A Graphical Representation for the Stochastic Pi-Calculus

Andrew Phillips Luca Cardelli

27th August 2005

Microsoft Research
7 J J Thomson Avenue
Cambridge, UK

Introduction

- Stochastic pi-calculus used to model and simulate a range of biological systems [Lecca and Priami, 2003, Priami et al., 2001, Regev et al., 2001]:
  - Able to model independent system components, which can be composed to predict emergent system behaviour.
  - Mathematical definition supports useful analysis techniques: type systems, behavioural equivalences, model checking.

- Mathematical syntax and semantics can limit accessibility to a wider audience:
  - Useful to present an alternative graphical view
  - Particularly welcomed by experimental systems biologists.

Outline

➢ Stochastic pi-calculus

➢ Graphical stochastic pi-calculus:

- Graphical syntax
- Graphical execution model

➢ Graphical biological examples:

- Evolved gene network [Francois and Hakim, 2004]
- Mapk signalling cascade [Huang and Ferrel, 1996]

➢ Automatic graph generation:

- Encoding pi-calculus to an open graph syntax
- Front end to a graphical simulator/debugger (ongoing)

The Stochastic Pi-Calculus (SPi)

Each channel x is associated with a stochastic rate given by $rate(x)$

$$
\begin{array}{rcll}
\pi & ::= & ?x(\vec{m}) & \text{Input} \\
 & | & !x(\vec{n}) & \text{Output} \\
 & | & \tau_r & \text{Delay} \\
P, Q & ::= & \pi_1.P_1 + \ldots + \pi_N.P_N & \text{Choice} \\
 & | & P_1 \mid \ldots \mid P_N & \text{Parallel} \\
 & | & \nu x\, P & \text{Restriction} \\
 & | & X(\vec{n}) & \text{Instance}
\end{array}
$$

$$\Gamma ::= X_1(\vec{m}_1) \triangleq P_1, \ldots, X_N(\vec{m}_N) \triangleq P_N \quad \text{Definitions}, \mathrm{fn}(P_i) \subseteq \vec{m}_i$$

The SPiM Programming Language (v0.04)

$$
\begin{array}{lcll}
Dec & ::= & \mathbf{new}\ x\{\texttt{@}r\}\texttt{:}t & \text{Channel Declaration} \\
 & | & \mathbf{type}\ n\ \texttt{=}\ t & \text{Type Declaration} \\
 & | & \mathbf{val}\ m\ \texttt{=}\ v & \text{Value Declaration} \\
 & | & \mathbf{run}\ P & \text{Process Declaration} \\
 & | & \mathbf{let}\ D_1\ \mathbf{and}\ \ldots\ \mathbf{and}\ D_N & \text{Definitions, } N \geq 1 \\
 & & & \\
D & ::= & X\texttt{(}m_1\texttt{,}\ldots\texttt{,}m_N\texttt{)}\ \texttt{=}\ P & \text{Definition, } N \geq 0 \\
 & & & \\
P & ::= & \texttt{()} & \text{Null Process} \\
 & | & \texttt{(}P_1\ \texttt{|}\ \ldots\ \texttt{|}\ P_M\texttt{)} & \text{Parallel, } M \geq 2 \\
 & | & X\texttt{(}v_1\texttt{,}\ldots\texttt{,}v_N\texttt{)} & \text{Instantiation, } N \geq 0 \\
 & | & \pi\{\texttt{;}\ P\} & \text{Action} \\
 & | & \mathbf{do}\ \pi_1\{\texttt{;}\ P_1\}\ \texttt{or}\ \ldots\ \texttt{or}\ \pi_M\{\texttt{;}\ P_M\} & \text{Choice, } M \geq 2 \\
 & | & \texttt{(}Dec_1 \ldots Dec_N\ P\texttt{)} & \text{Declarations, } N \geq 0 \\
 & & & \\
\pi & ::= & \texttt{!}x\ \{\texttt{(}v_1\texttt{,}\ldots\texttt{,}v_N\texttt{)}\} & \text{Output, } N \geq 0 \\
 & | & \texttt{?}x\ \{\texttt{(}m_1\texttt{,}\ldots\texttt{,}m_N\texttt{)}\} & \text{Input, } N \geq 0 \\
 & | & \texttt{delay@}r & \text{Delay}
\end{array}
$$

The Graphical Stochastic Pi-Calculus (GSPi)

A normal form for SPi, with each summation or guarded process as a definition:

$$
\begin{array}{rcll}
\pi ::= & & ?x(\vec{m}) & \text{Input} \\
& \mid & !x(\vec{n}) & \text{Output} \\
& \mid & \tau_r & \text{Delay} \\
P, Q ::= & & P_1 \mid \ldots \mid P_N & \text{Parallel} \\
& \mid & \nu x\, P & \text{Restriction} \\
& \mid & X(\vec{n}) & \text{Instance}
\end{array}
$$

$$
\begin{array}{rcl}
\Gamma ::= & X(\vec{m}) \triangleq \nu x_1 ... \nu x_M\, (\pi_1.X_1(\vec{n}_1) + \ldots + \pi_N.X_N(\vec{n}_N)) & \text{Summation} \\
\mid & X(\vec{m}) \triangleq \nu x_1 ... \nu x_M\, (X_1(\vec{n}_1) \mid \ldots \mid X_N(\vec{n}_N)) & \text{Composition}
\end{array}
$$

Graphical Representation: Definitions

➢ A collection of mutually recursive definitions:

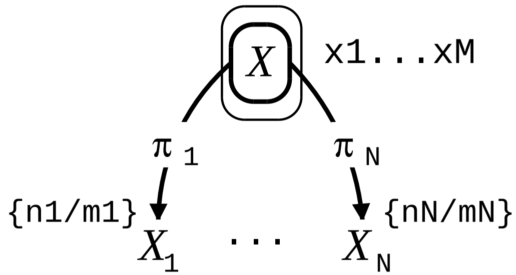
$$X_1(m_1) \triangleq C_1, \ldots, X_N(m_N) \triangleq C_N$$

➢ Displayed as a directed graph with nodes $X_1 \ldots X_N$ and with edges between these nodes.

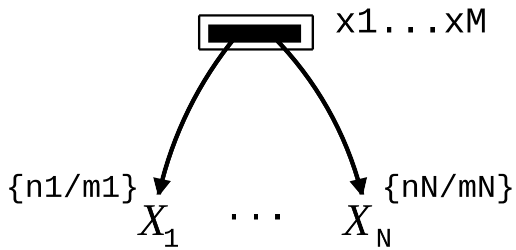
➢ Each definition $X(m) \triangleq C$ displayed as a node X with zero or more edges to subsequent nodes .

Graphical Representation: Definitions

$$X(\vec{m}) \triangleq \nu x_1 ... \nu x_M \, (\pi_1.X_1(\vec{n}_1) + \ldots + \pi_N.X_N(\vec{n}_N))$$

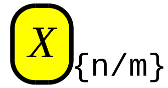


$$X(\vec{m}) \triangleq \nu x_1 ... \nu x_M \, (X_1(\vec{n}_1) \mid \ldots \mid X_N(\vec{n}_N))$$

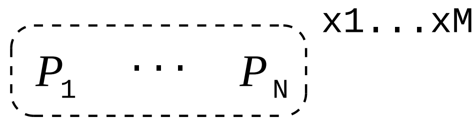


Graphical Representation: Processes

$X(\vec{n})$

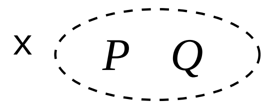


$\nu x_1 ... \nu x_M \, (P_1 \mid \ldots \mid P_N)$

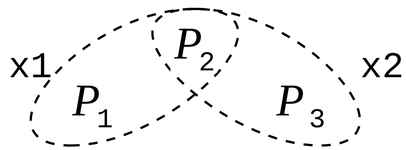


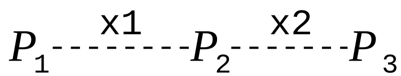
Restriction as Complexation:

A complex of P and Q modelled as a restriction $\nu x \, (P \mid Q)$



$\nu x_1 \, \nu x_2 \, (P_1 \mid P_2 \mid P_3)$,$x_1 \notin \mathrm{fn}(P_3)$, $x_2 \notin \mathrm{fn}(P_1)$





Graphical Reduction: Execution Model

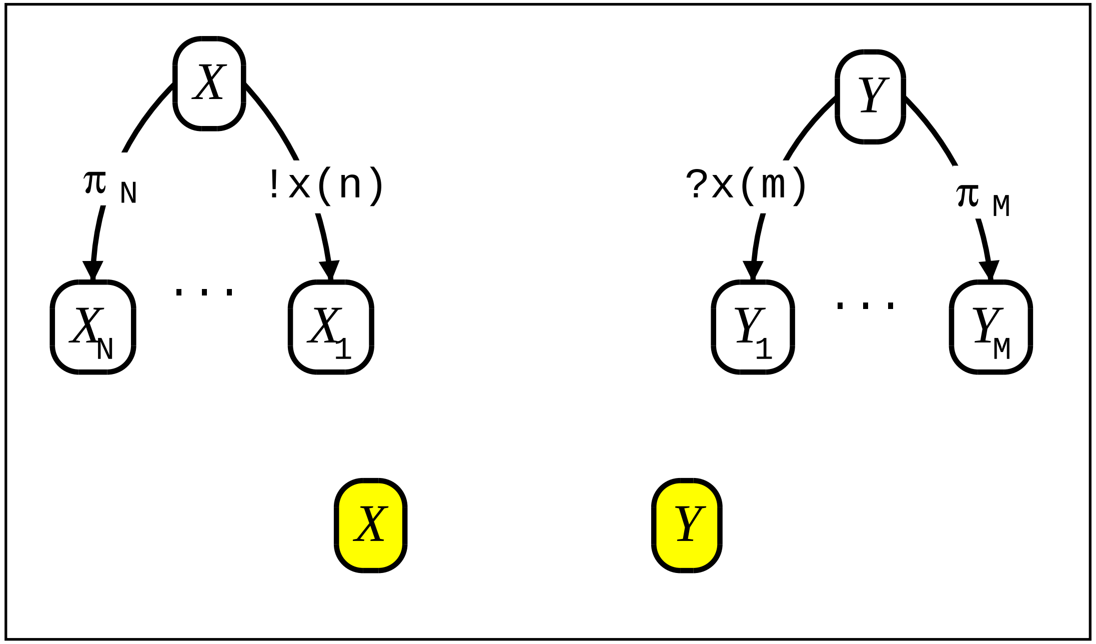
Reduction in SPi:

$$
\begin{array}{rcll}
!x(\vec{n}).P+\Sigma \mid ?x(\vec{m}).Q+\Sigma' & \xrightarrow{rate(x)} & P \mid Q_{\{\vec{n}/\vec{m}\}} & (1)\\
\tau_r.P+\Sigma & \xrightarrow{r} & P & (2)\\
P \xrightarrow{r} P' \quad \Rightarrow \quad P \mid Q & \xrightarrow{r} & P' \mid Q & (3)\\
P \xrightarrow{r} P' \quad \Rightarrow \quad \nu x\, P & \xrightarrow{r} & \nu x\, P' & (4)\\
Q \equiv P \xrightarrow{r} P' \equiv Q' \quad \Rightarrow \quad Q & \xrightarrow{r} & Q' & (5)
\end{array}
$$

Reduction in $\mathrm{GSPi} \subset \mathrm{SPi}$:

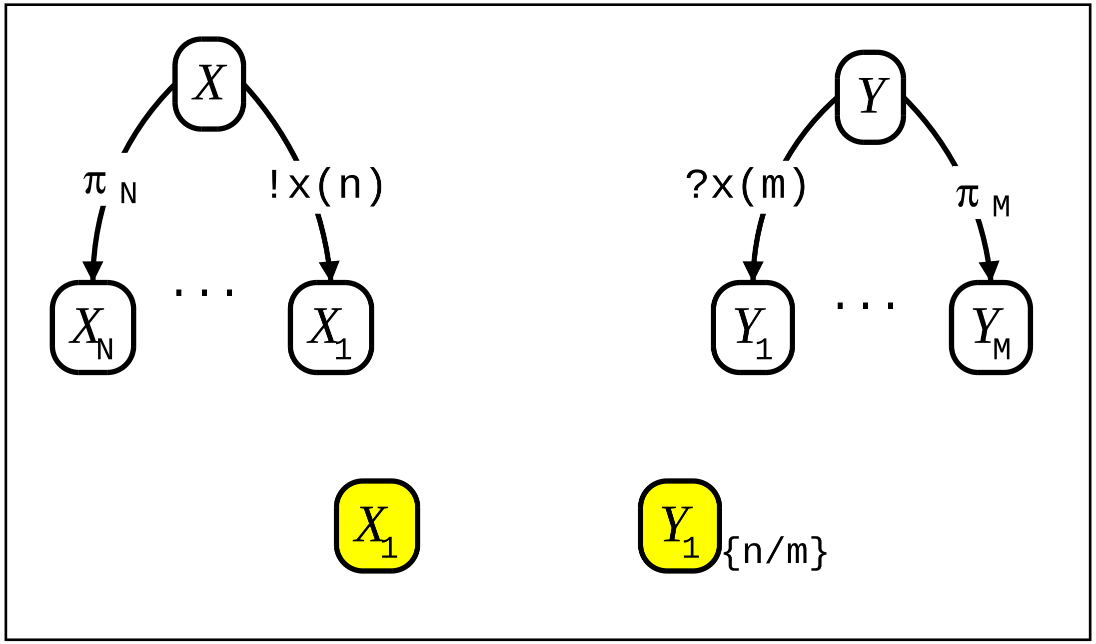
Proposition 1. $\forall P \in \mathrm{GSPi}.P \xrightarrow{r} P' \Rightarrow \exists P'' \in \mathrm{GSPi}.P' \equiv P''$

