



?

The current generation: BlenX

act ::= a!y | a?x | expose(x, T) | hide(x) | unhide(x) | ch(x,T) | delay(r) | die(r)

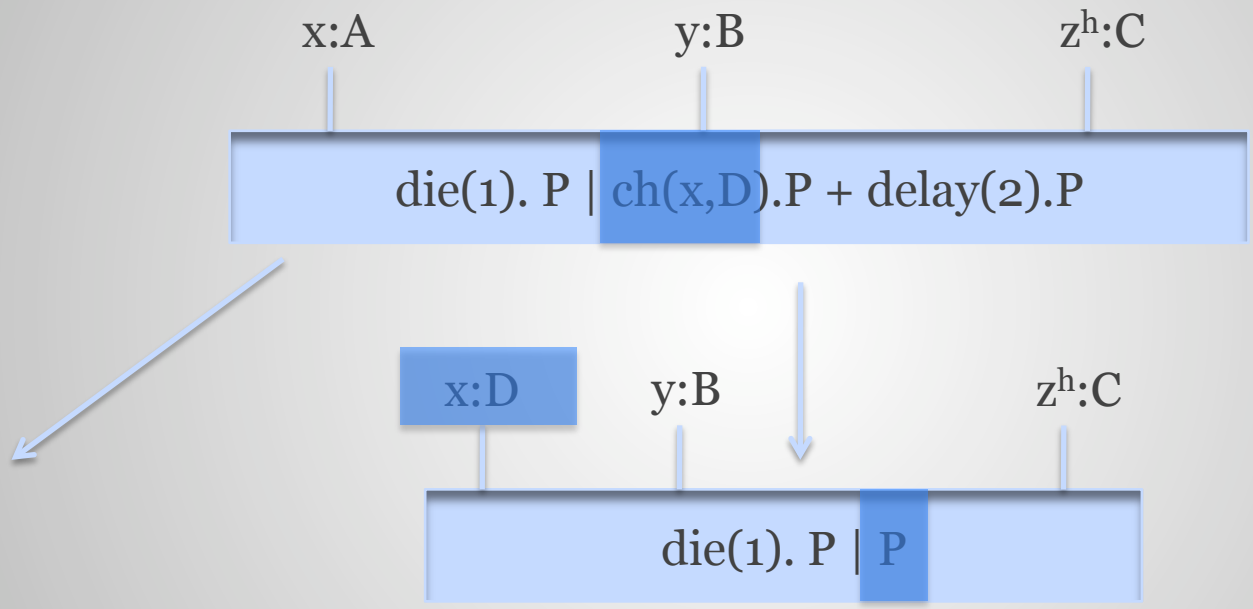
$\mathcal{B}$ ::= $\beta(x{:}T) \mid \beta^h(x{:}T) \mid \beta^c(x{:}T) \mid \beta(x{:}T)\mathcal{B} \mid \beta^h(x{:}T)\mathcal{B} \mid \beta^c(x{:}T)\mathcal{B}$

P ::= 0 | P1|P2 |!act.P | M | if CE then P
M ::= act.P | M + M
CE ::= (Id,Id) | (Id, hidden) | (Id, unhidden) | (Id, Bound) | CE and CE | CE or CE | not CE

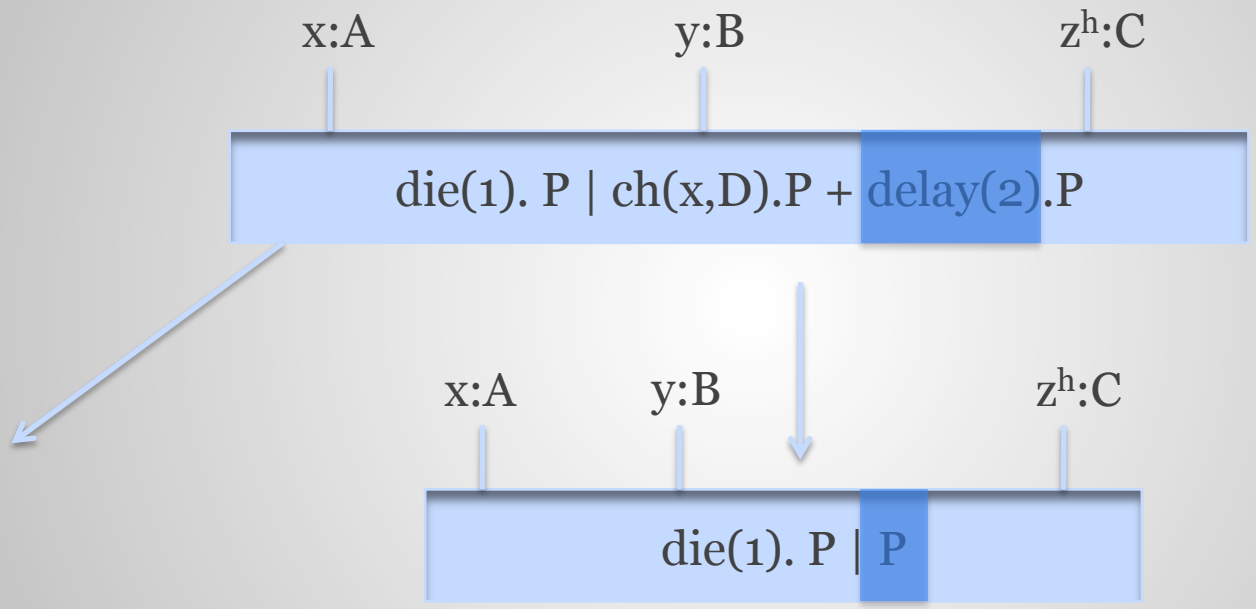
B ::= Nil | $\mathcal{B}$[P]| B|B

E ::= when (C) V
C ::= time = real | steps = decimal | |Id::r | relop Decimal |C and C | C or C | not C
V ::= join(B) | split(B_1,B_2) | delete(n) | new(n) | update(var, fun)