Graphical Reduction: Communication



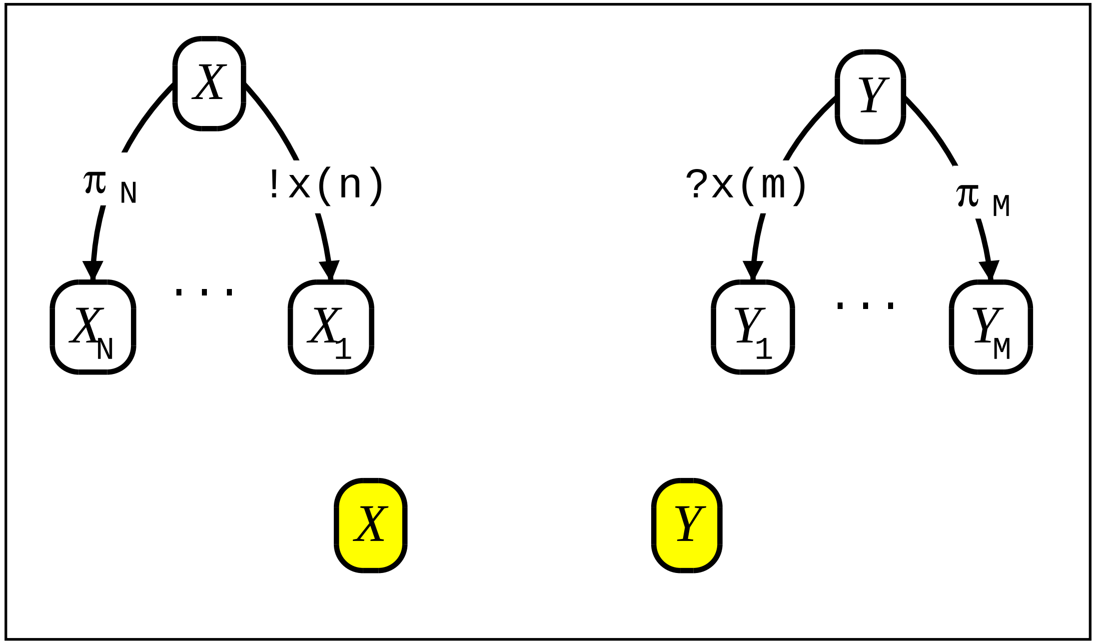
$$\frac{X(\vec{z}) \triangleq\ !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z})\ ,\ Y \triangleq\ ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})}{\underline{X(\vec{z}) \mid Y(\vec{z})} \xrightarrow{rate(x)} X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}}}$$

Graphical Reduction: Communication



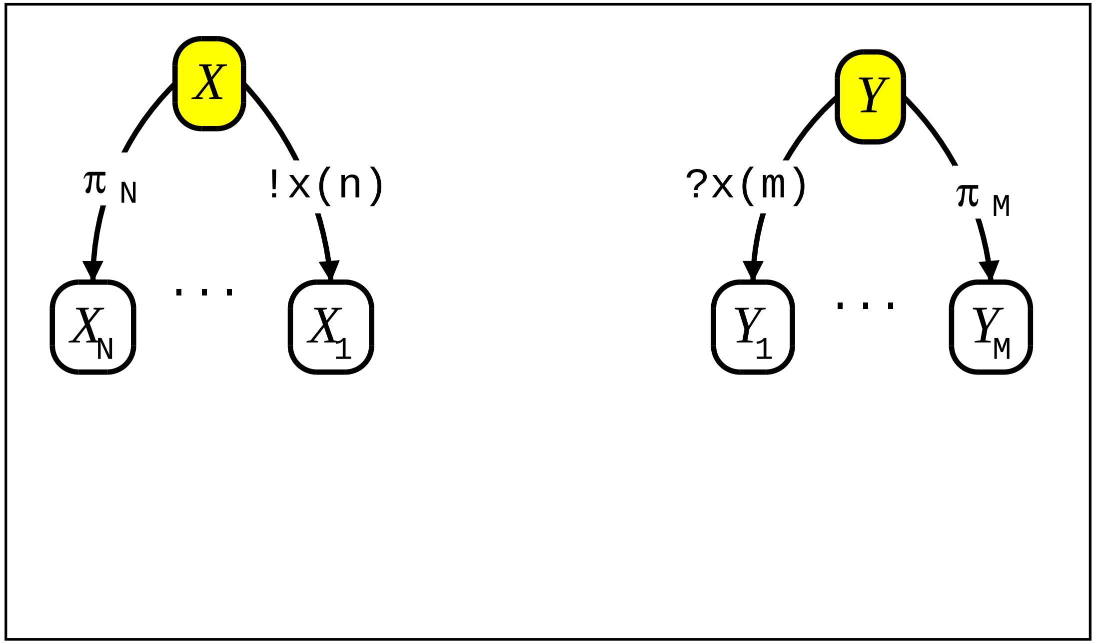
$$\frac{X(\vec{z}) \triangleq\ !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z})\ ,\ Y \triangleq\ ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})}{X(\vec{z}) \mid Y(\vec{z}) \xrightarrow{rate(x)} \underline{X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}}}}$$

Graphical Reduction: Communication



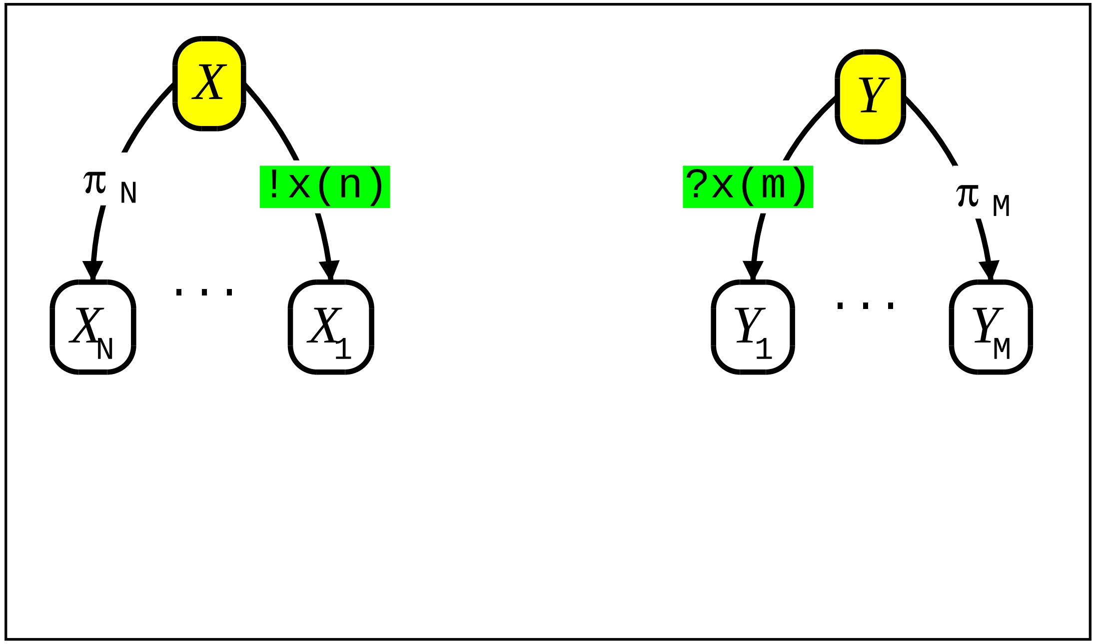
$$\frac{X(\vec{z}) \triangleq\ !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z})\ ,\ Y \triangleq\ ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})}{\underline{X(\vec{z}) \mid Y(\vec{z})} \xrightarrow{rate(x)} X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}}}$$

Inline Graphical Reduction: Communication



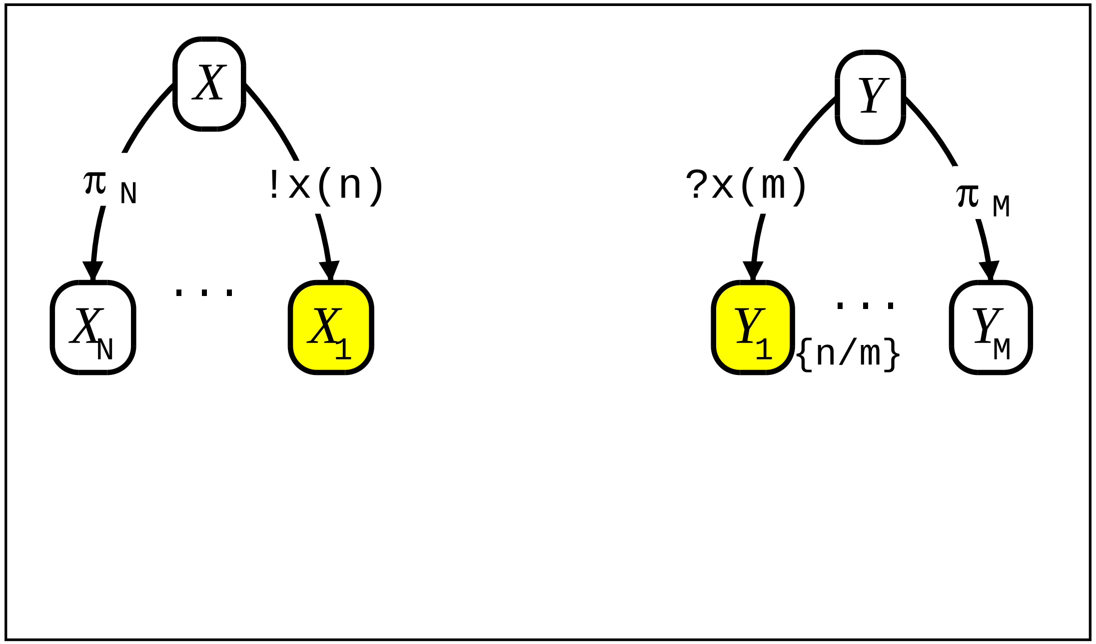
$$\frac{X(\vec{z}) \triangleq\ !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z})\ ,\ Y \triangleq\ ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})}{\underline{X(\vec{z}) \mid Y(\vec{z})} \xrightarrow{rate(x)} X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}}}$$

Inline Graphical Reduction: Communication



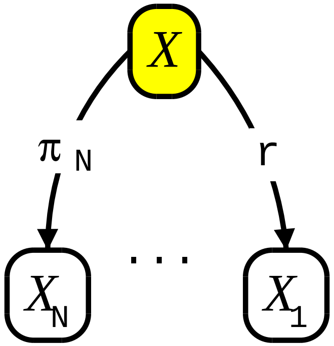
$$\frac{X(\vec{z}) \triangleq\ !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z})\ ,\ Y \triangleq\ ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})}{\underline{X(\vec{z}) \mid Y(\vec{z})} \xrightarrow{rate(x)} X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}}}$$

Inline Graphical Reduction: Communication



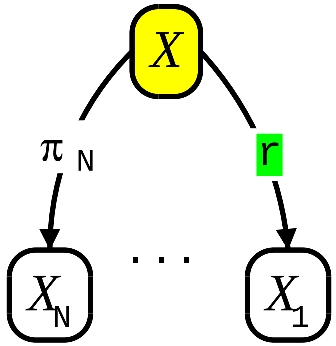
$$\frac{X(\vec{z}) \triangleq !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z}) \ , \ Y \triangleq ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})}{X(\vec{z}) \mid Y(\vec{z}) \xrightarrow{rate(x)} \underline{X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}}}}$$

Inline Graphical Reduction: Delay



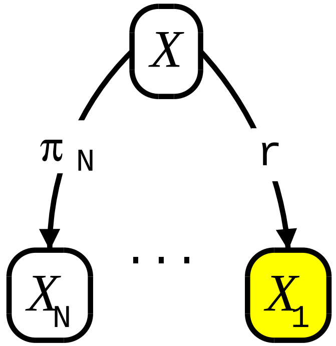
$$\frac{X(\vec{z}) \triangleq \tau_r.X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z})}{\underline{X(\vec{z})} \xrightarrow{r} X_1(\vec{z})}$$

Inline Graphical Reduction: Delay



$$\frac{X(\vec{z}) \triangleq \tau_r.X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z})}{\underline{X(\vec{z})} \xrightarrow{r} X_1(\vec{z})}$$

Inline Graphical Reduction: Delay



$$\frac{X(\vec{z}) \triangleq \tau_r.X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z})}{X(\vec{z}) \xrightarrow{r} \underline{X_1(\vec{z})}}$$

Graphical Representation: Benefits

➢ Static graphical representation used to:

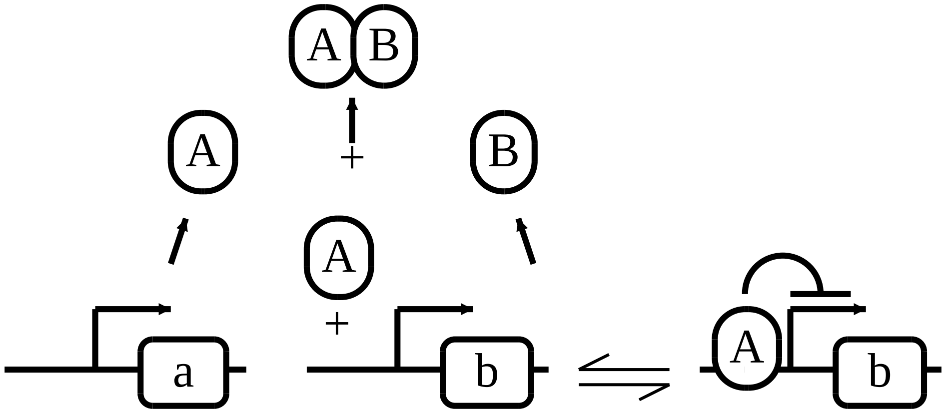
- Clarify the connectivity between process definitions
- Highlight cycles, which are key to many biological systems.

➢ Dynamic graphical representation used to:

- Visualise the execution trace of a model
- Clarify the overall system function
- Graphically debug pi-calculus code.

Evolved Gene Network [Francois and Hakim, 2004]

➢ Gene networks are evolved in silico to perform specific functions, e.g.:

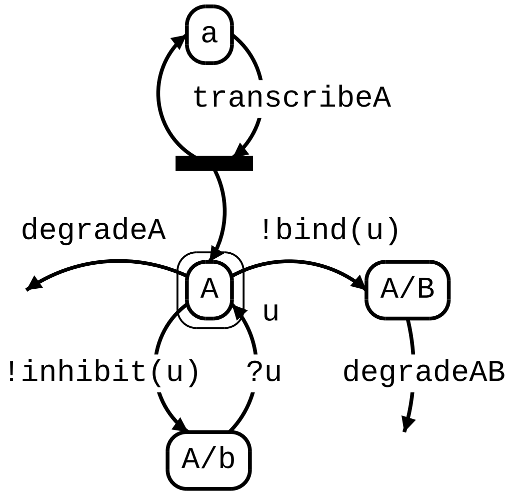


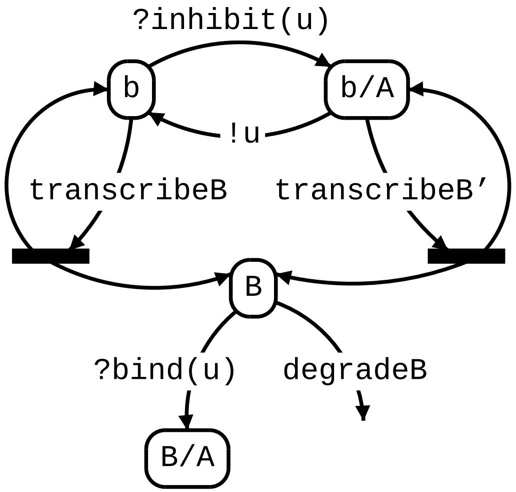
➢ Genes a and b can produce proteins A and B respectively:

- ❑ A and B can bind irreversibly to produce AB, which eventually degrades.
- ❑ A can also bind reversibly to gene b, slowing the transcription of B.

➢ What is the function of this system?

Evolved Gene Network: Definitions





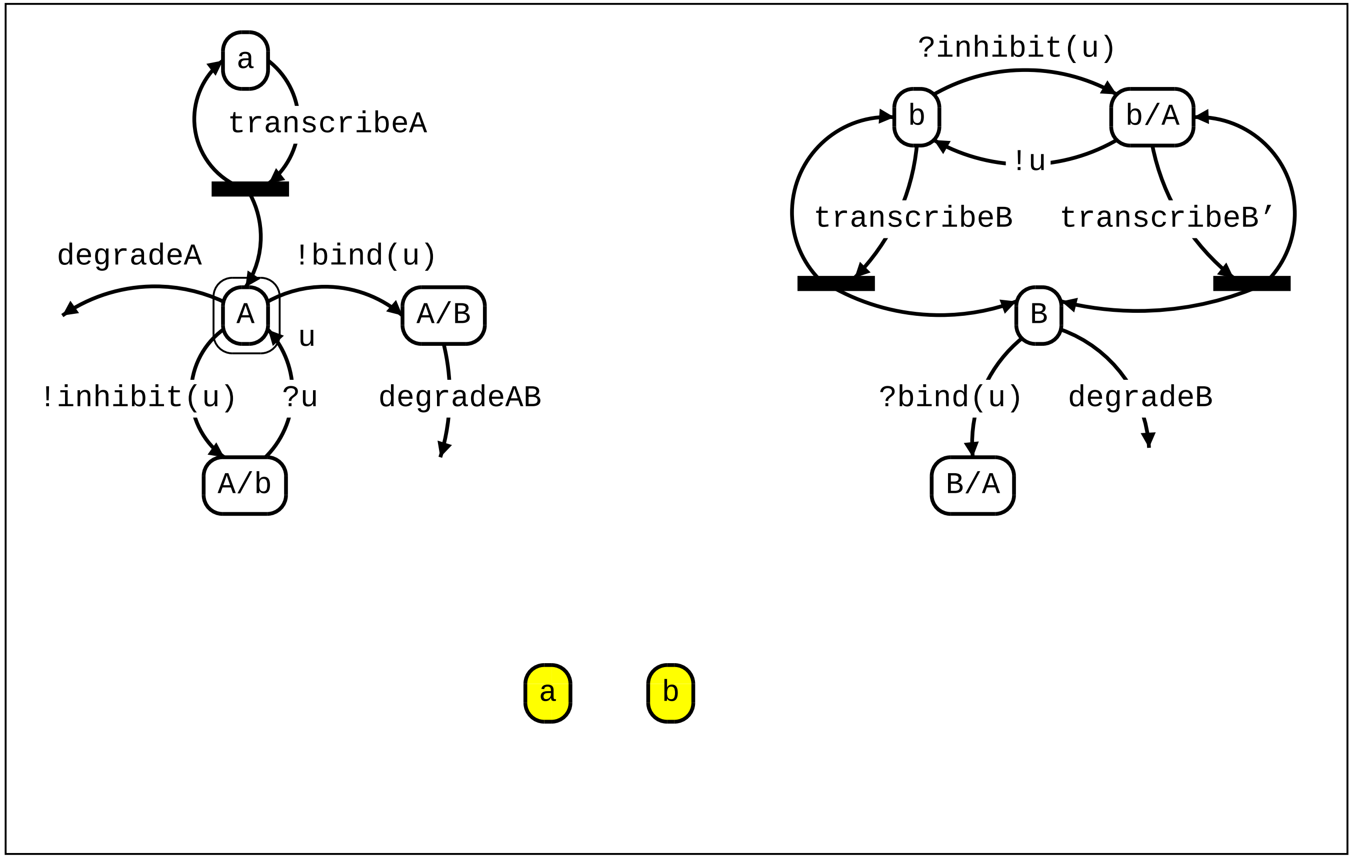
$$
\begin{aligned}
a(\vec{z}) &\triangleq \tau_{transcribeA}.(A() \mid a())\\
A(\vec{z}) &\triangleq \nu u\,(\tau_{degradeA}\\
&\qquad +!bind(u).AB(u)\\
&\qquad +!inhibit(u).Ab(u))\\
Ab(u) &\triangleq ?u.A()\\
AB(u) &\triangleq \tau_{degradeAB}
\end{aligned}
$$

$$
\begin{aligned}
b(\vec{z}) &\triangleq \tau_{transcribeB}.(B() \mid b())\\
&\quad +?inhibit(u).bA(u)\\
bA(u) &\triangleq \tau_{transcribeB'}.(B() \mid bA(u))\\
&\quad +!u.b()\\
B(\vec{z}) &\triangleq \tau_{degradeB}\\
&\quad +?bind(u).BA(u)
\end{aligned}
$$

Evolved Gene Network: SPiM Code

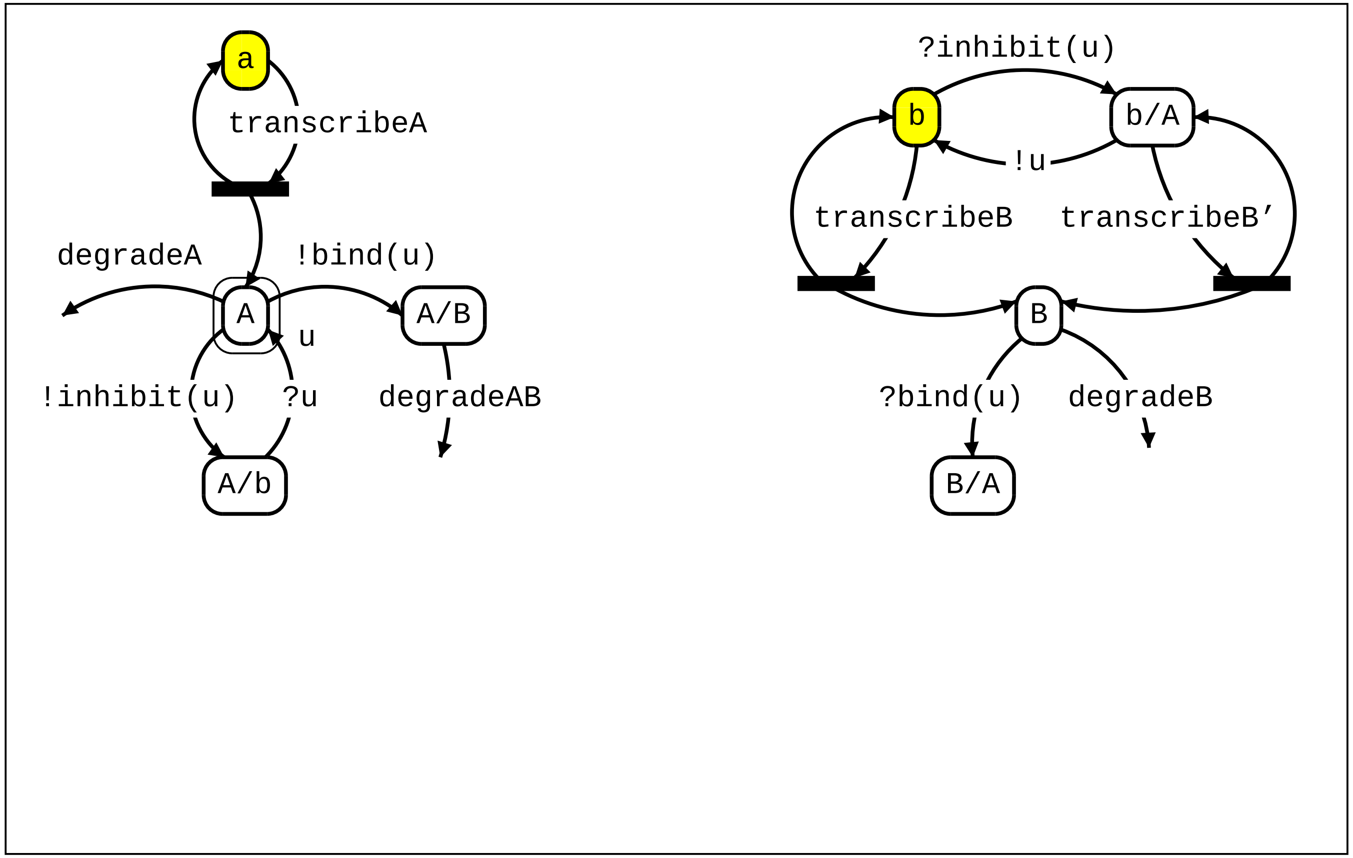
```
let a() = delay@transcribeA; ( A() | a() )
and A() = (
  new u@0.42:chan
  do delay@degradeA
  or !bind; A_B()
  or !inhibit(u); A_b(u))
and A_b(u:chan) = ?u; A()
and A_B() = delay@degradeAB
let b() =
  do delay@transcribeB; ( B() | b() )
  or ?inhibit(u); b_A(u)
and b_A(u:chan) =
  do !u; b()
  or delay@transcribeB'; B(); b_A(u)
and B() =  do delay@degradeB  or ?bind
run (a() | b())
```

Evolved Gene Network



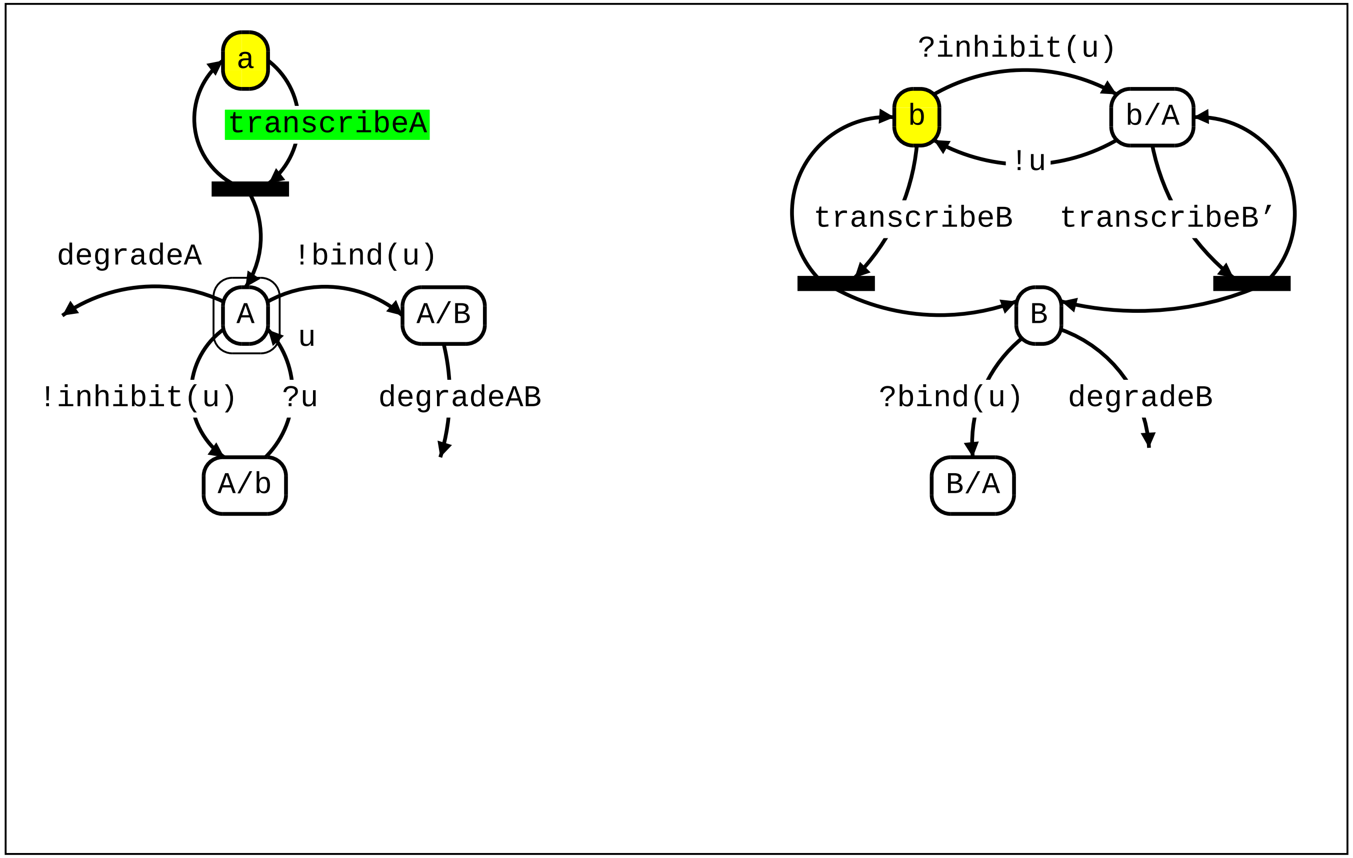
Initially there is one copy of each gene, a and b