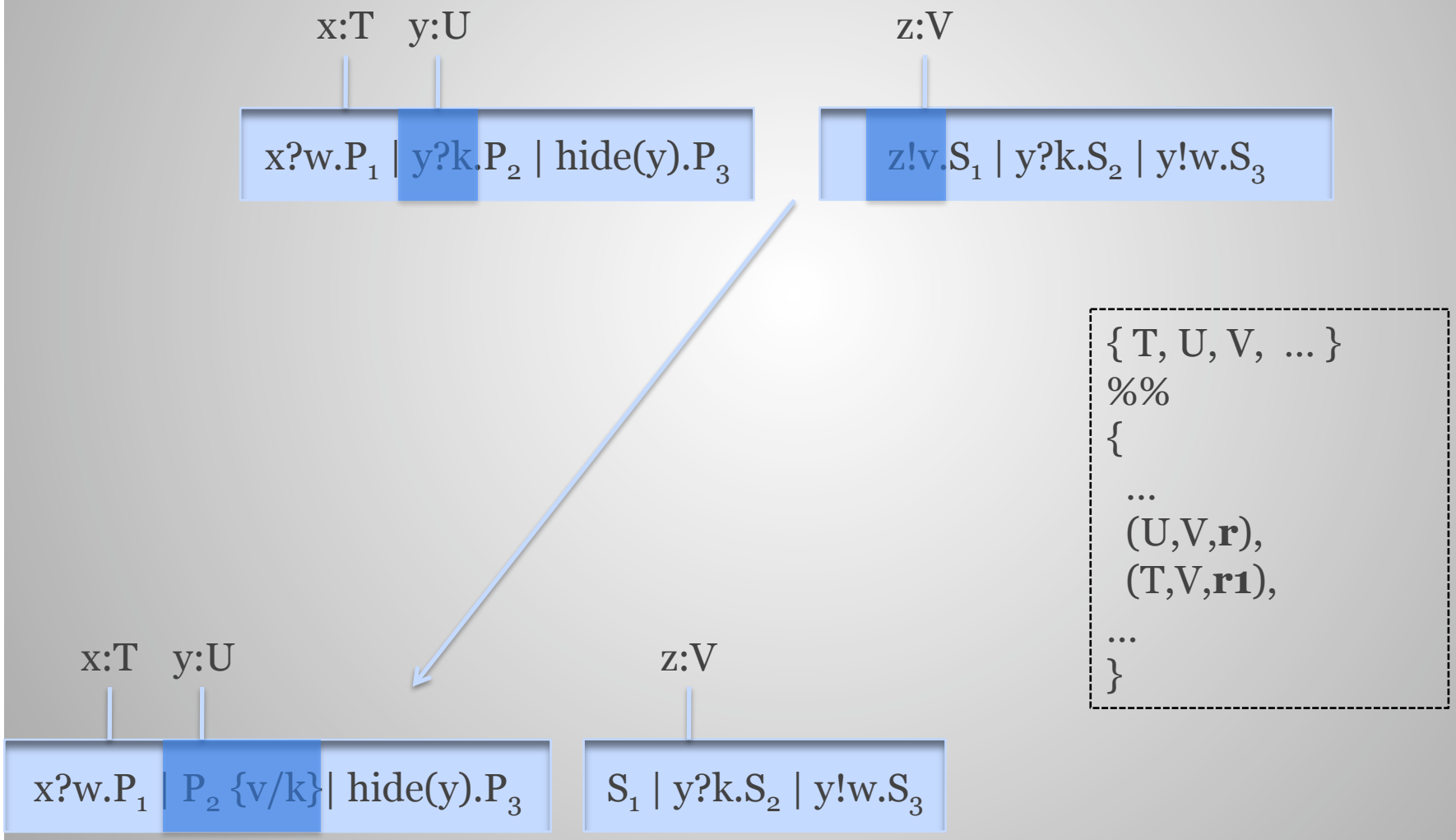
BlenX: monomolecular actions



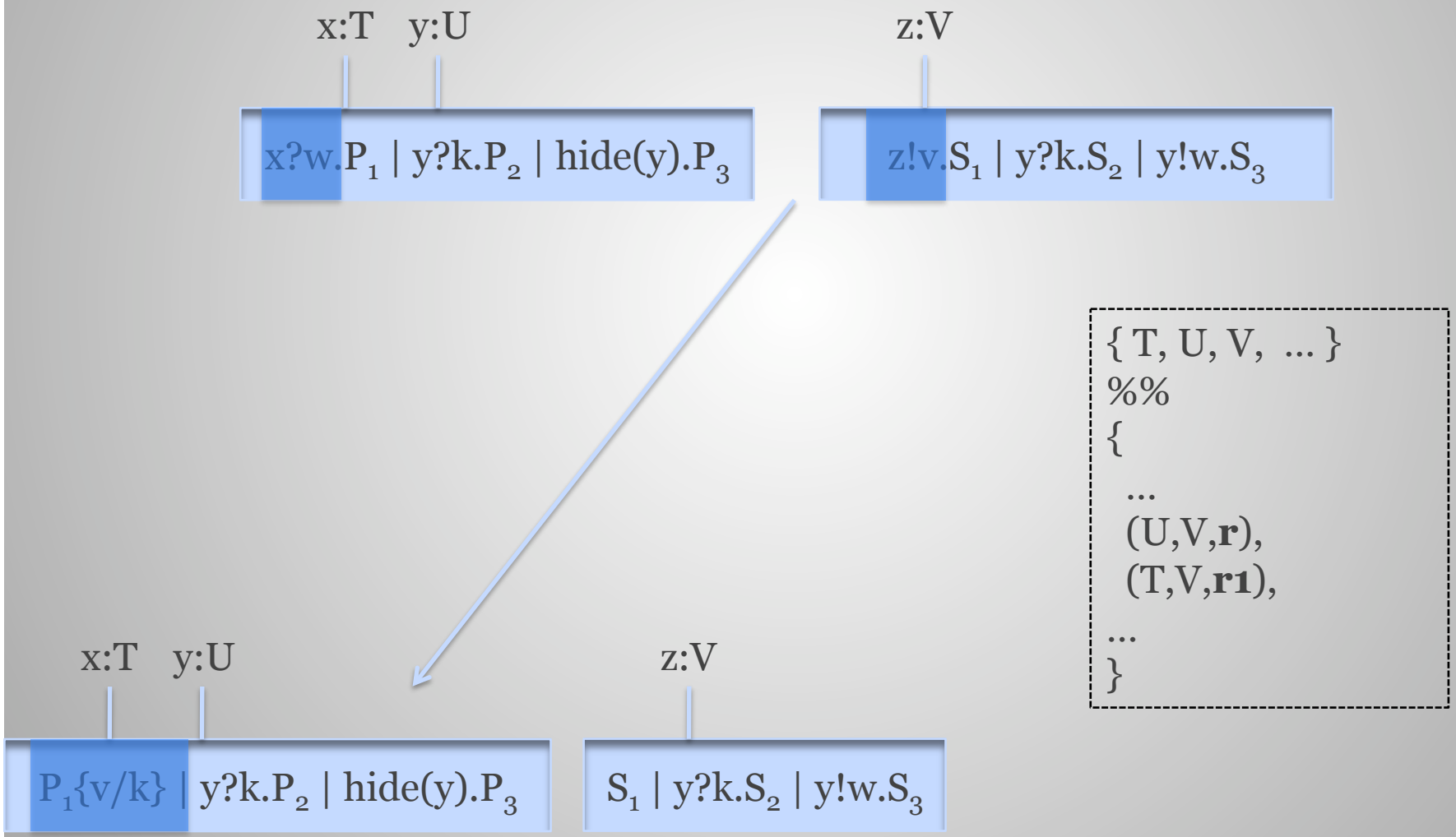
BlenX: monomolecular actions



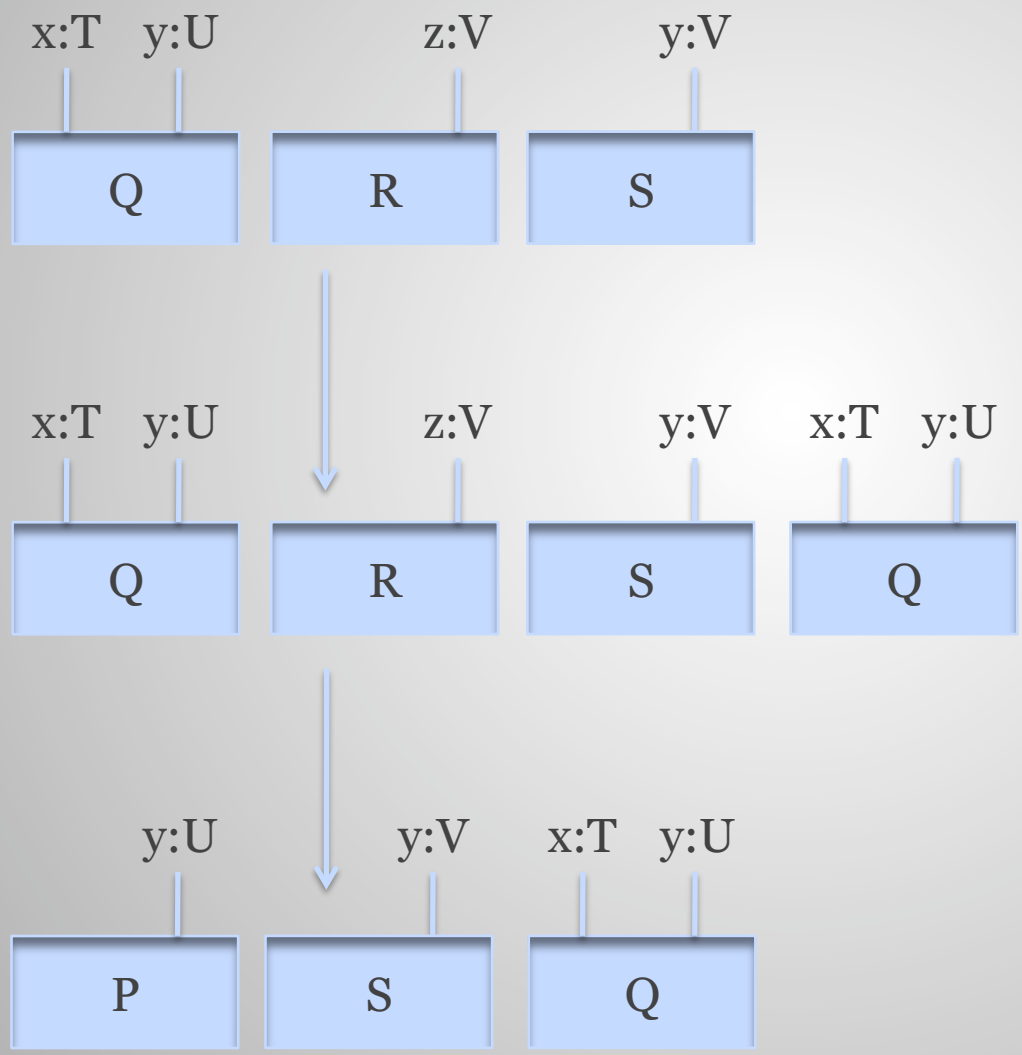
BlenX: bimolecular actions



BlenX: bimolecular actions



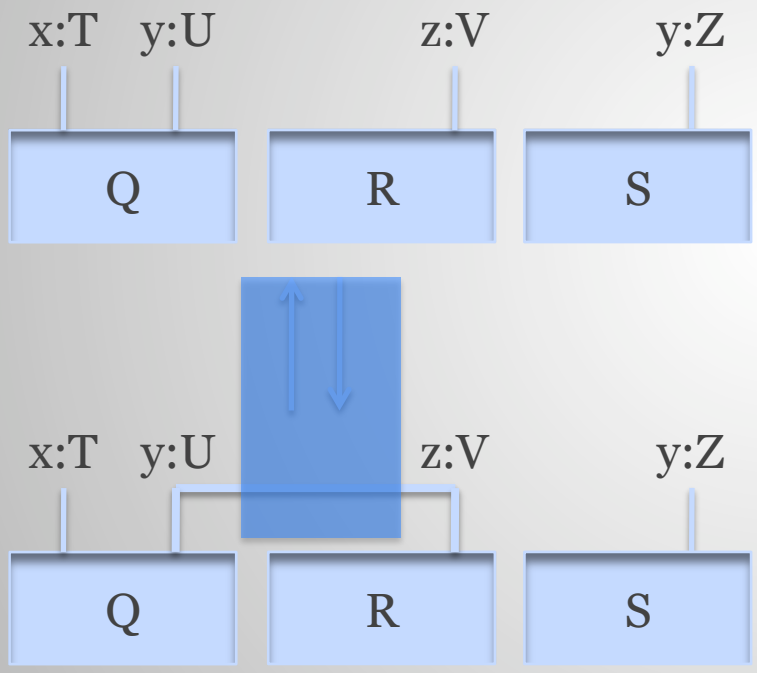
BlenX: events



when (Q :: 10) **new** (1);

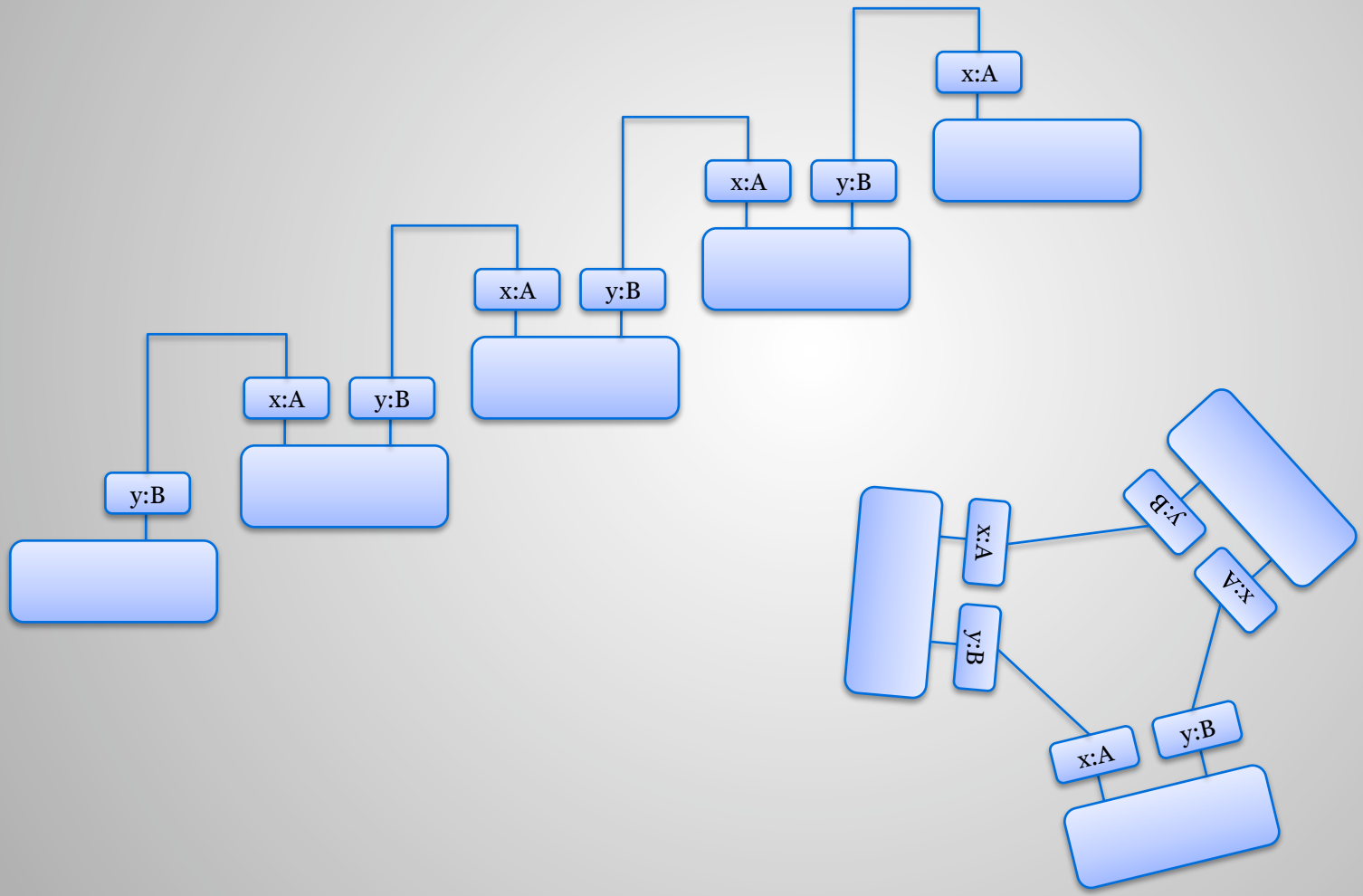
when (R , Q :: f) **join** (P);
let f = |S|*sqrt(|Q|)/k

BlenX: complexes

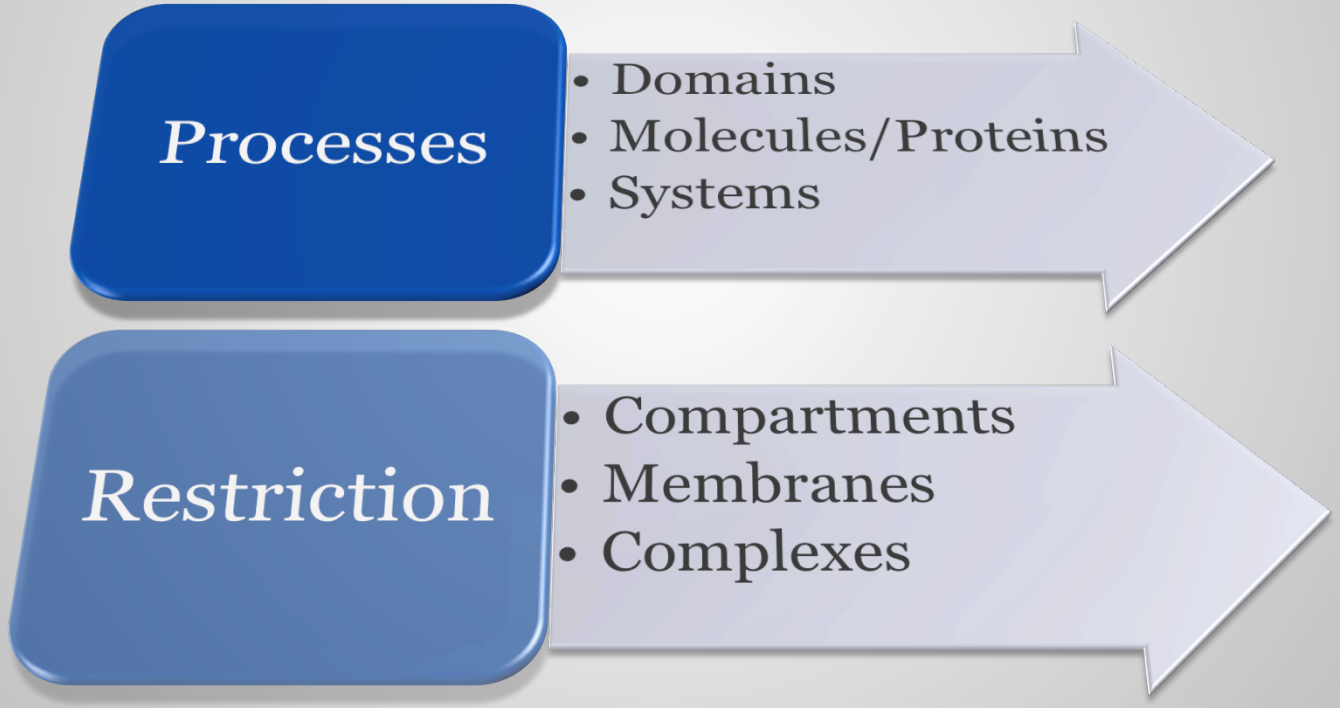



```
{
  T,U,V,Z, ...
}
%%
{
  (U,Z,3.5),
  (U,V,1.5,2.5,10)
}
```