Evolved Gene Network



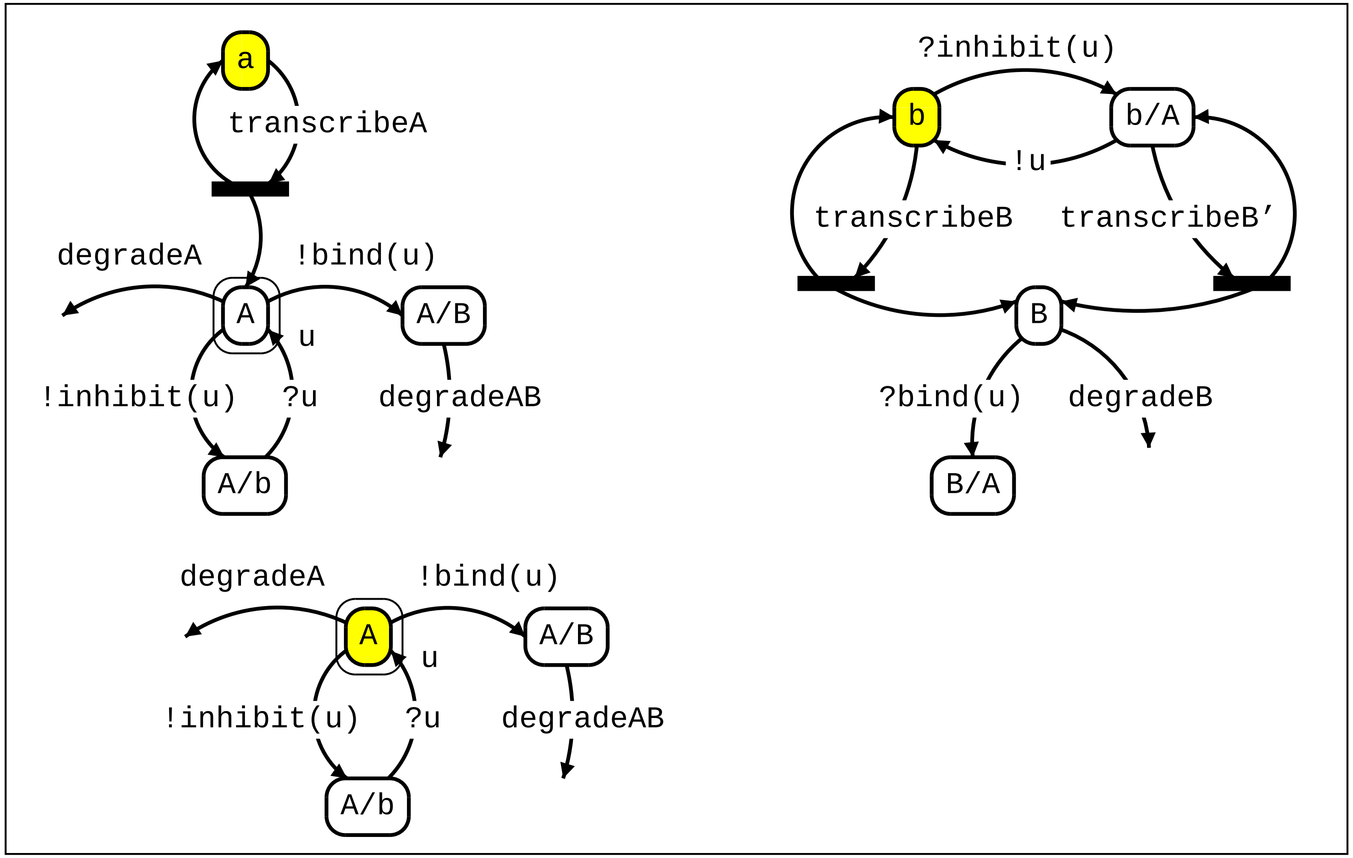
Represent the behaviour of each gene as a separate graph

Evolved Gene Network



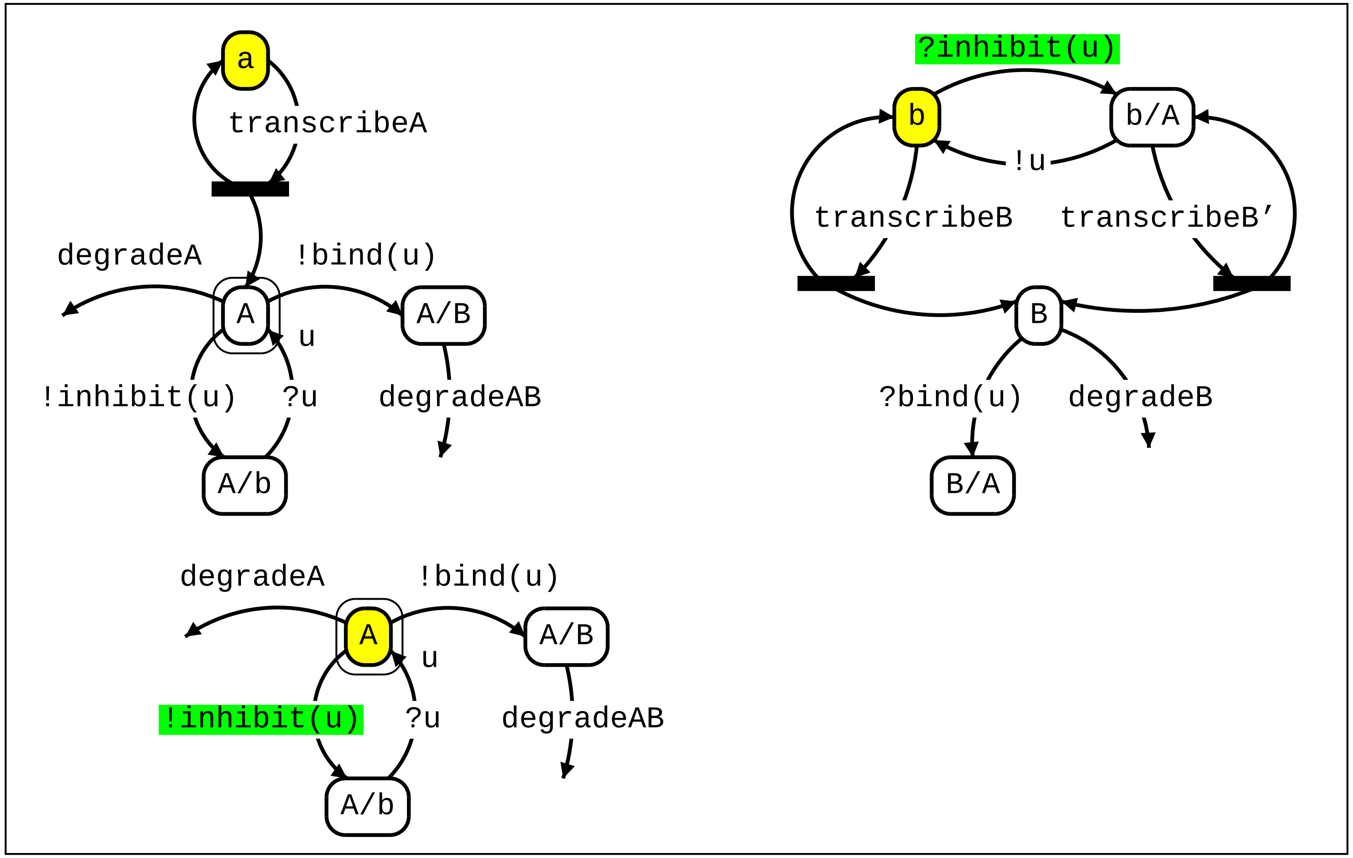
Gene a can transcribe a new protein A at rate $transcribeA$

Evolved Gene Network



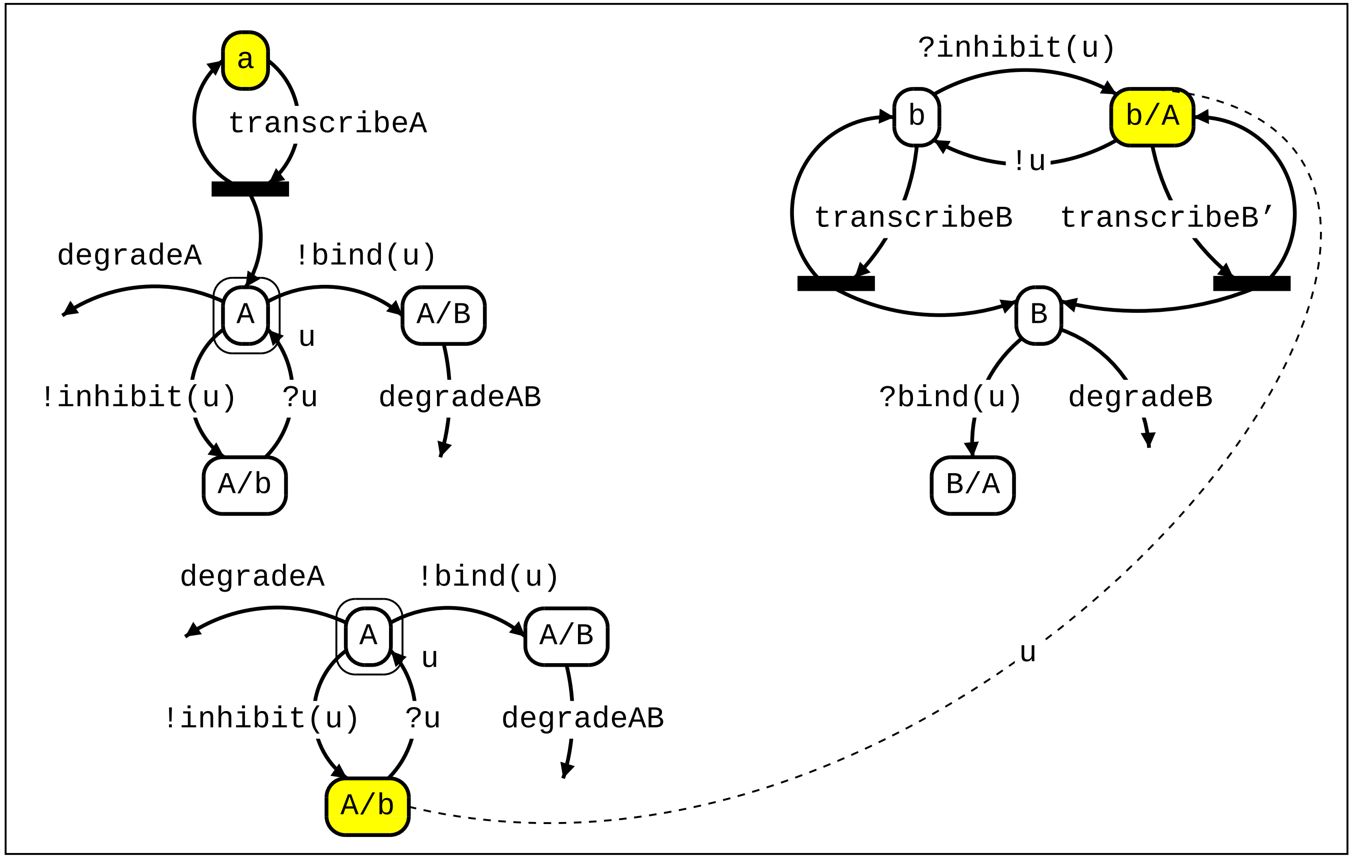
A new protein A is transcribed

Evolved Gene Network



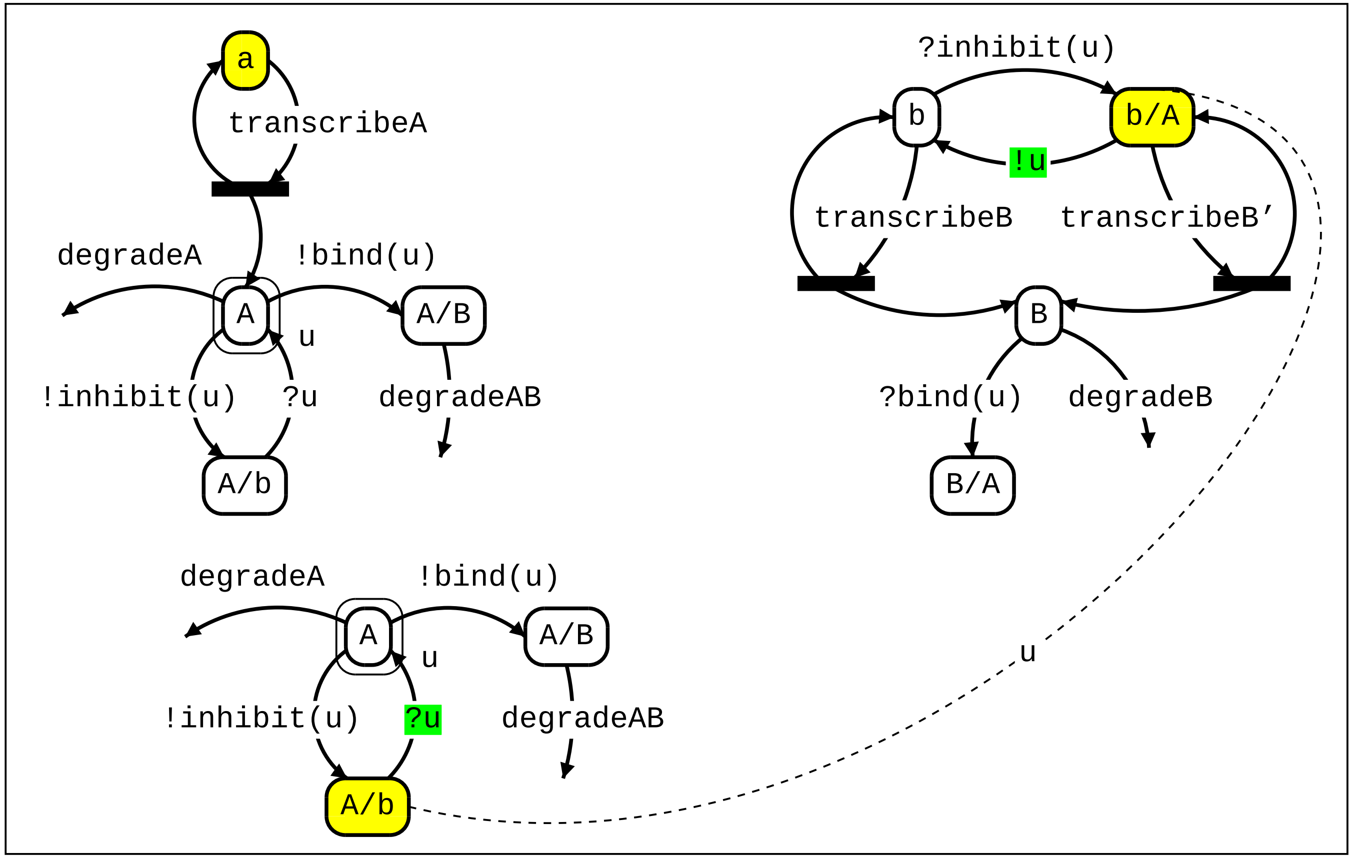
Protein A can bind to gene b to inhibit production of protein B

Evolved Gene Network



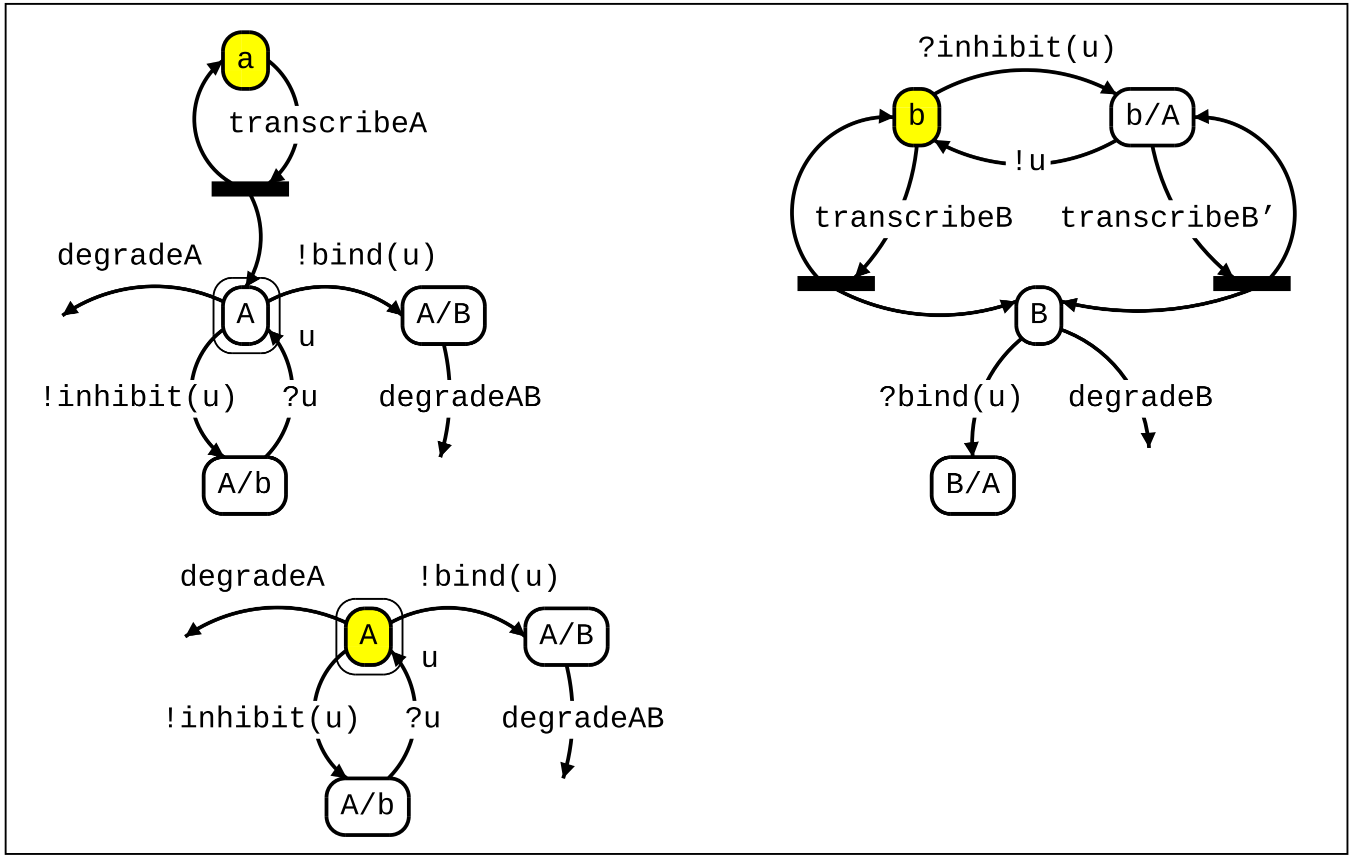
Protein A is bound to gene b by the private channel u

Evolved Gene Network



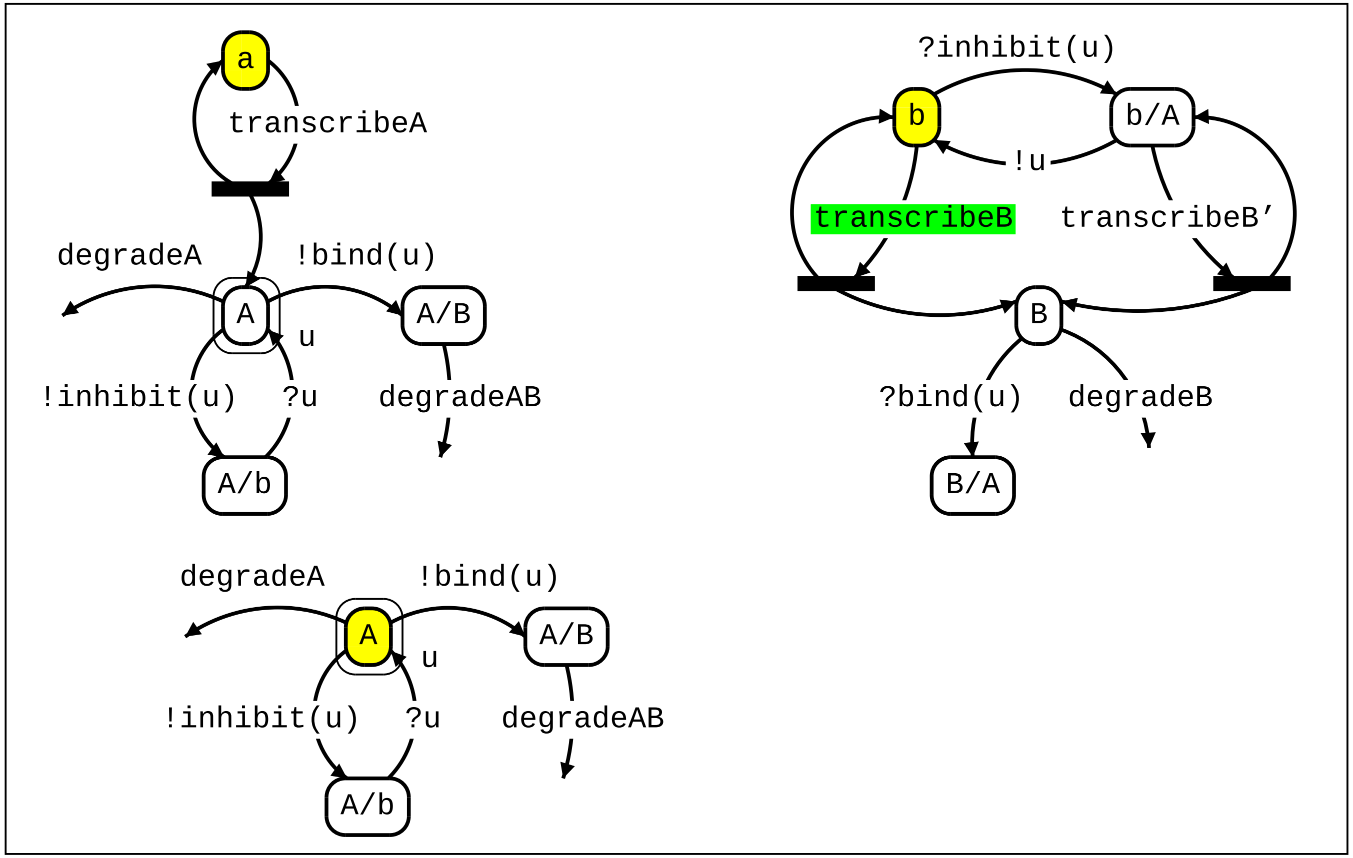
Protein A can unbind from gene b using channel u