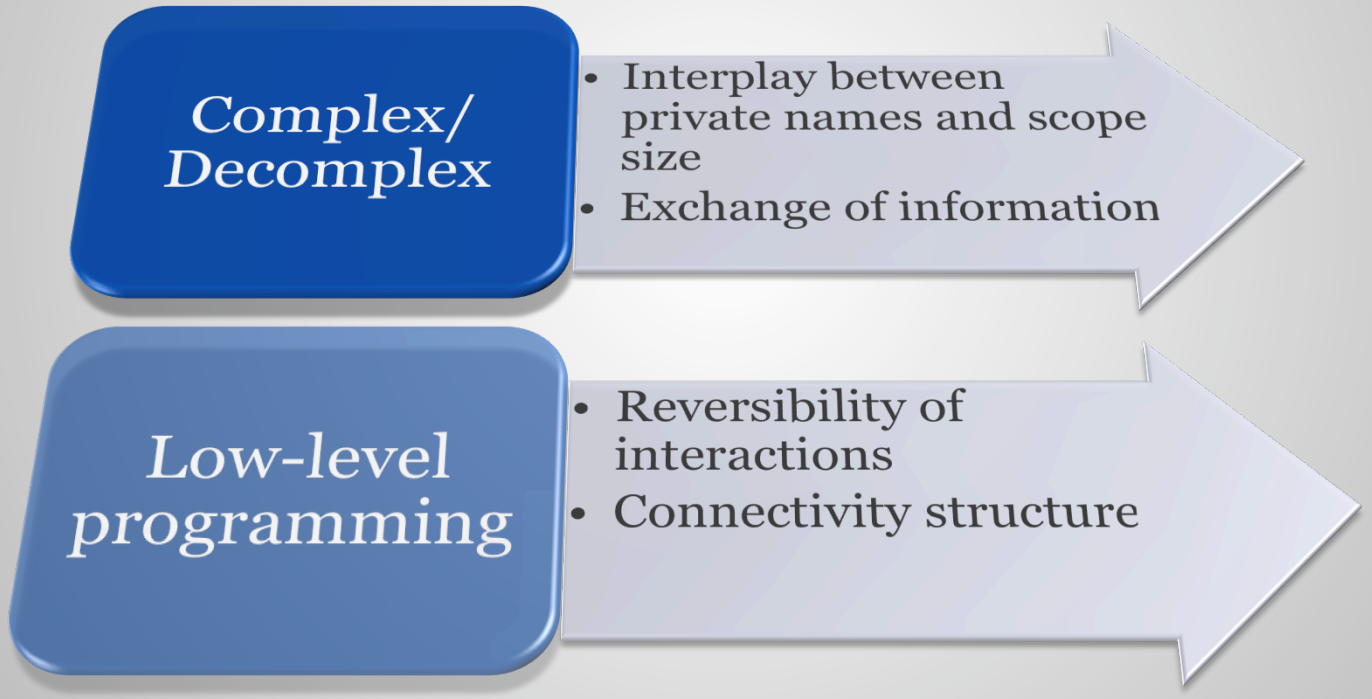
BlenX: structures



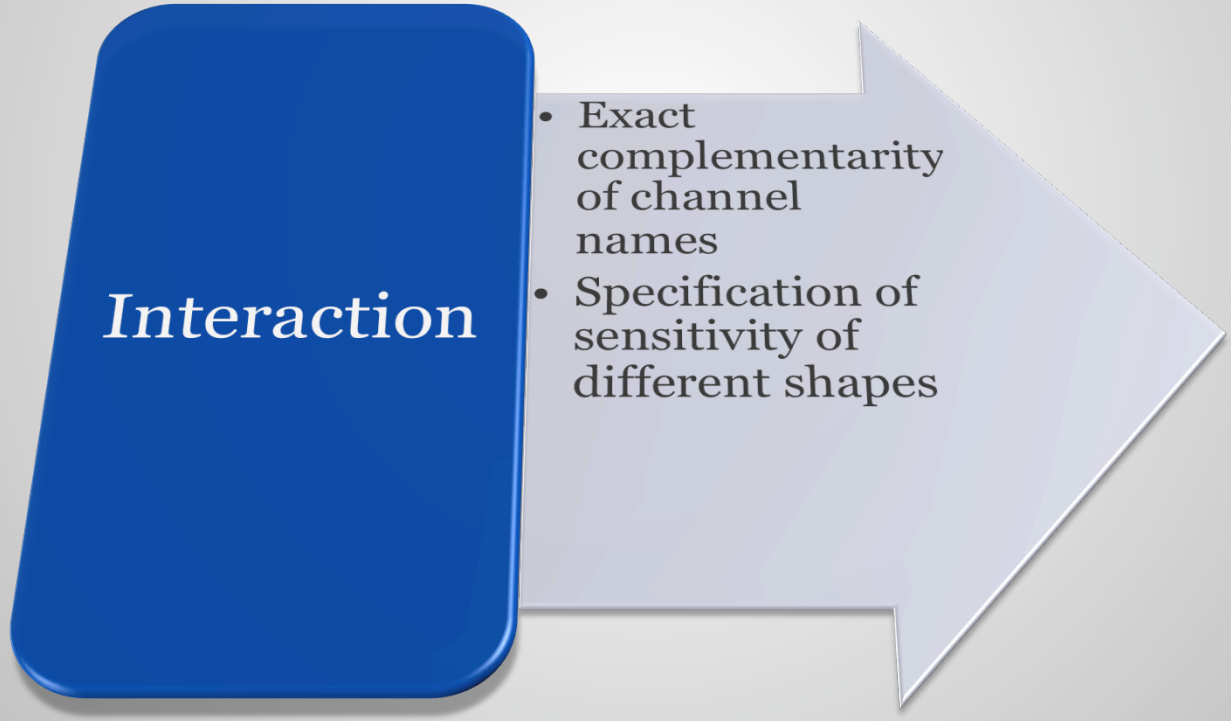
BlenX: 1st concern



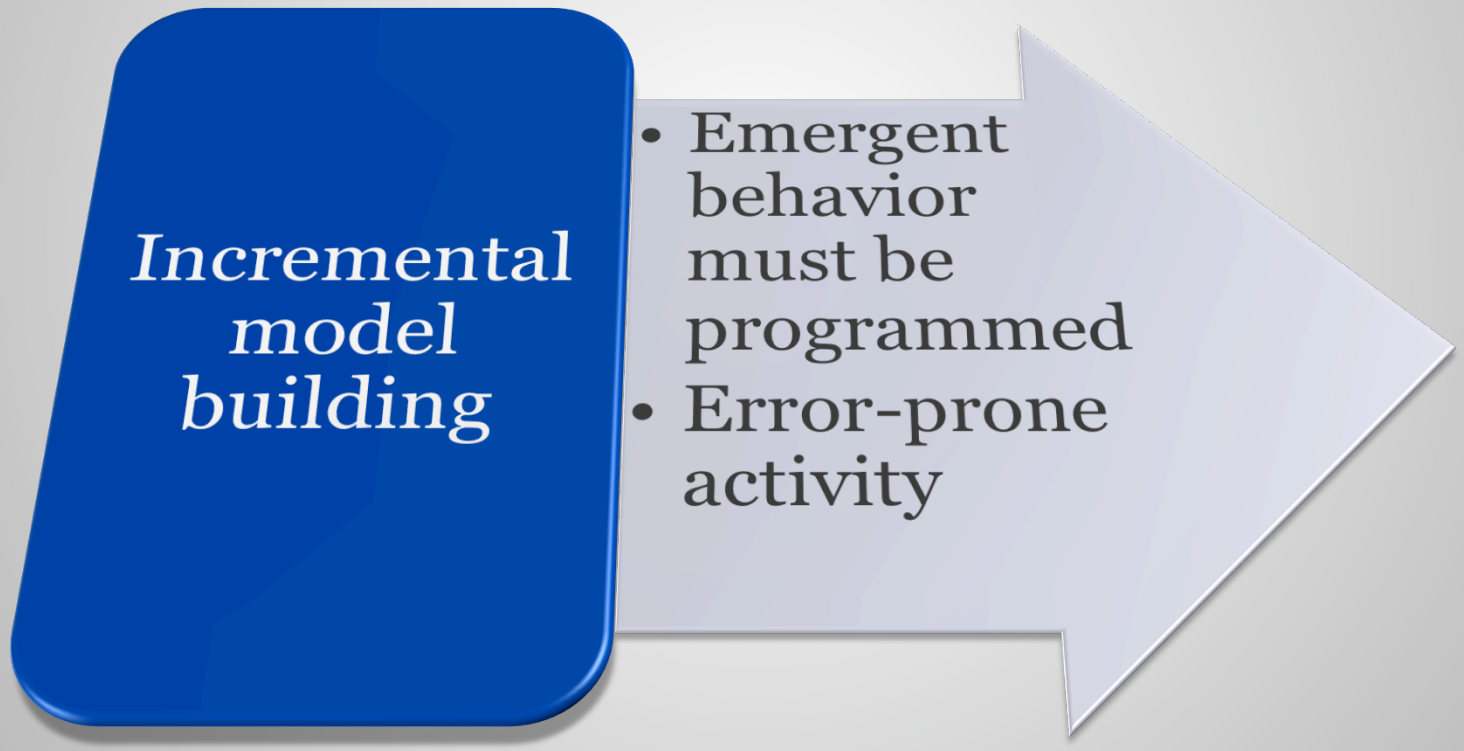
BlenX: 2nd concern



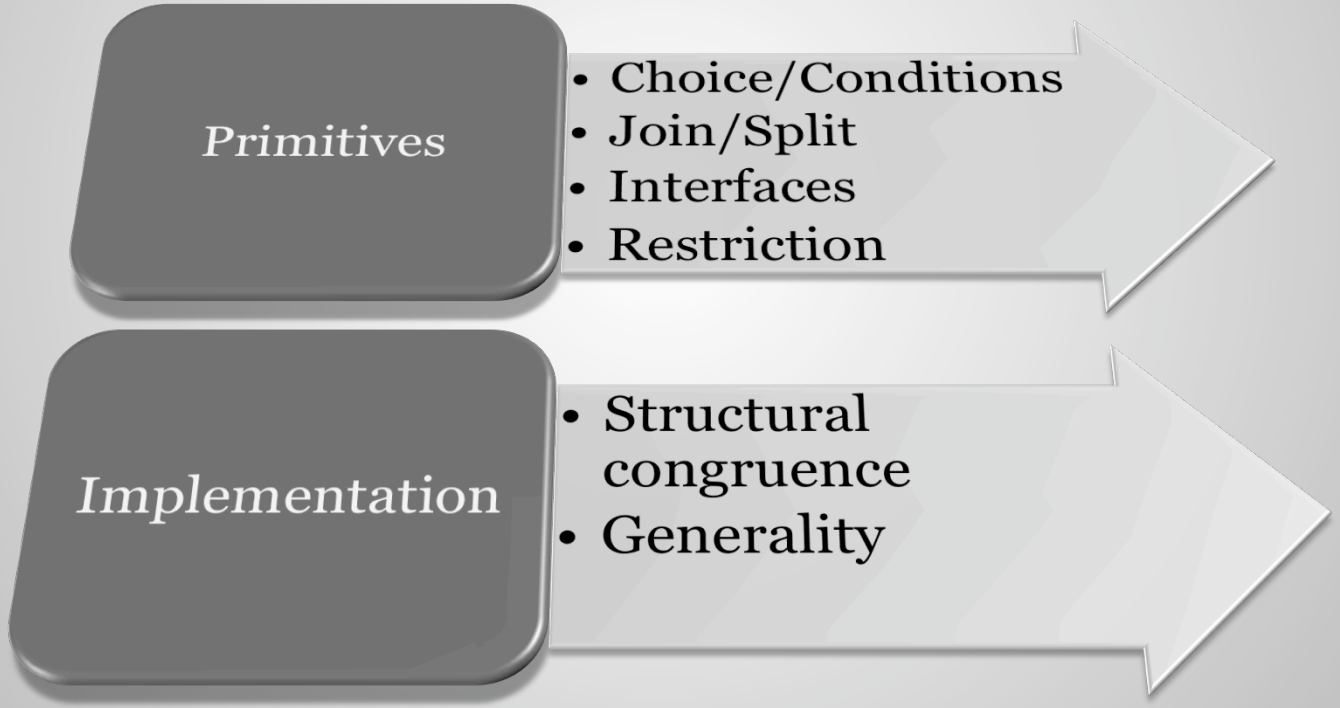
BlenX: 3rd concern



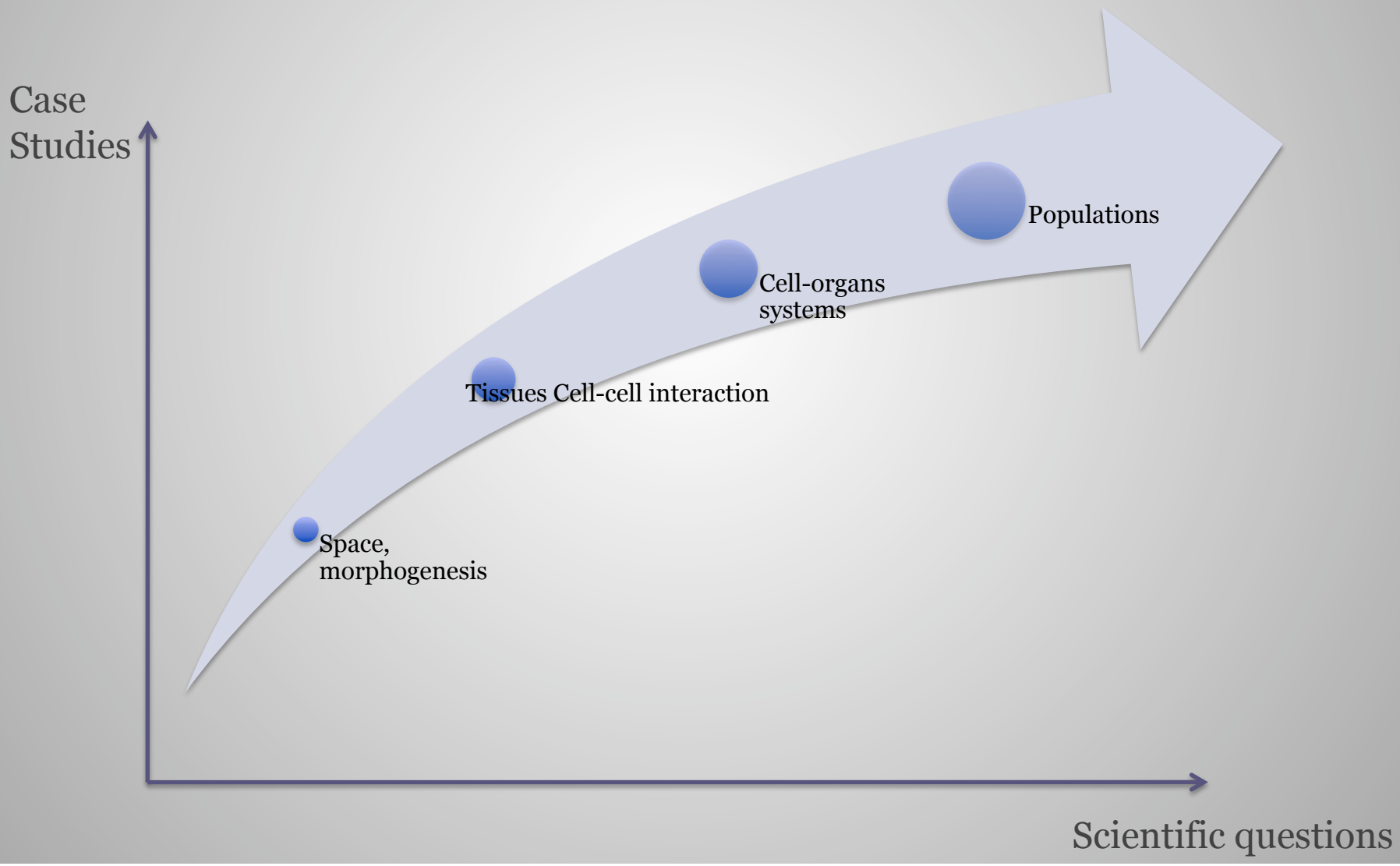
BlenX: 4th concern



BlenX: further concerns



The current generation: evolutionary pressure



Other species

Sπ-calculus dialects, BioAmbients, Brane calculi,

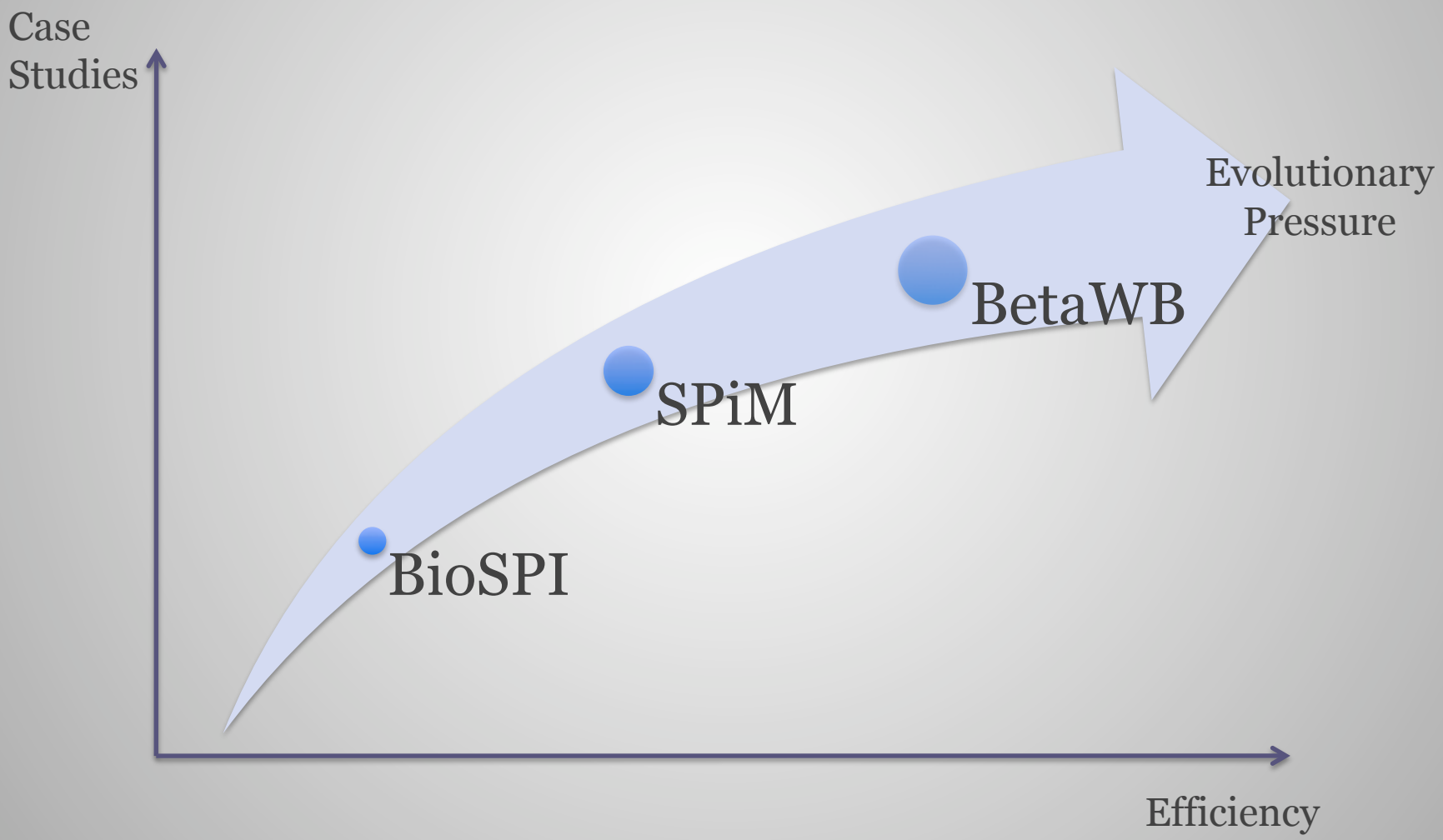
κ-calculus, CCS-R, BioPEPA, …

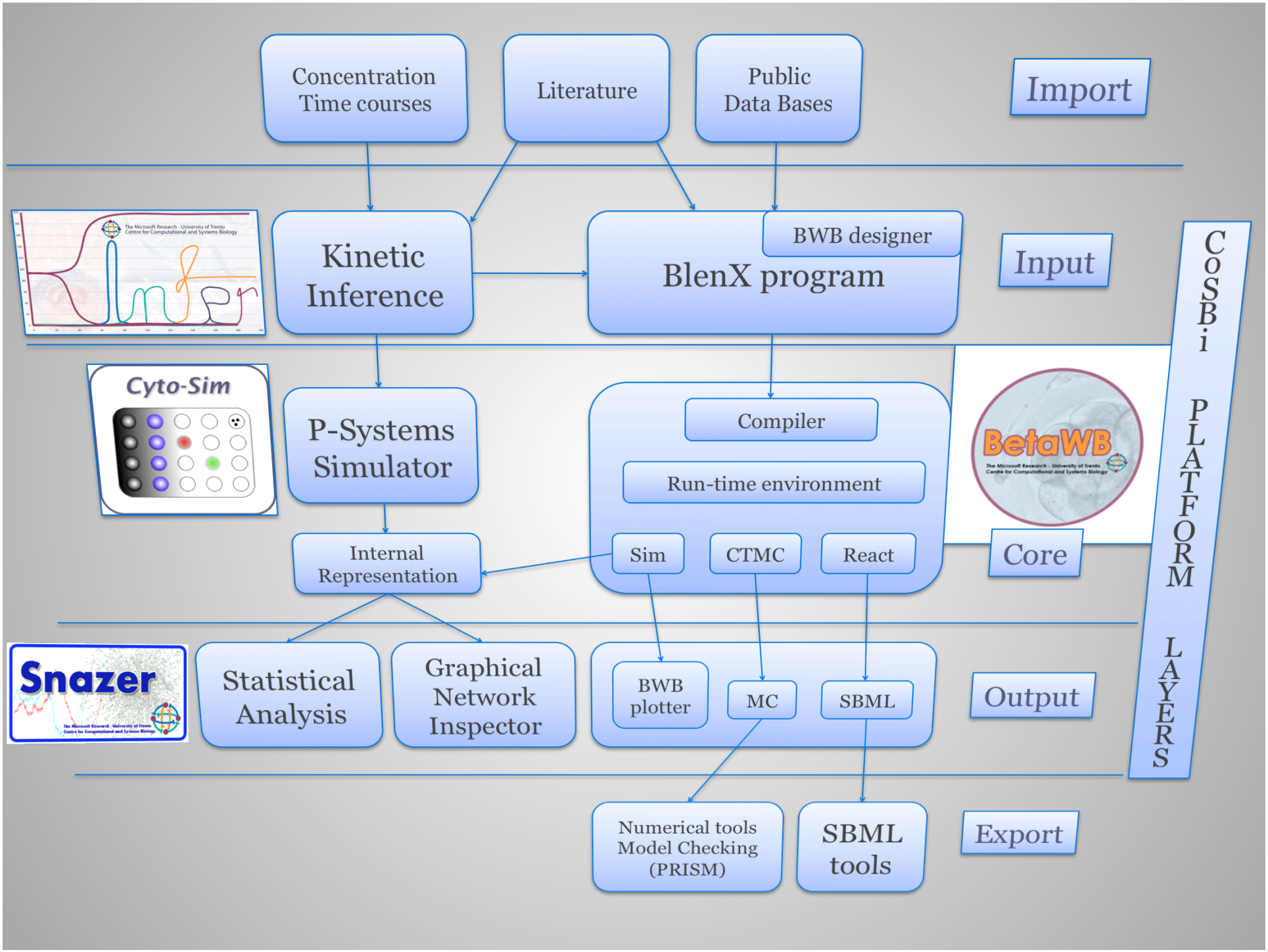
P-systems, Petri nets, statecharts, BioUML,
cellular automata, …