Evolved Gene Network



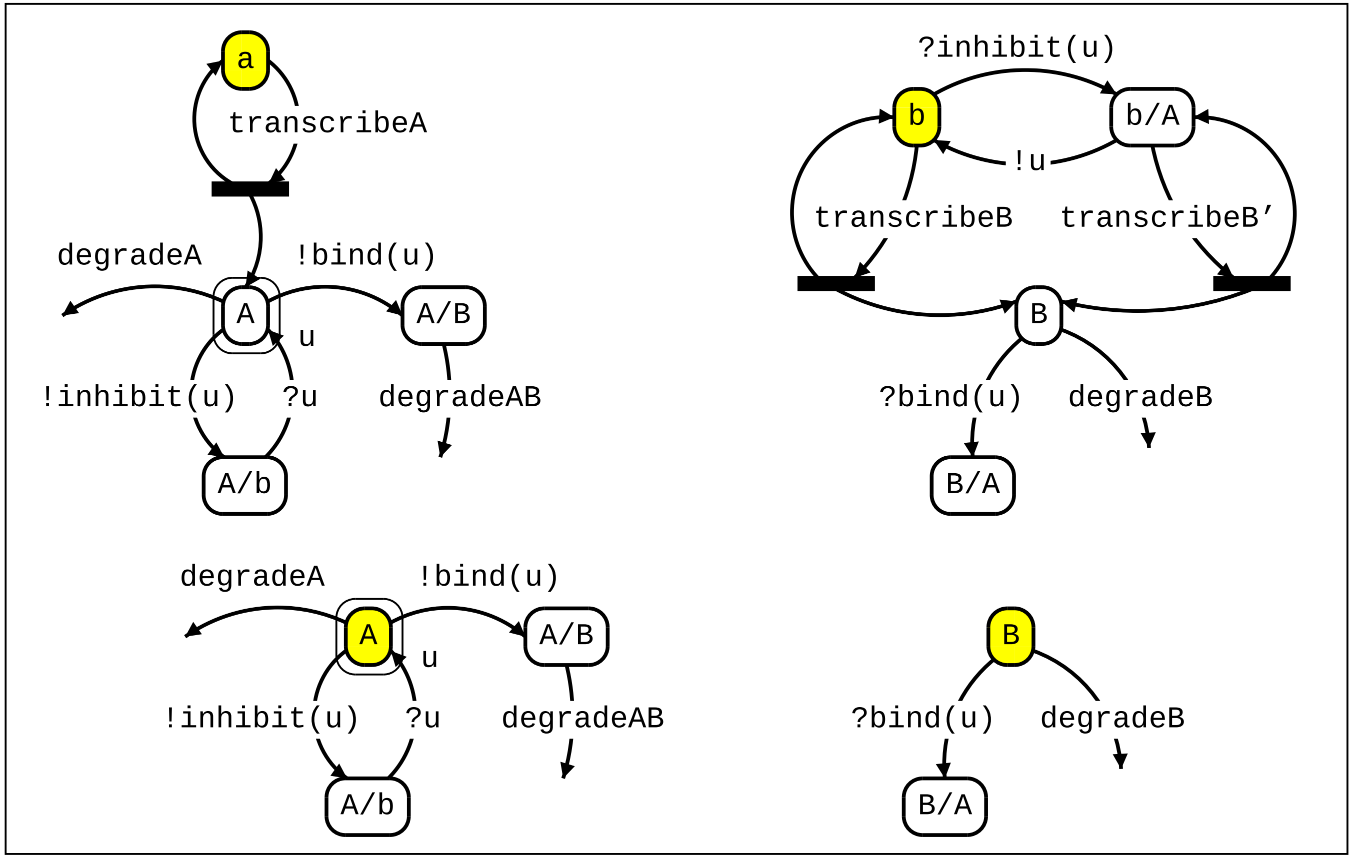
Protein A is no longer bound to gene b

Evolved Gene Network



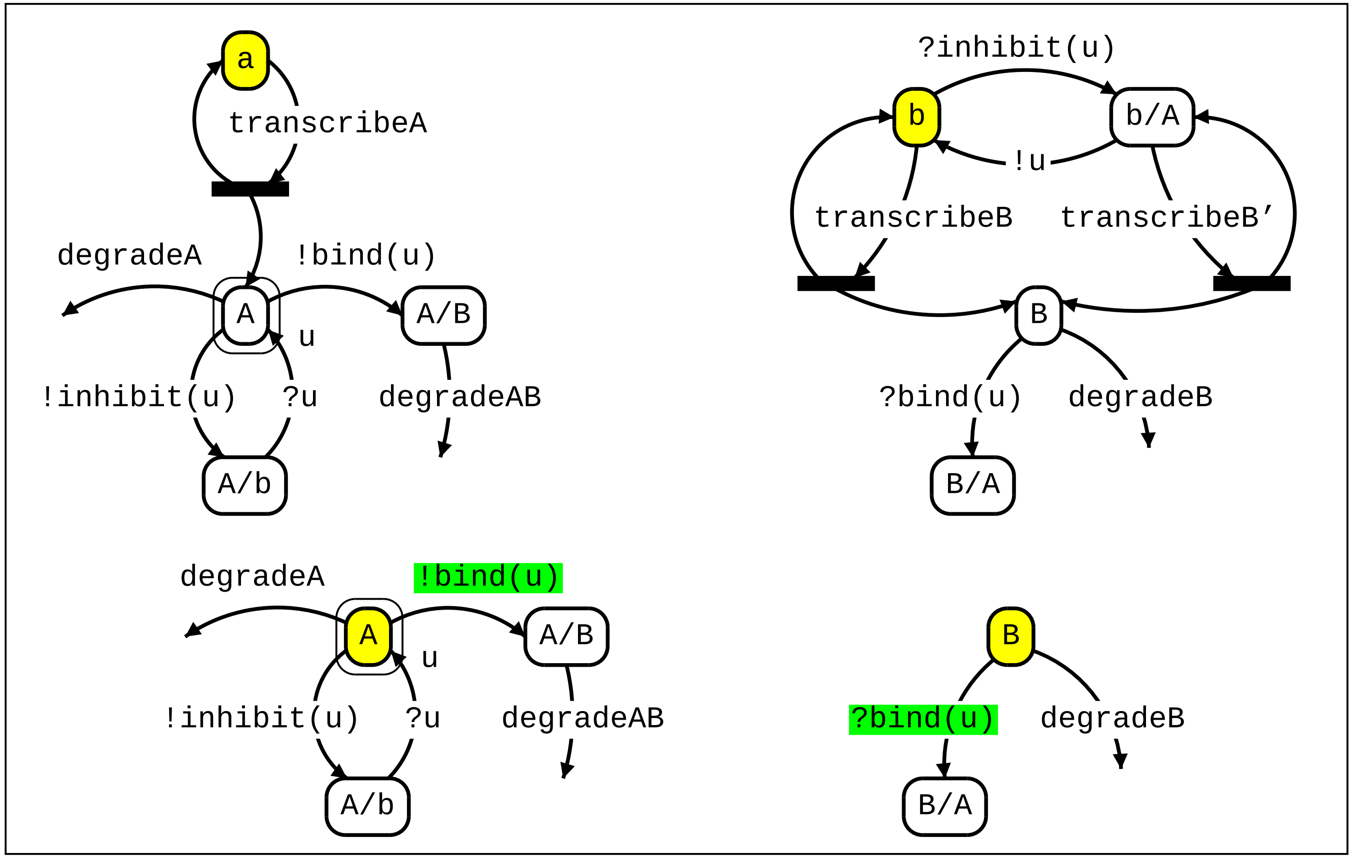
Gene b can transcribe a new protein B with rate $transcribeB$