ODEs

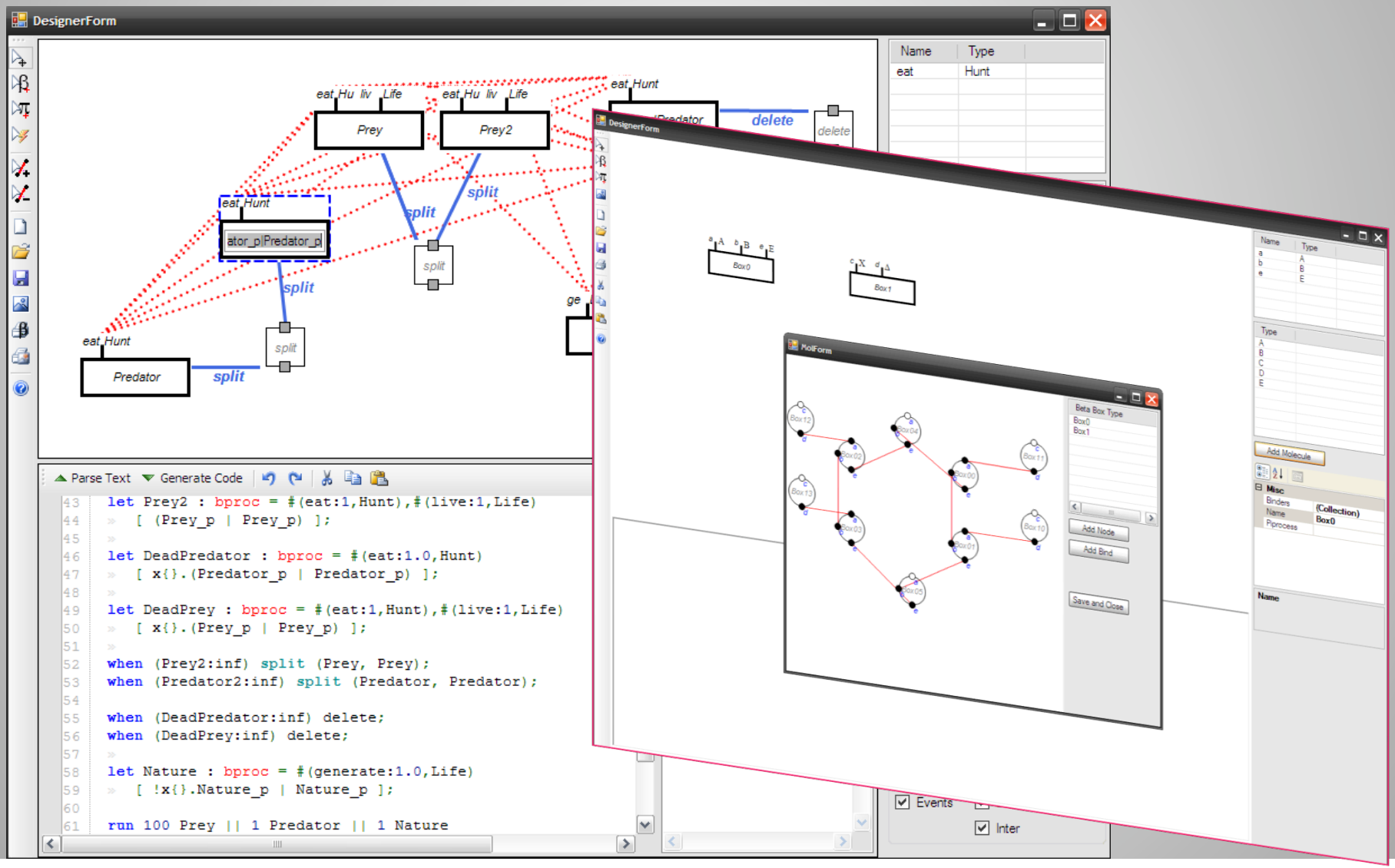
Computational tools

Computational support

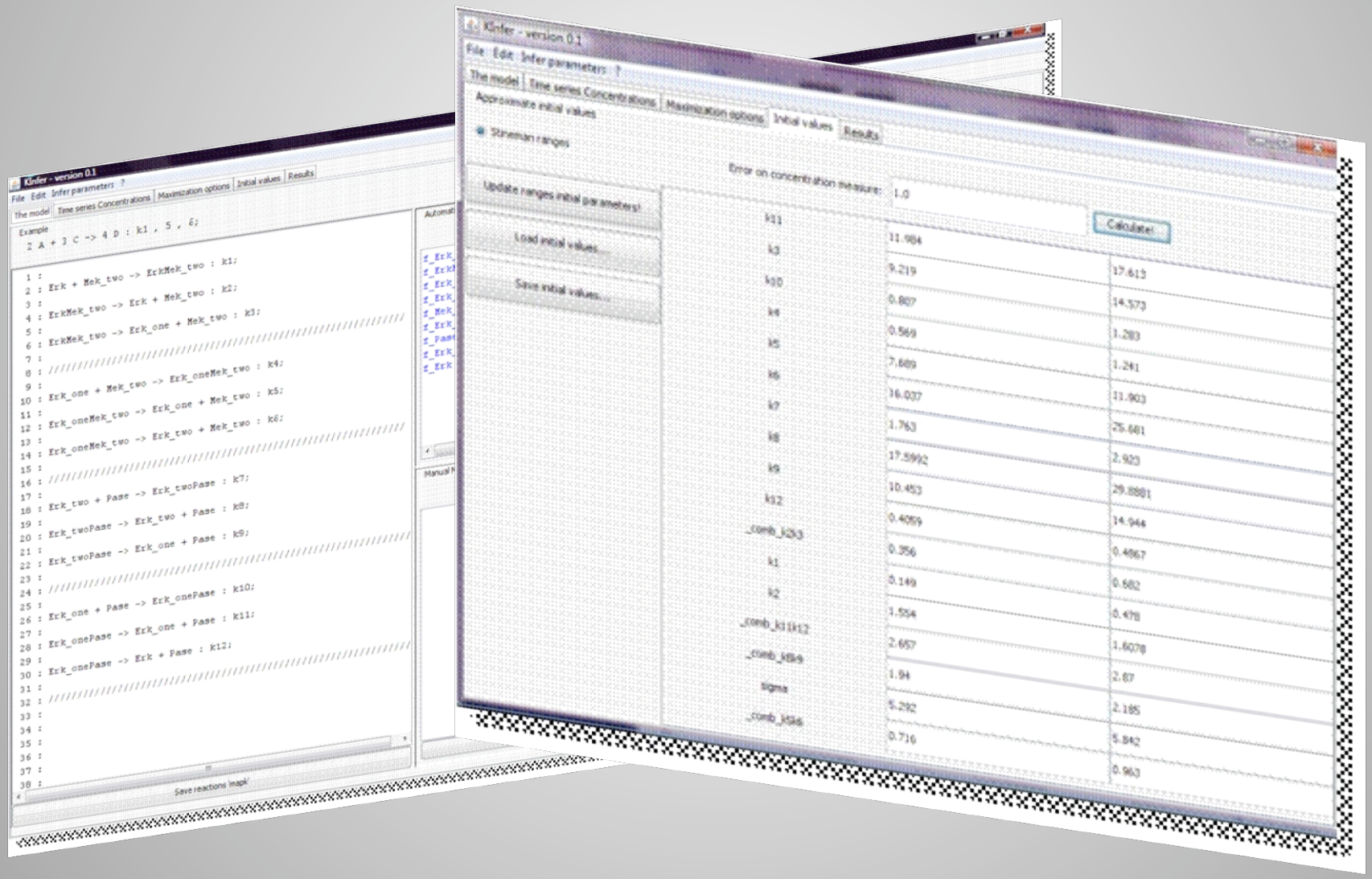



Concentration Time courses
Literature
Public Data Bases
Import
Kinetic Inference
BWB designer
BlenX program
Input
Cyto-Sim
P-Systems Simulator
Compiler
Run-time environment
Sim
CTMC
React
Internal Representation
BetaWB
Core
COSBi PLATFORM LAYERS
Snazer
Statistical Analysis
Graphical Network Inspector
BWB plotter
MC
SBML
Output
Numerical tools Model Checking (PRISM)
SBML tools
Export

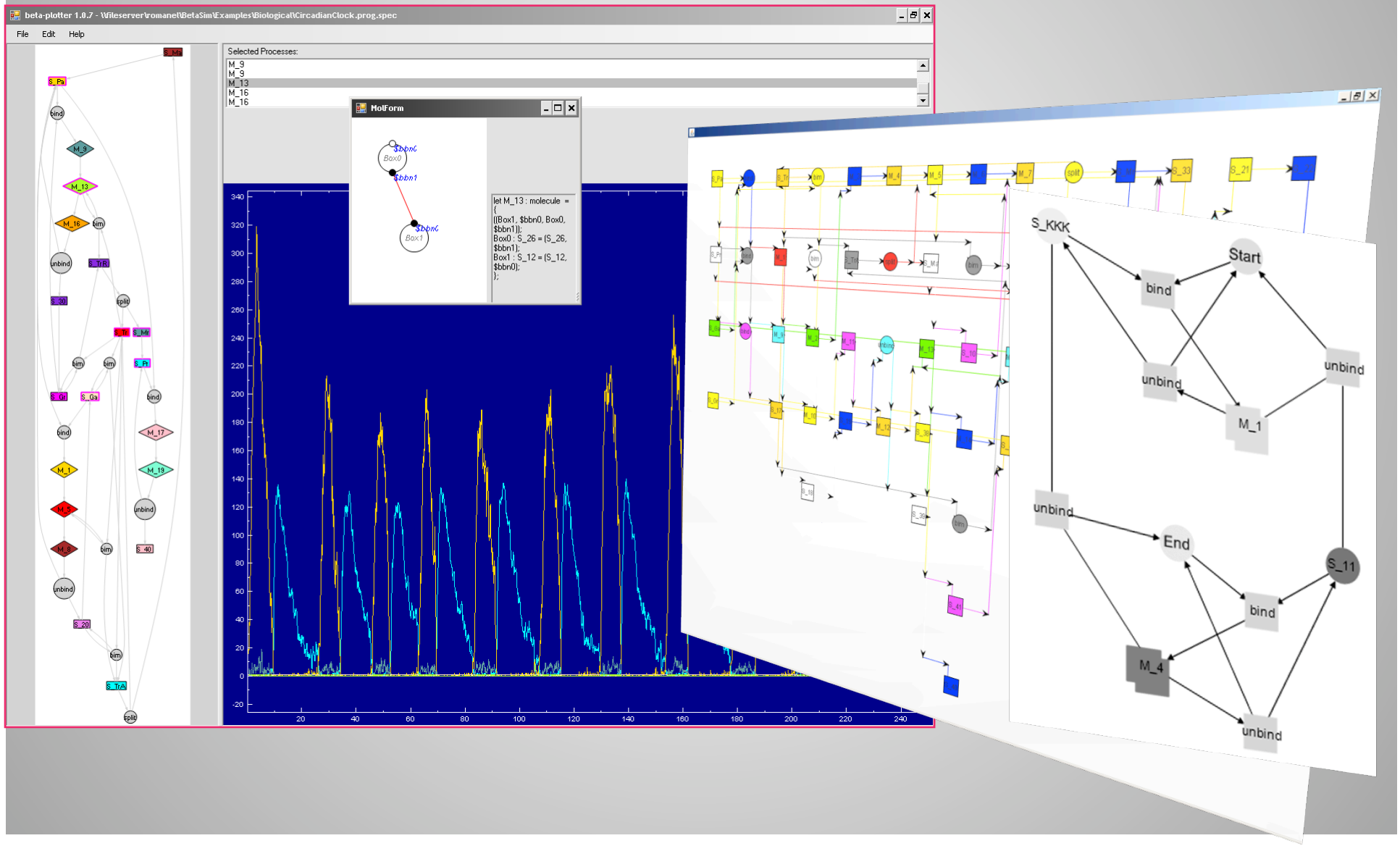
Computational support: Input



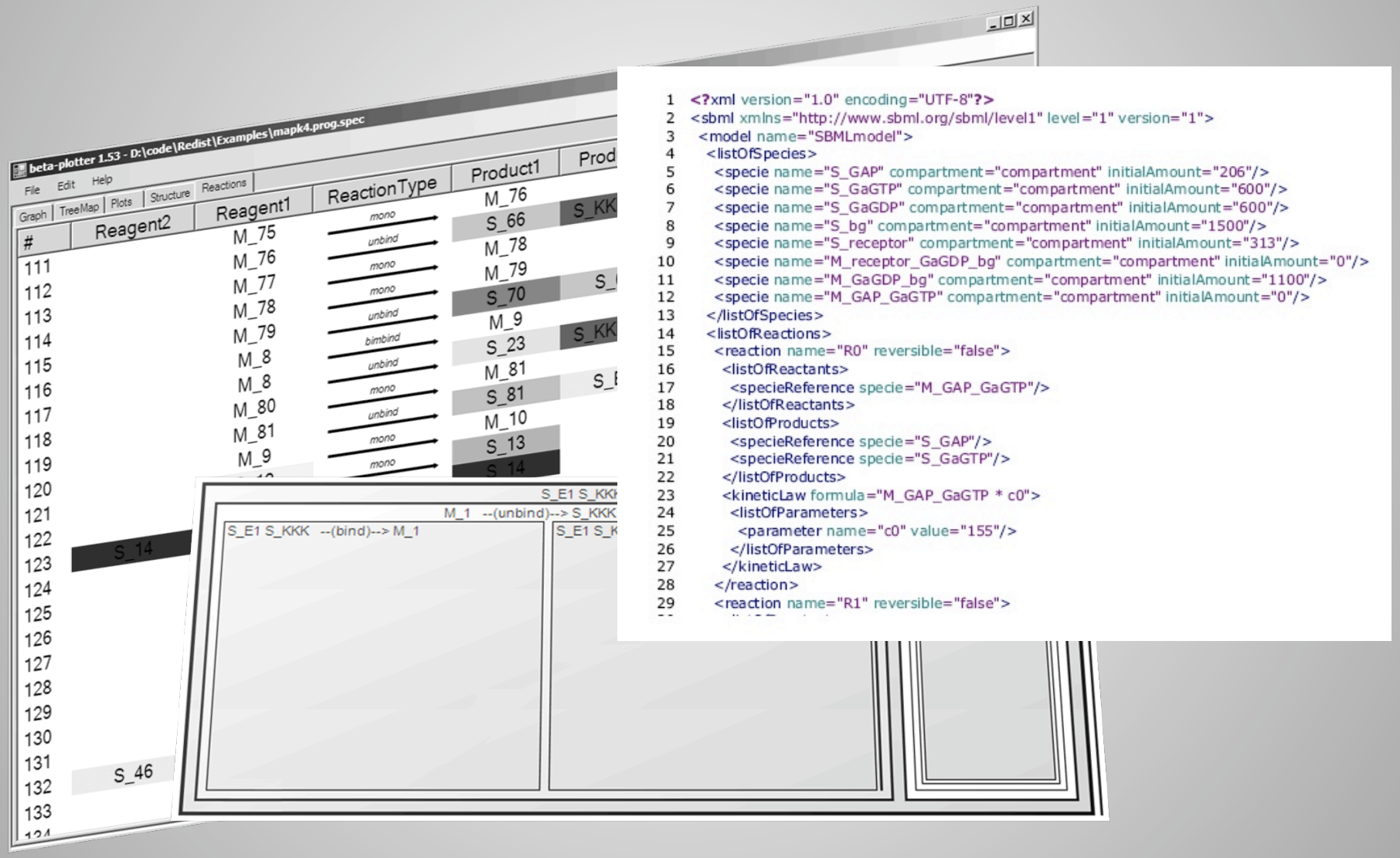
Computational support: further features



Computational support: Output

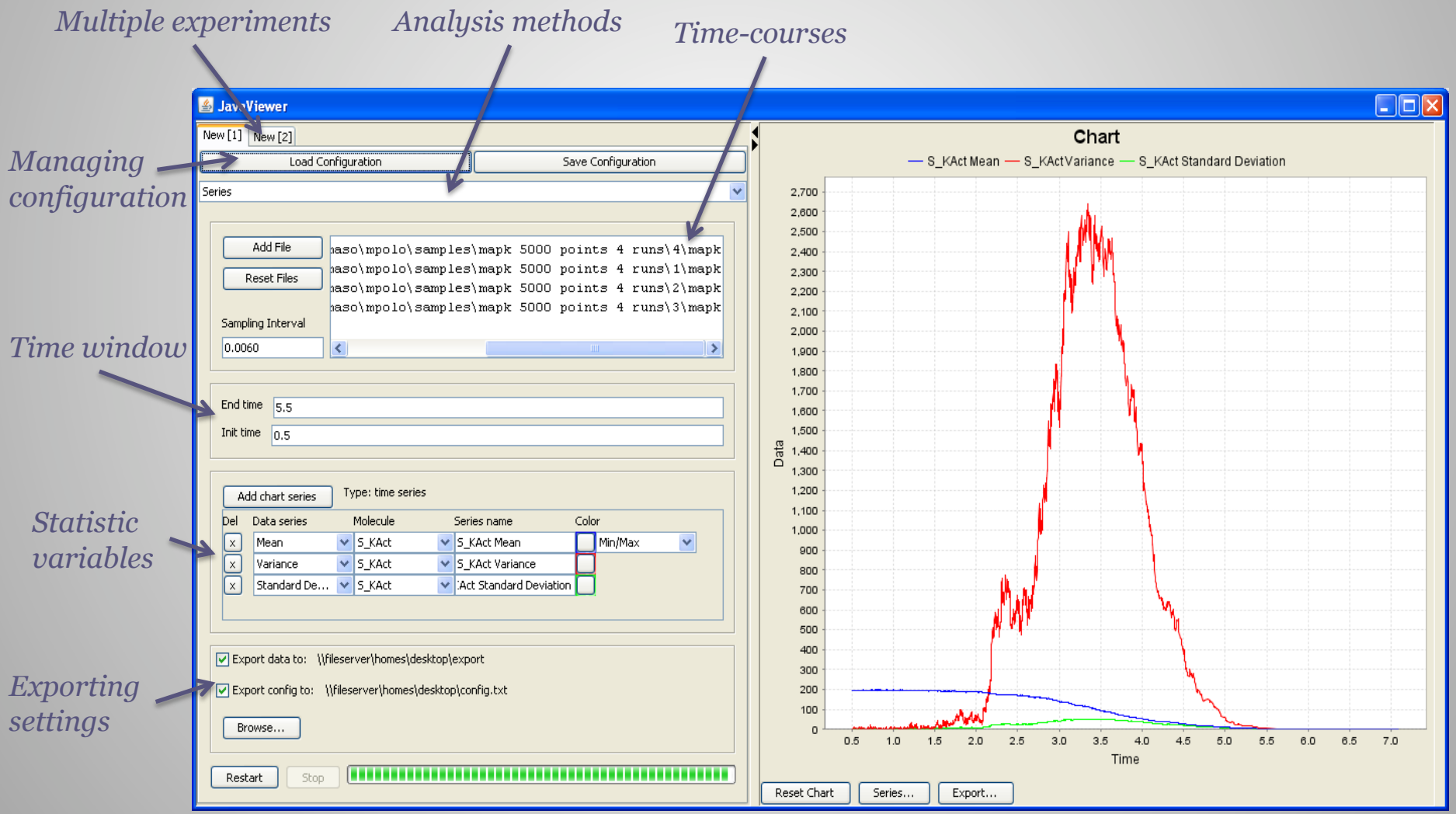


Computational support: Output

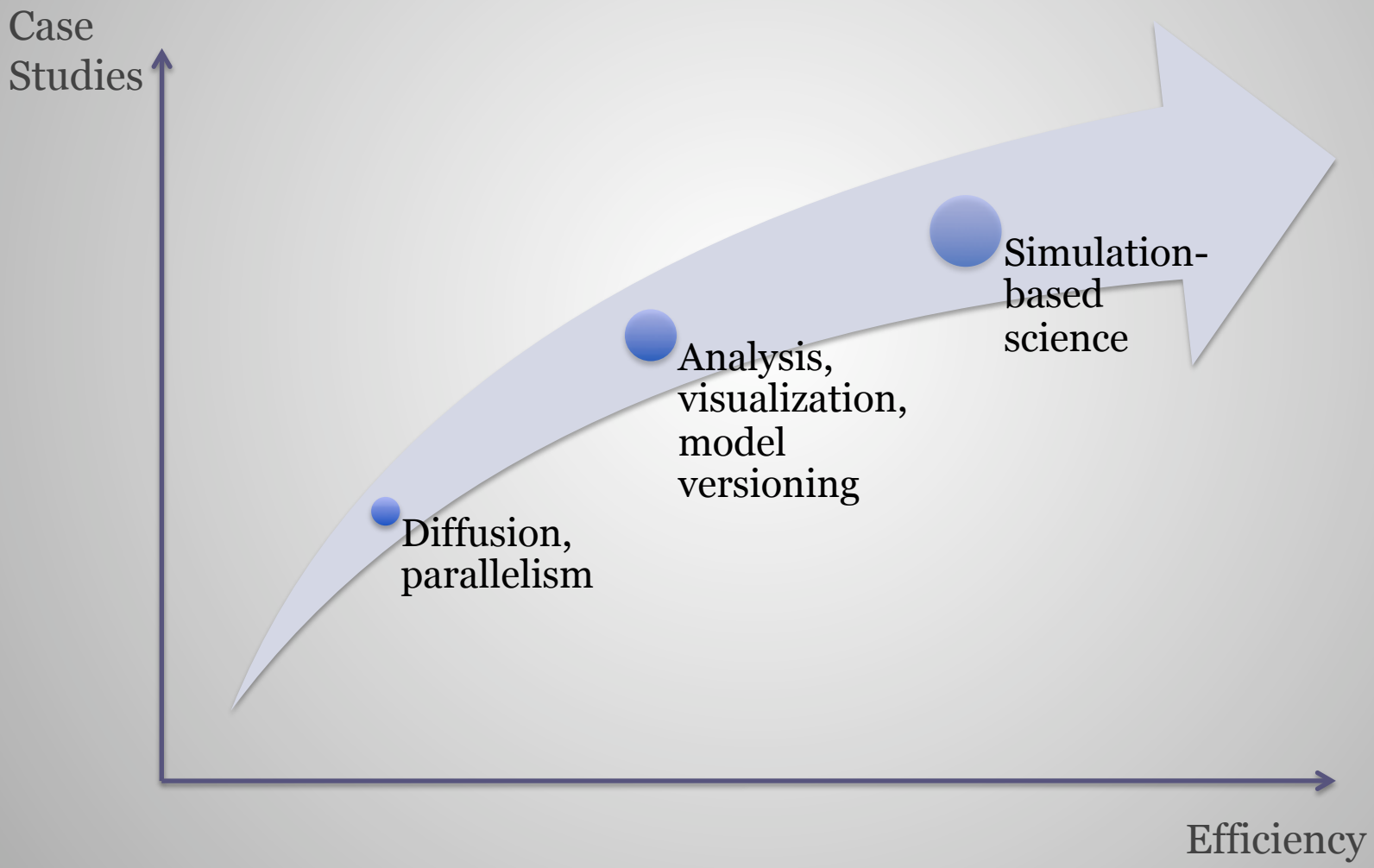



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <sbml xmlns="http://www.sbml.org/sbml/level1" level="1" version="1">
3   <model name="SBMLmodel">
4    <listOfSpecies>
5     <specie name="S_GAP" compartment="compartment" initialAmount="206"/>
6     <specie name="S_GaGTP" compartment="compartment" initialAmount="600"/>
7     <specie name="S_GaGDP" compartment="compartment" initialAmount="600"/>
8     <specie name="S_bg" compartment="compartment" initialAmount="1500"/>
9     <specie name="S_receptor" compartment="compartment" initialAmount="313"/>
10    <specie name="M_receptor_GaGDP_bg" compartment="compartment" initialAmount="0"/>
11    <specie name="M_GaGDP_bg" compartment="compartment" initialAmount="1100"/>
12    <specie name="M_GAP_GaGTP" compartment="compartment" initialAmount="0"/>
13   </listOfSpecies>
14   <listOfReactions>
15    <reaction name="R0" reversible="false">
16     <listOfReactants>
17      <specieReference specie="M_GAP_GaGTP"/>
18     </listOfReactants>
19     <listOfProducts>
20      <specieReference specie="S_GAP"/>
21      <specieReference specie="S_GaGTP"/>
22     </listOfProducts>
23     <kineticLaw formula="M_GAP_GaGTP * c0">
24      <listOfParameters>
25       <parameter name="c0" value="155"/>
26      </listOfParameters>
27     </kineticLaw>
28    </reaction>
29    <reaction name="R1" reversible="false">
```

Computational support: further features

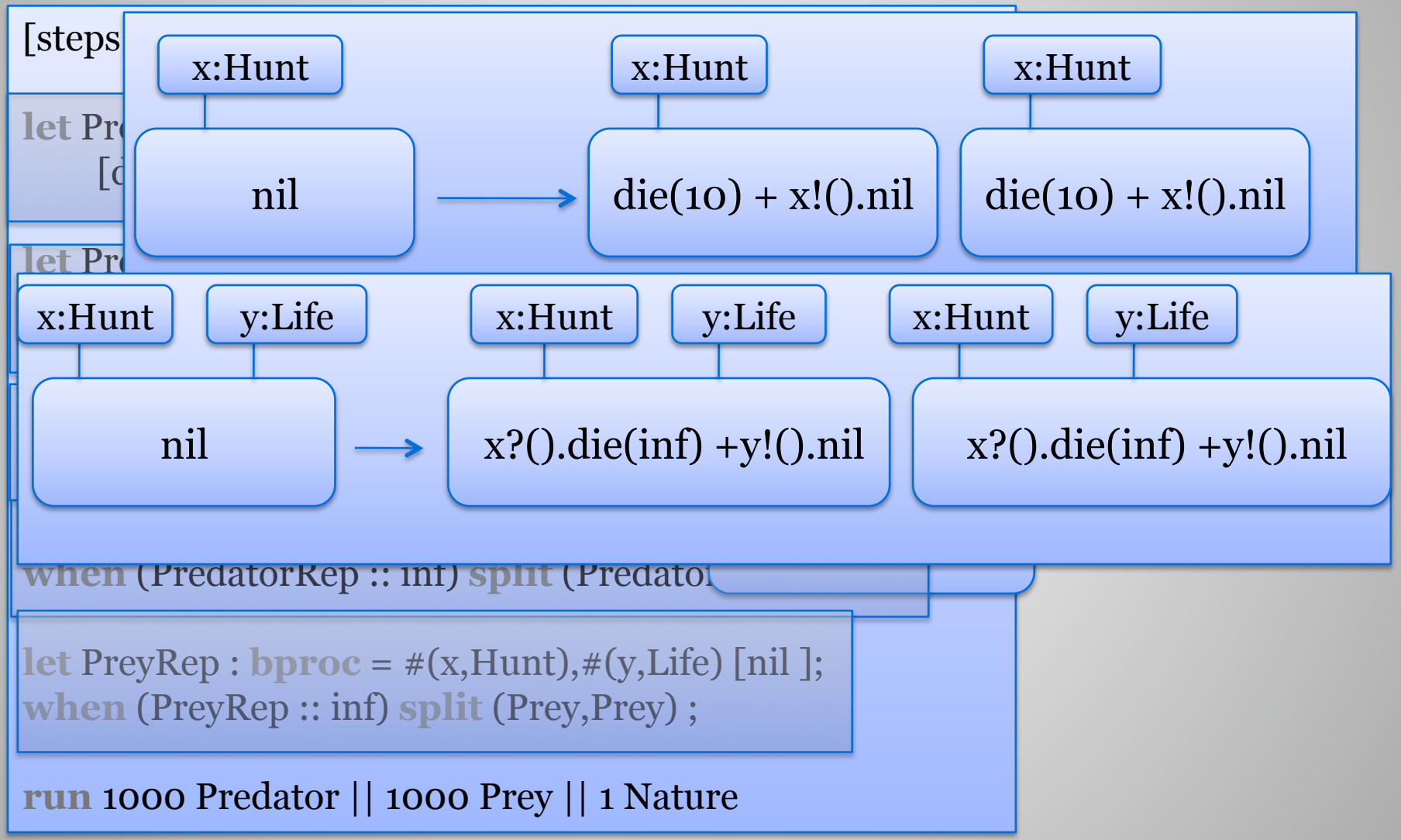


Computational support: evolutionary pressure

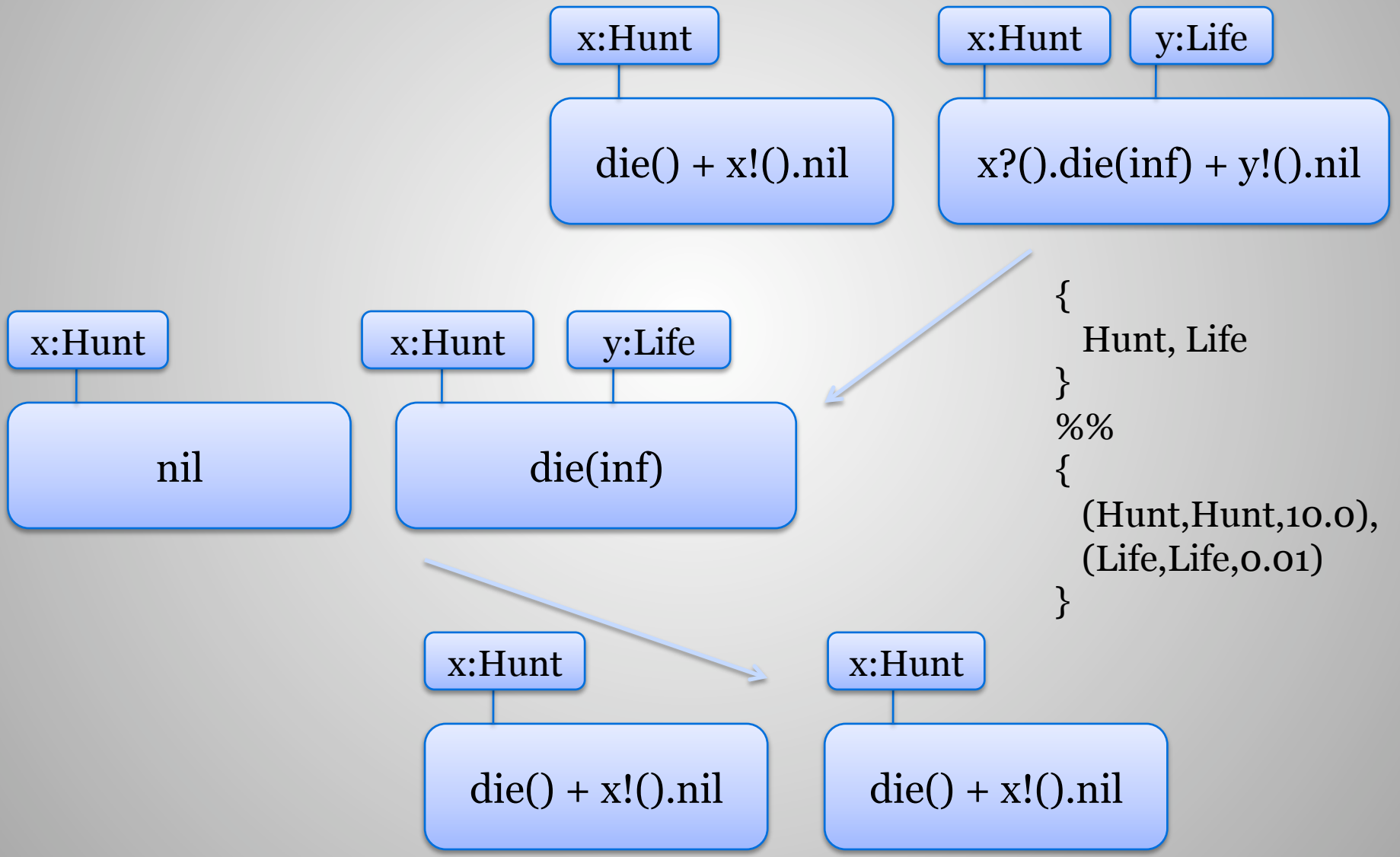


Examples

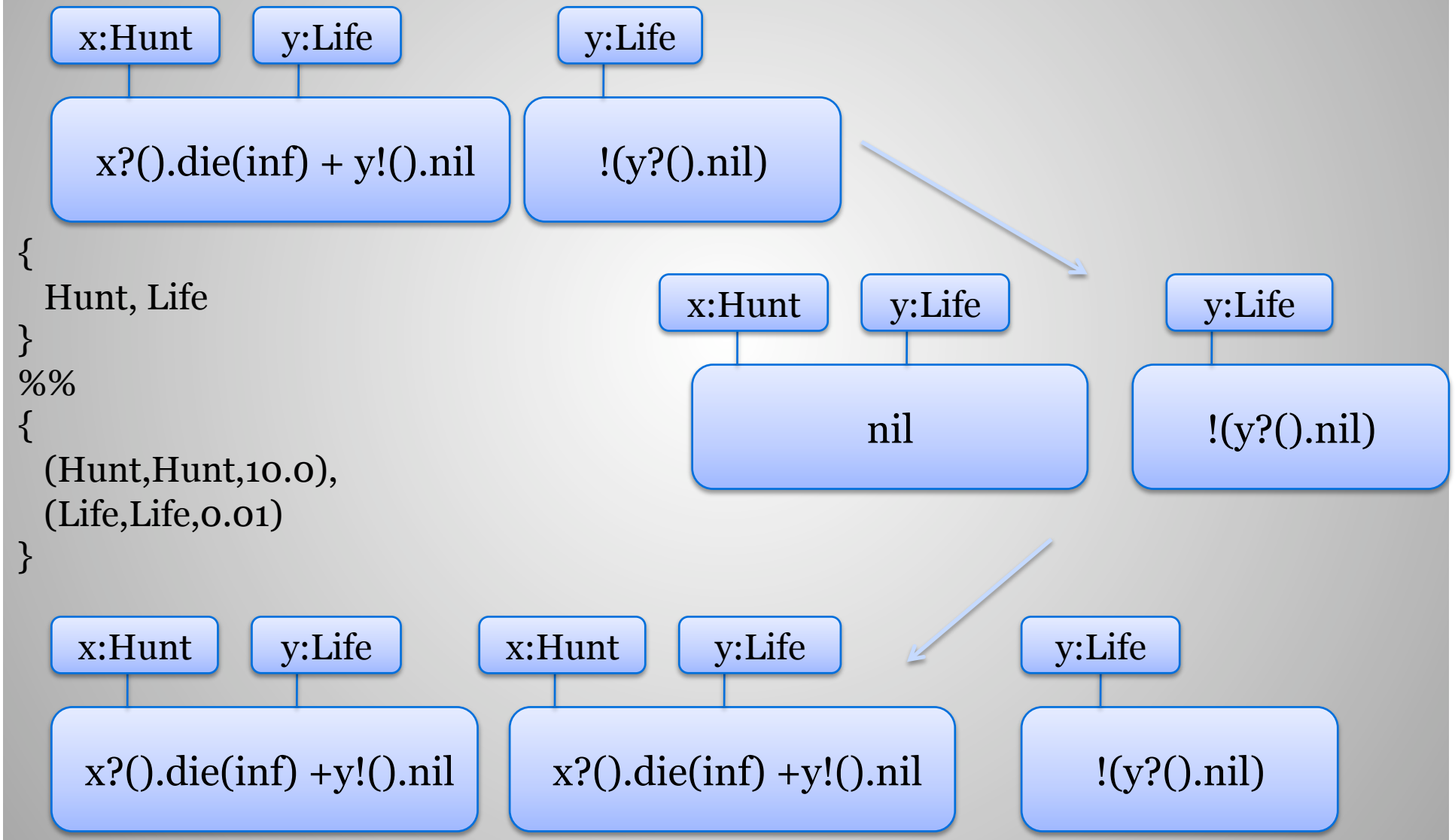
Predator - Prey



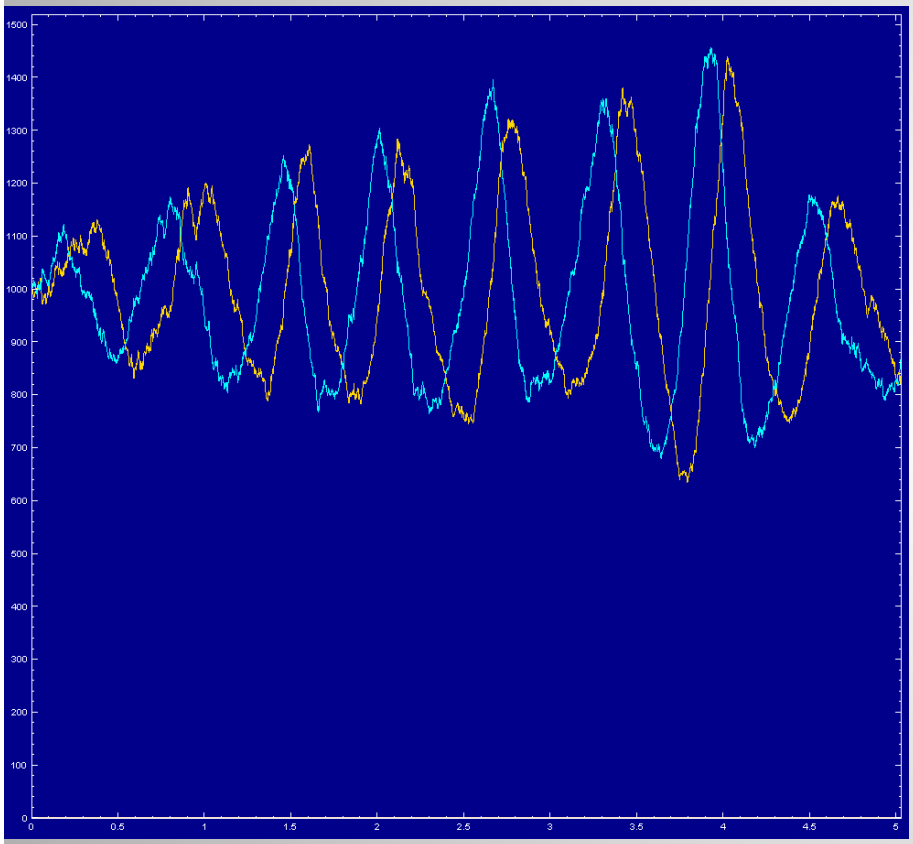
Predator - Prey

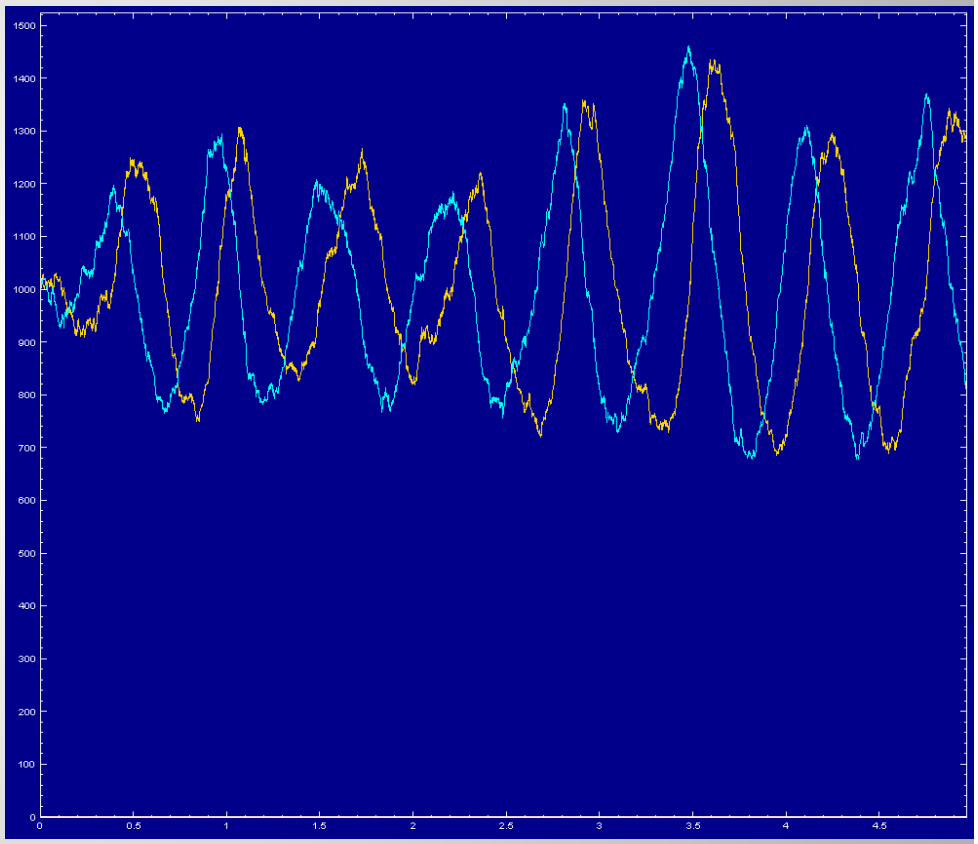


Predator - Prey



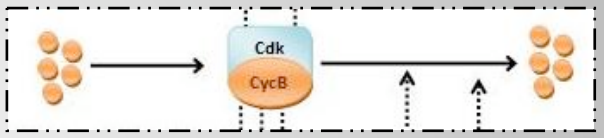
Predator - Prey





Cell cycle: synthesis/degradation

$$\frac{d|CYCBT|}{dt} = \frac{k1}{\alpha} - k2p * |CYCBT| - \ldots$$



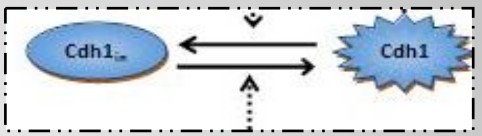
Function file

```
let alpha : const = 0.00236012;
let k1 : const =  0.04;
let k2p : const =  0.04;
let d_dtCYCBT_1 : function = k1/alpha;
let d_dtCYCBT_2 : function = k2p*|CYCBT|;
```

Model file

```
let CYCBT: bproc = #(x,CYCBT)[ nil ];
when(CYCBT : : d_dtCYCBT_1) new(1);
when(CYCBT : : d_dtCYCBT_2) delete(1);
```

Cell cycle: CDH1 act/inact



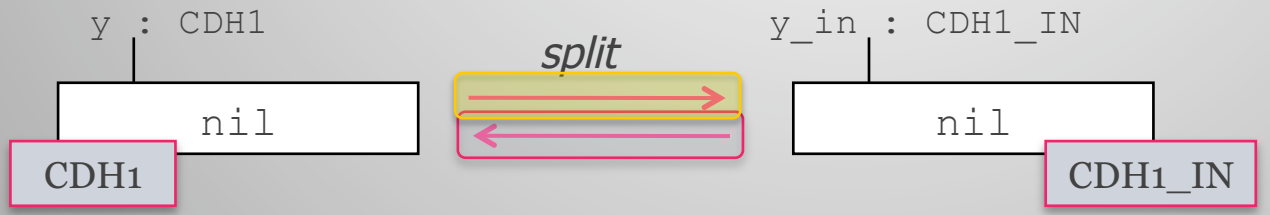
$$\frac{d|CDH1|}{dt} = \frac{k3p * (|CDH1\_IN|)}{J3 + \alpha * |CDH1\_IN|} + \dots - \frac{k4p * \alpha * |SK| * |CDH1|}{J4 + \alpha * |CDH1|} - \dots$$

Function file

```
let d_dtCDH1_1 : function = (k3p*(|CDH1_IN|))/(J3+alpha*|CDH1_IN|);
let d_dtCDH1_4 : function = (k4p*alpha*|SK|*|CDH1|)/(J4+alpha*|CDH1|);
```

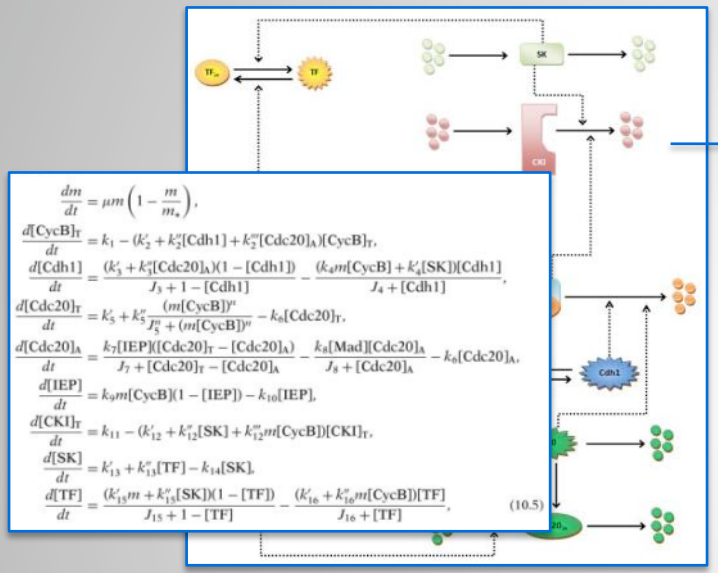
Model file

```
let CDH1 : bproc = #(y,CDH1)[ nil ];
let CDH1_IN : bproc = #(y_in,CDH1_IN) [ nil ];
when(CDH1_IN : : d_dtCDH1_1 ) split(Nil, CDH1);
when(CDH1 : : d_dtCDH1_4 ) split(Nil, CDH1_IN);
```



Total amount of CDH1 is constant as set in the initial conditions

Cell cycle



$$\frac{dm}{dt} = \mu m\left(1 - \frac{m}{m_*}\right),$$
$$\frac{d[\mathrm{CycB}]_\mathrm{T}}{dt} = k_1 - (k_2' + k_2''[\mathrm{Cdh1}] + k_2'''[\mathrm{Cdc20}]_\mathrm{A})[\mathrm{CycB}]_\mathrm{T},$$
$$\frac{d[\mathrm{Cdh1}]}{dt} = \frac{(k_3' + k_3''[\mathrm{Cdc20}]_\mathrm{A})(1 - [\mathrm{Cdh1}])}{J_3 + 1 - [\mathrm{Cdh1}]} - \frac{(k_4 m[\mathrm{CycB}] + k_4'[\mathrm{SK}])[\mathrm{Cdh1}]}{J_4 + [\mathrm{Cdh1}]},$$
$$\frac{d[\mathrm{Cdc20}]_\mathrm{T}}{dt} = k_5' + k_5''\frac{(m[\mathrm{CycB}])^n}{J_5^n + (m[\mathrm{CycB}])^n} - k_6[\mathrm{Cdc20}]_\mathrm{T},$$
$$\frac{d[\mathrm{Cdc20}]_\mathrm{A}}{dt} = \frac{k_7[\mathrm{IEP}]([\mathrm{Cdc20}]_\mathrm{T} - [\mathrm{Cdc20}]_\mathrm{A})}{J_7 + [\mathrm{Cdc20}]_\mathrm{T} - [\mathrm{Cdc20}]_\mathrm{A}} - \frac{k_8[\mathrm{Mad}][\mathrm{Cdc20}]_\mathrm{A}}{J_8 + [\mathrm{Cdc20}]_\mathrm{A}} - k_6[\mathrm{Cdc20}]_\mathrm{A},$$
$$\frac{d[\mathrm{IEP}]}{dt} = k_9 m[\mathrm{CycB}](1 - [\mathrm{IEP}]) - k_{10}[\mathrm{IEP}],$$
$$\frac{d[\mathrm{CKI}]_\mathrm{T}}{dt} = k_{11} - (k_{12}' + k_{12}''[\mathrm{SK}] + k_{12}'''m[\mathrm{CycB}])[\mathrm{CKI}]_\mathrm{T},$$
$$\frac{d[\mathrm{SK}]}{dt} = k_{13}' + k_{13}''[\mathrm{TF}] - k_{14}[\mathrm{SK}],$$
$$\frac{d[\mathrm{TF}]}{dt} = \frac{(k_{15}'m + k_{15}''[\mathrm{SK}])(1 - [\mathrm{TF}])}{J_{15} + 1 - [\mathrm{TF}]} - \frac{(k_{16}' + k_{16}''m[\mathrm{CycB}])[\mathrm{TF}]}{J_{16} + [\mathrm{TF}]}, \quad (10.5)$$

Function file

```
let J15 : const =  0.01  ;
…
let n : const =  4.0  ;

let SIGMA : function = alpha*|CYCBT| + alpha*|CKIT|+1/keq;
le                    (2*alpha*|CYCBT|*alpha*|CKIT|)/(SIGMA + sqrt(SIGMA*SIGMA -
4
```

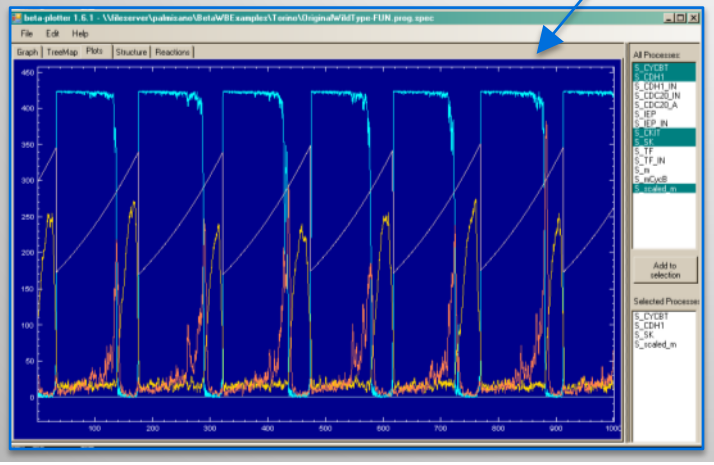
Typing file

```
{CDH1, CDH1_IN, CYCBT, CDC20_A, M, IEP_IN, CDC20_IN, IEP, SK, TF,  TF_IN, CKIT }
```

```
let d_dtCDC20_IN_1 : function = k5p/alpha;
let d_dtCDC20_IN_2 : function = (k5s)/(alpha*(1+pow((J5/(m*alphaDimer)),n)));
let d_dtCDC20_IN_3 : function = (k8*|CDC20_A|)/(J8+alpha*|CDC20_A|);
let d_dtCDC20_IN_4 : function = (k7*alpha*|IEP|*|CDC20_IN|)/(J7+alpha*|CDC20_IN|);
let d_dtCDC20_IN_5 : function = k6*|CDC20_IN|;

let d_dtCDH1_1 : function = (k3p*(|CDH1_IN|))/(J3+alpha*|CDH1_IN|);
let d_dtCDH1_2 : function = (k3s*alpha*|CDC20_A|*|CDH1_IN|)/(J3+alpha*|CDH1_IN|);
let d_dtCDH1_3 : function = (k4*m*alphaDimer*|CDH1|)/(J4+alpha*|CDH1|);
let d_dtCDH1_4 : function = (k4p*alpha*|SK|*|CDH1|)/(J4+alpha*|CDH1|);
```

Model file

```
[steps = 5000, delta = 0.2]

let CYCBT: bproc = #(x,CYCBT)[ nil ];
when(CYCBT:: d_dtCYCBT_1) new(1); when(CYCBT:: d_dtCYCBT_2) delete(1);
when(CYCBT:: d_dtCYCBT_3) delete(1); when(CYCBT:: d_dtCYCBT_4) delete(1);

let CDH1: bproc = #(y,CDH1)[ nil ];                    let CDH1_IN : bproc = #(y_in,CDH1_IN) [ nil ];
when(CDH1_IN :: d_dtCDH1_1 ) split(Nil, CDH1); when(CDH1_IN :: d_dtCDH1_2 ) split(Nil, CDH1);
when(CDH1 :: d_dtCDH1_3 ) split(Nil, CDH1_IN); when(CDH1 :: d_dtCDH1_4 ) split(Nil, CDH1_IN);

let CDC20_IN : bproc = #(a,CDC20_IN)[ nil ];     let CDC20_A : bproc = #(a,CDC20_A)[ nil ];
when(CDC20_IN :: d_dtCDC20_IN_1 ) new(1); when(CDC20_IN :: d_dtCDC20_IN_2 ) new(1);
when(CDC20_IN :: d_dtCDC20_IN_5 ) delete(1); when(CDC20_IN :: d_dtCDC20_IN_4 ) split(Nil, CDC20_A);
when(CDC20_A :: d_dtCDC20_A_2) split(Nil,CDC20_IN); when(CDC20_A :: d_dtCDC20_A_3) delete(1);

let IEP : bproc = #(y,IEP)[ nil ];                     let IEP_IN : bproc = #(y_in,IEP_IN) [ nil ];
when(IEP_IN :: d_dtIEP_1 ) split(Nil, IEP); when(IEP :: d_dtIEP_2 ) split(Nil, IEP_IN);

let CKIT : bproc = #(x,CKIT )[ nil ];
when(CKIT :: d_dtCKIT_1 ) new(1); when(CKIT :: d_dtCKIT_2 ) delete(1);
when(CKIT :: d_dtCKIT_3 ) delete(1); when(CKIT :: d_dtCKIT_4 ) delete(1);

let SK : bproc = #(x,SK)[ nil ];
when(SK :: d_dtSK_2) new(1); when(SK :: d_dtSK_3) delete(1);

let TF : bproc = #(y,TF)[ nil ];                       let TF_IN : bproc = #(y_in,TF_IN) [ nil ];
when(TF_IN :: d_dtTF_1) split(Nil, TF); when(TF_IN :: d_dtTF_2) split(Nil, TF);
when(TF :: d_dtTF_3) split(Nil, TF_IN); when(TF :: d_dtTF_4) split(Nil, TF_IN);

when ( : mCycB -> 0.2, mCycB <- 0.1 :  ) update (m, mass_div);
when ( : scaled_m -> 100000000 :  ) update (m, mass_div);

run 25 CKIT || 97 CYCBT || 39 SK || 5 CDH1 || 419 CDH1_IN || 0 CDC20_A ||
24 CDC20_IN || 40 IEP || 384 IEP_IN || 15 TF || 409 TF_IN
```



Summing up

A novel “evolving” programming language bio-inspired

A computational platform to support testing of the language

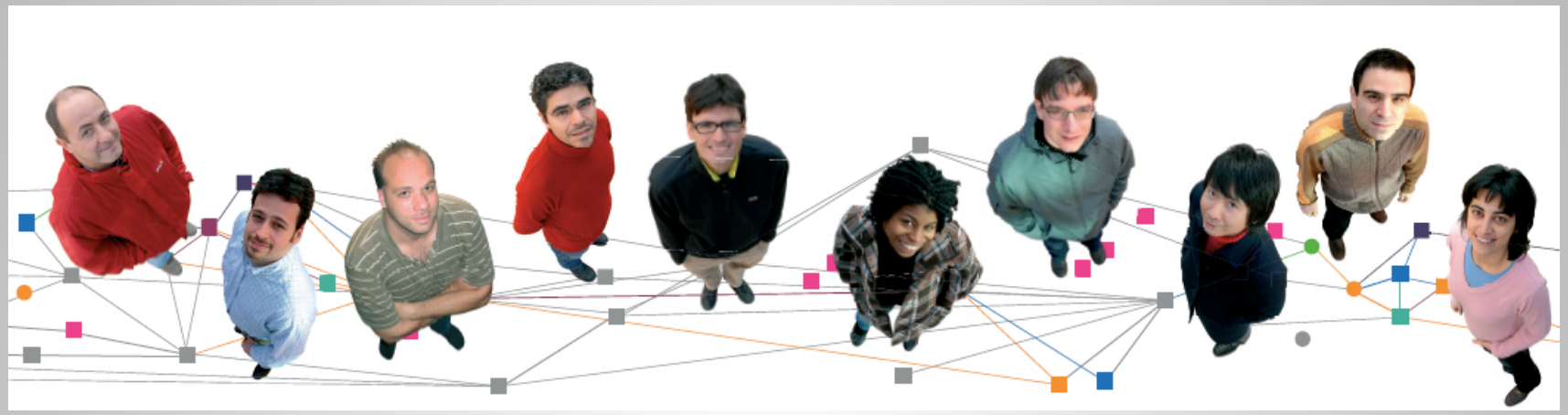
Primary applicative domain: systems biology, but also computer science

References

www.cosbi.eu

priami@cosbi.eu

Acknowledgements



Thank you for your time