Evolved Gene Network



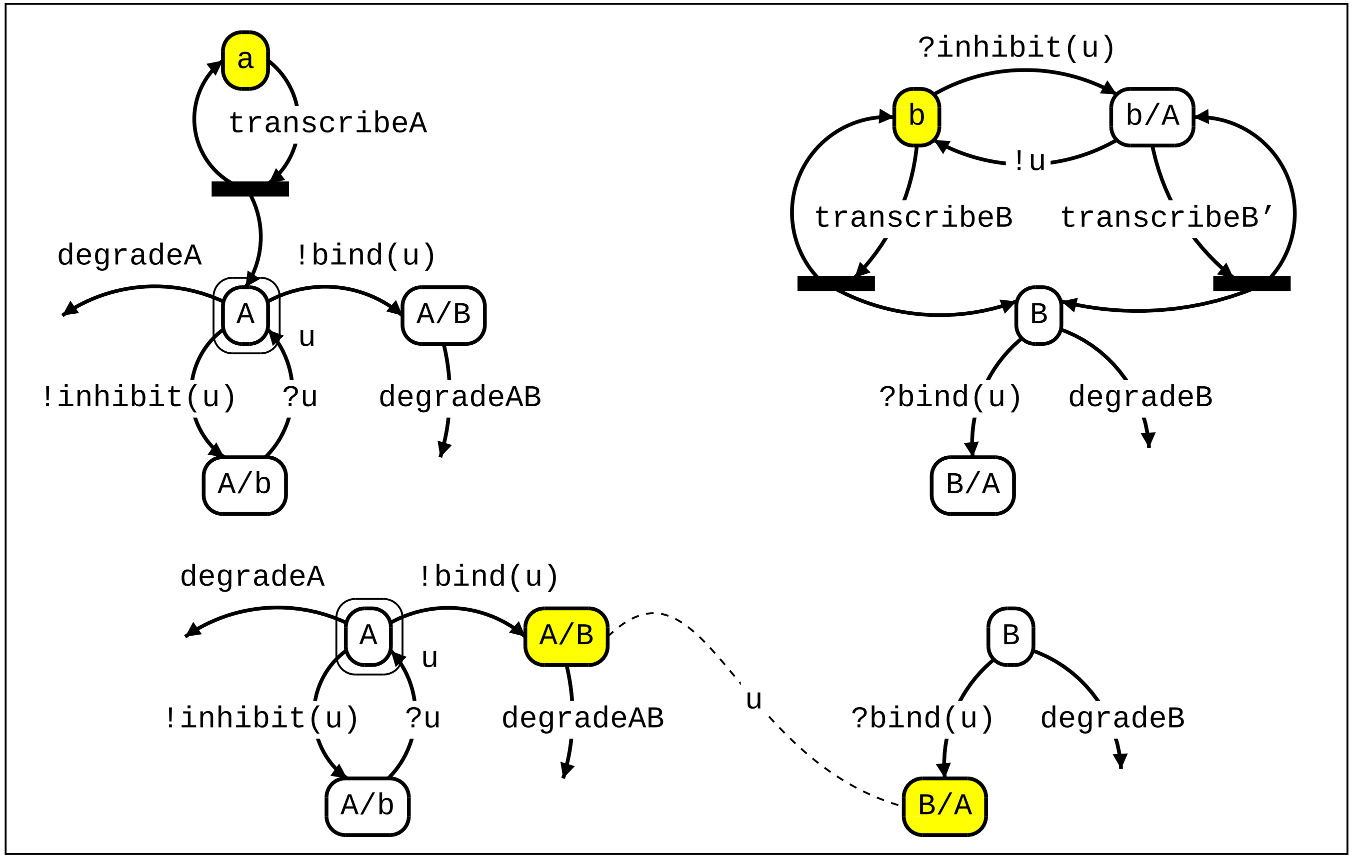
A new protein B is transcribed

Evolved Gene Network



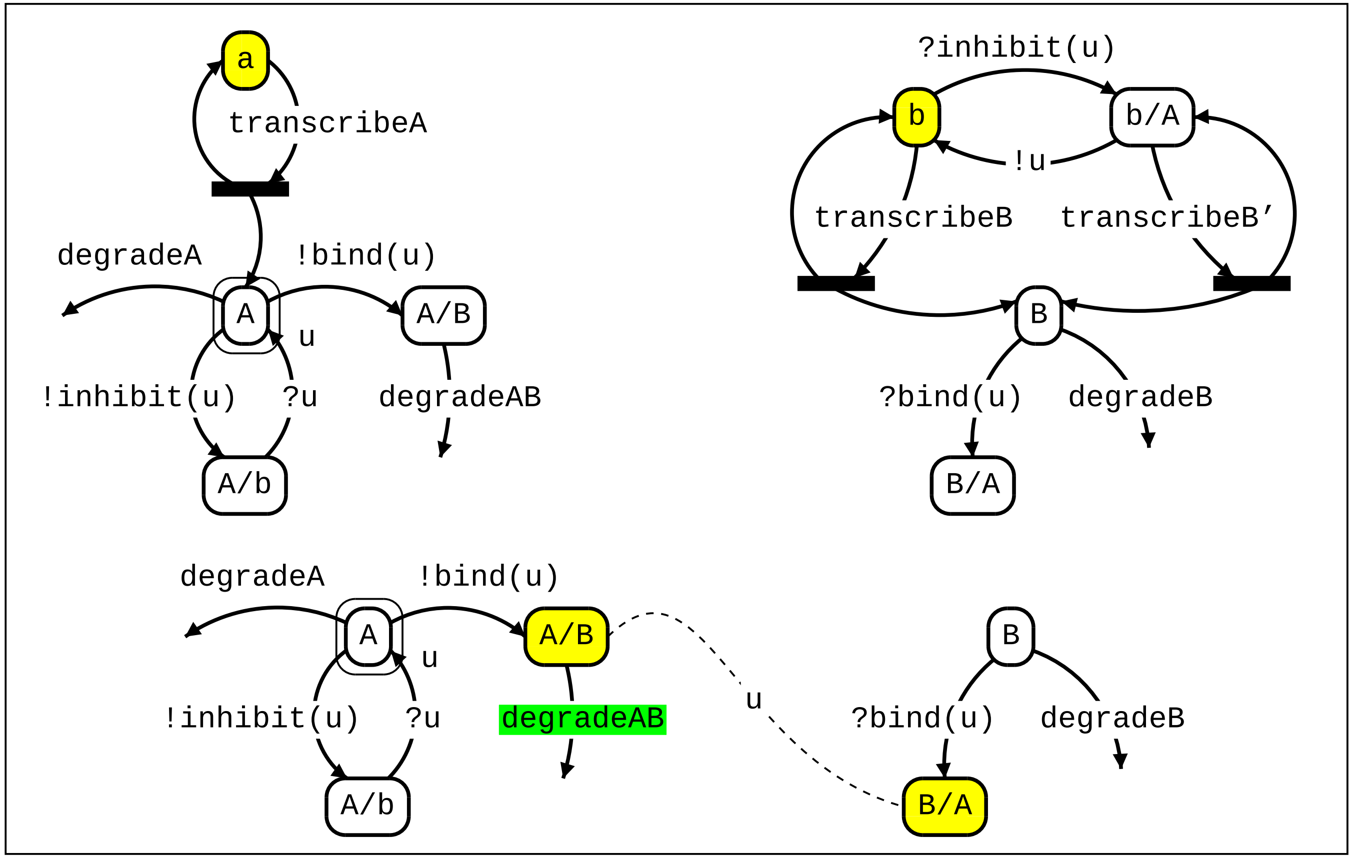
Protein A can bind with protein B

Evolved Gene Network



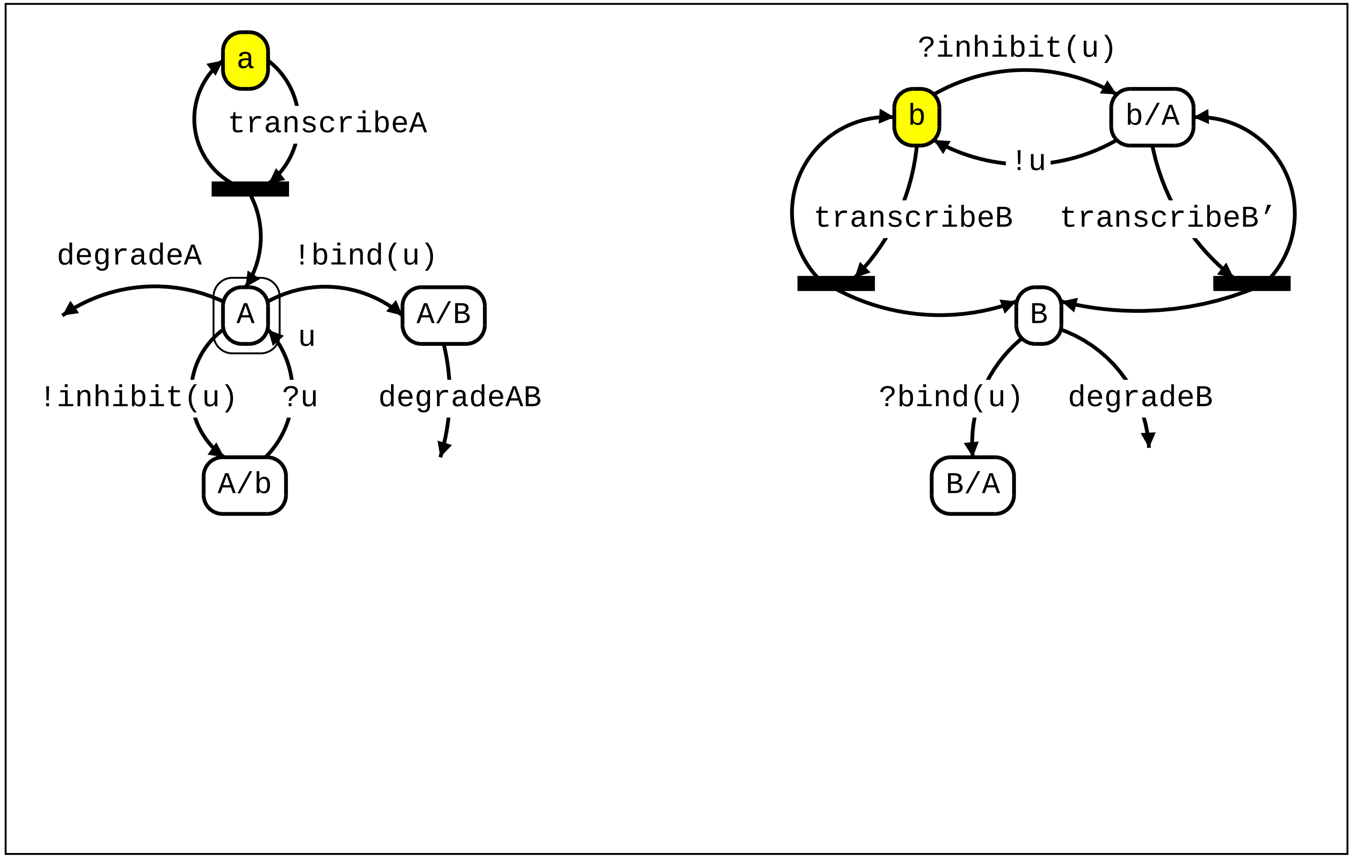
Protein A and B are irreversibly bound

Evolved Gene Network



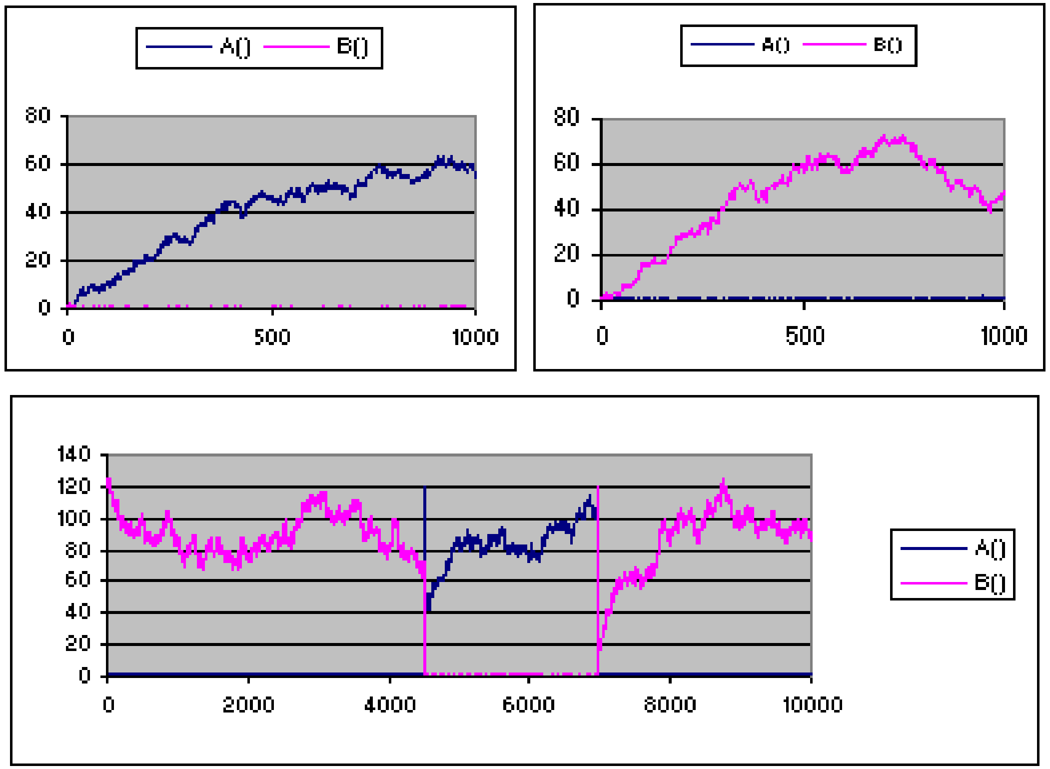
Complex AB can be degraded

Evolved Gene Network

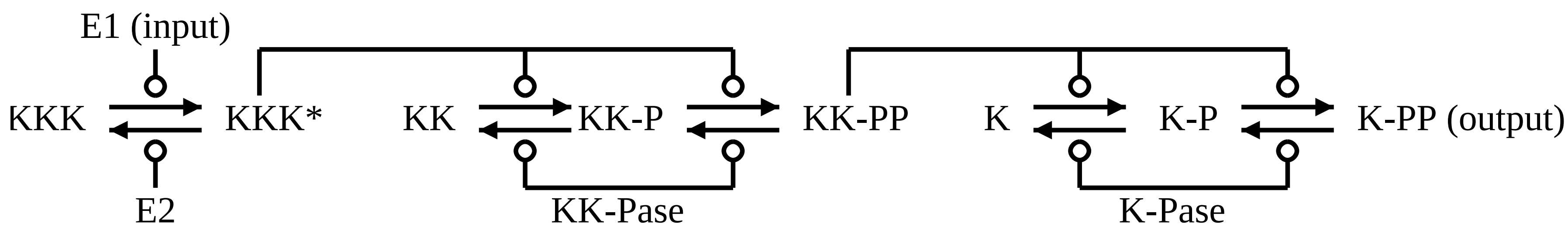


Complex AB has been degraded

Evolved Gene Network: Simulation Results



Mapk Cascade [Huang and Ferrel, 1996]



- ➢ System originally described using a set of reaction equations
  - ❑ Converted to ordinary differential equations, which were solved numerically
  - ❑ Response curves shown to be steeply sigmoidal (≃Hill 5).

- ➢ System functions as follows:
  - ❑ The enzyme E1 drives the transformation from KKK to KKK*
  - ❑ KKK* drives the transformation from KK to KK-P to KK-PP
  - ❑ KK-PP drives the transformation from K to K-P to K-PP

Mapk Cascade: Equations

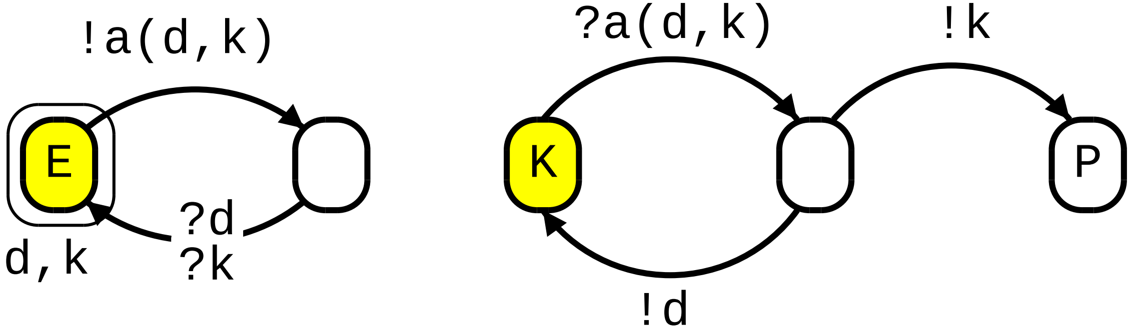
Reaction Equation:

$$E + K \ {}_{d}\!\overset{a}{\rightleftarrows} \ E{\cdot}K \ \overset{k}{\rightarrow} \ E + P$$

Pi-calculus Processes:

$$\begin{array}{rcl} E(a) & \triangleq & \nu d\, \nu k\, !a(d,k).(?d.E(a) \ + \ ?k.E(a)) \\ K(a) & \triangleq & ?a(d,k).(!d.K(a) \ + \ !k.P()) \end{array}$$

Graphical Representation:

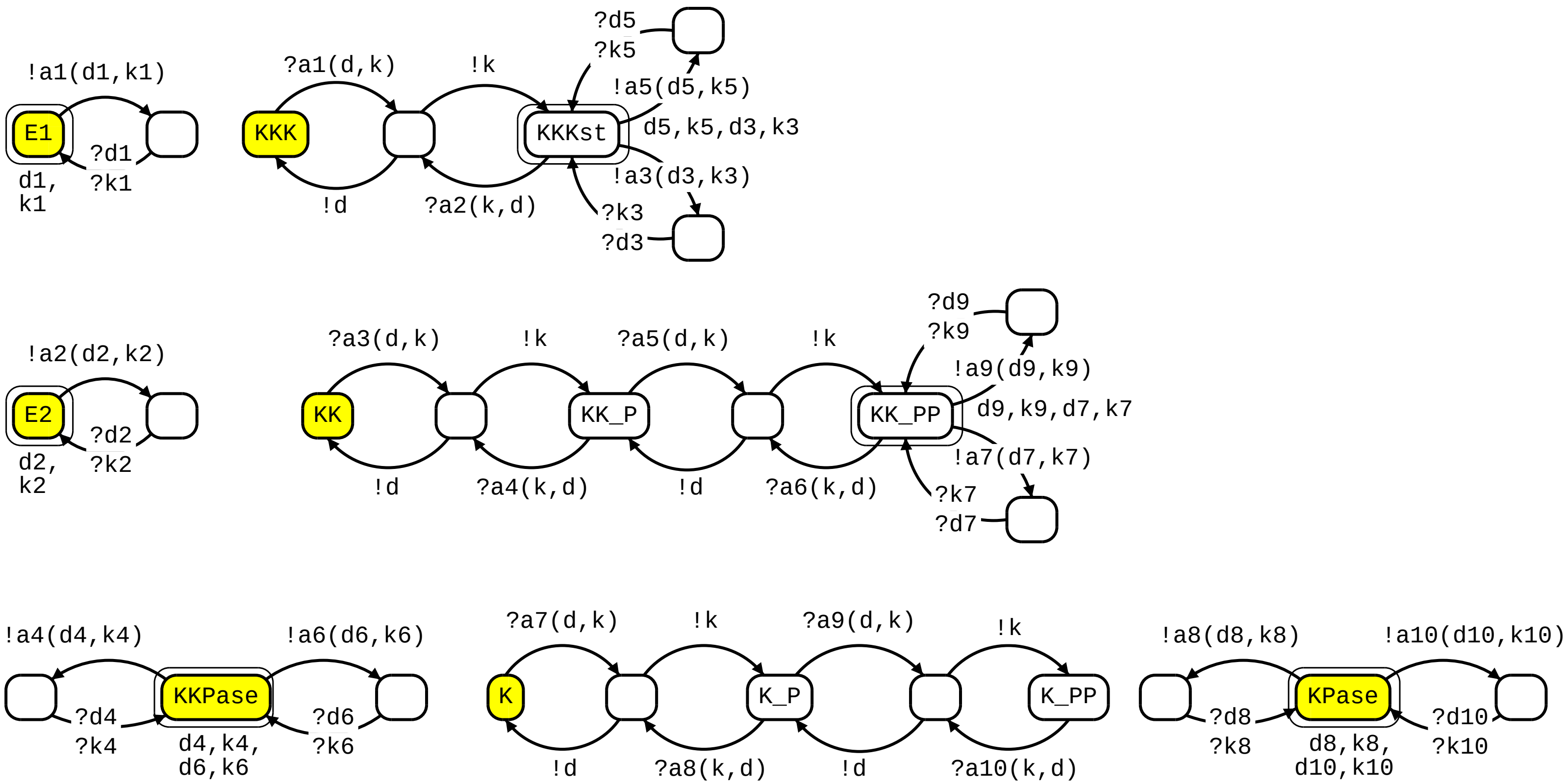


Mapk Cascade: SPiM Code

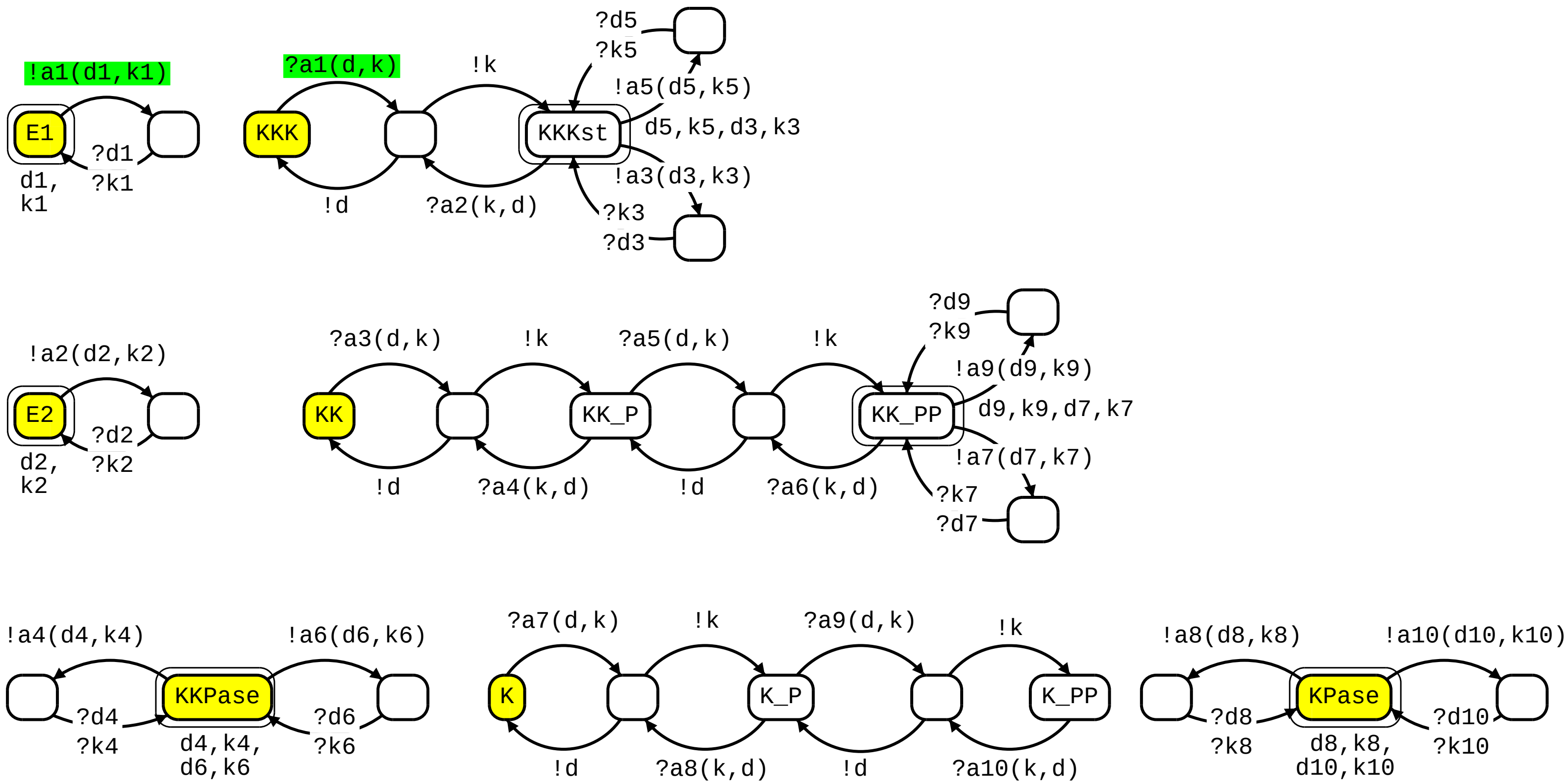
```
let E1() = (
  new k1@rk1:chan new d1@rd1:chan
  !a1(d1,k1); do ?d1;E1() or ?k1;E1()
)
let E2() = (
  new k2@rk2:chan new d2@rd2:chan
  !a2(d2,k2); do ?d2;E2() or ?k2;E2()
)
let KKK() = ?a1(d,k); KKK_E(d,k)
and KKK_E(d:chan,k:chan) =
  do !d;KKK() or !k;KKKst()
and KKKst() = (
  new d3@rd3:chan new k3@rk3:chan
  new d5@rd5:chan new k5@rk5:chan
  do ?a2(k,d); KKK_E(d,k)
  or !a3(d3,k3); (do ?d3;KKKst() or ?k3;KKKst())
  or !a5(d5,k5); (do ?d5;KKKst() or ?k5;KKKst())
)
let KK() = ?a3(d,k); KK_E(d,k)
and KK_E(d:chan,k:chan) = do !d;KK() or !k;KK_P()
and KK_P() =
  do ?a4(k,d);KK_E(d,k) or ?a5(d,k);KK_P_E(d,k)
and KK_P_E(d:chan,k:chan) =
  do !d;KK_P() or !k;KK_PP()
and KK_PP() = (
  new d7@rd7:chan new k7@rk7:chan
  new d9@rd9:chan new k9@rk9:chan
  do ?a6(k,d); KK_P_E(d,k)
  or !a7(d7,k7); (do ?d7;KK_PP() or ?k7;KK_PP())
  or !a9(d9,k9); (do ?d9;KK_PP() or ?k9;KK_PP())
)
let K() = ?a7(d,k); K_E(d,k)
and K_E(d:chan,k:chan) = do !d; K() or !k; K_P()
and K_P() = do ?a8(k,d);K_E(d,k)  or ?a9(d,k);K_P_E(d,k)
and K_P_E(d:chan,k:chan) = do !d;K_P() or !k;K_PP()
and K_PP() = ?a10(k,d); K_P_E(d,k)

let KKPase() = (
  new d4@rd4:chan new k4@rk4:chan
  new d6@rd6:chan new k6@rk6:chan
  do !a4(d4,k4); (do ?d4;KKPase() or ?k4;KKPase())
  or !a6(d6,k6); (do ?d6;KKPase() or ?k6;KKPase())
)
let KPase() = (
  new d8@rd8:chan new k8@rk8:chan
  new d10@rd10:chan new k10@rk10:chan
  do !a8(d8,k8); (do ?d8;KPase() or ?k8;KPase())
  or !a10(d10,k10); (do ?d10;KPase() or ?k10;KPase())
)
run 100 of (KKK() | KK() | K())
run ( E2() | KKPase() | KPase() | E1())
```

Mapk Cascade

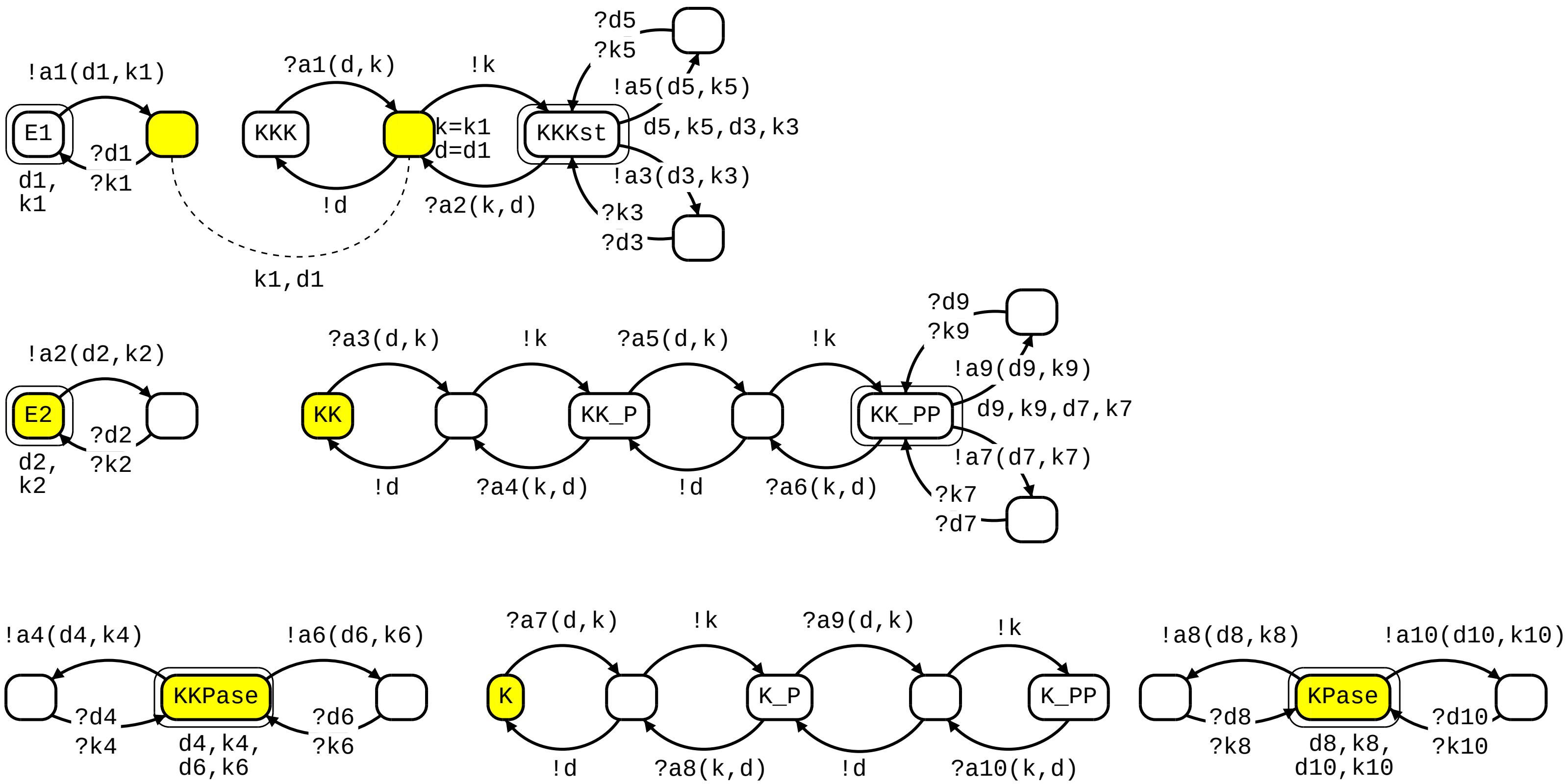


Mapk Cascade



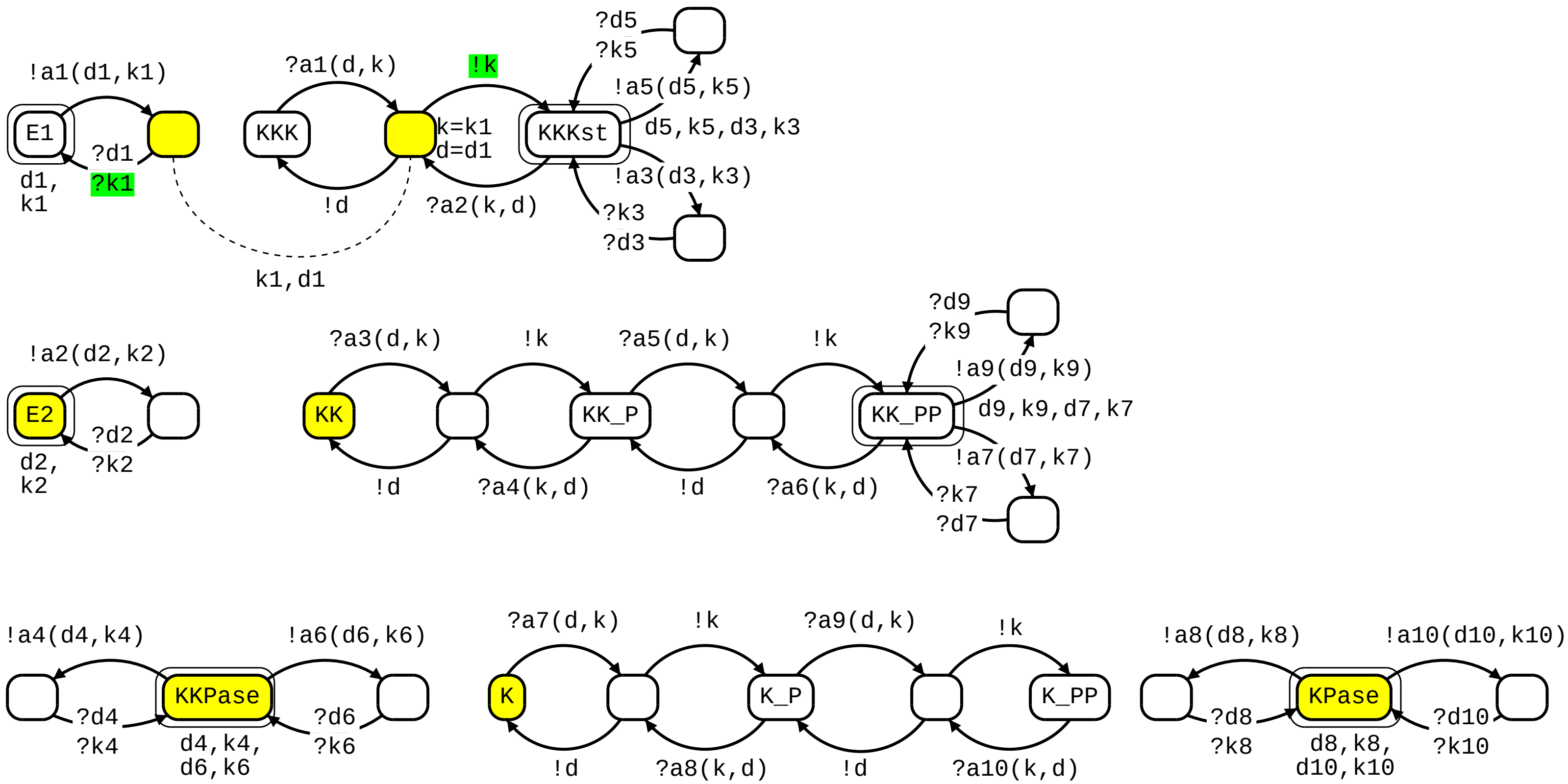
Enzyme E_1 can bind to substrate KKK using channel a_1

Mapk Cascade



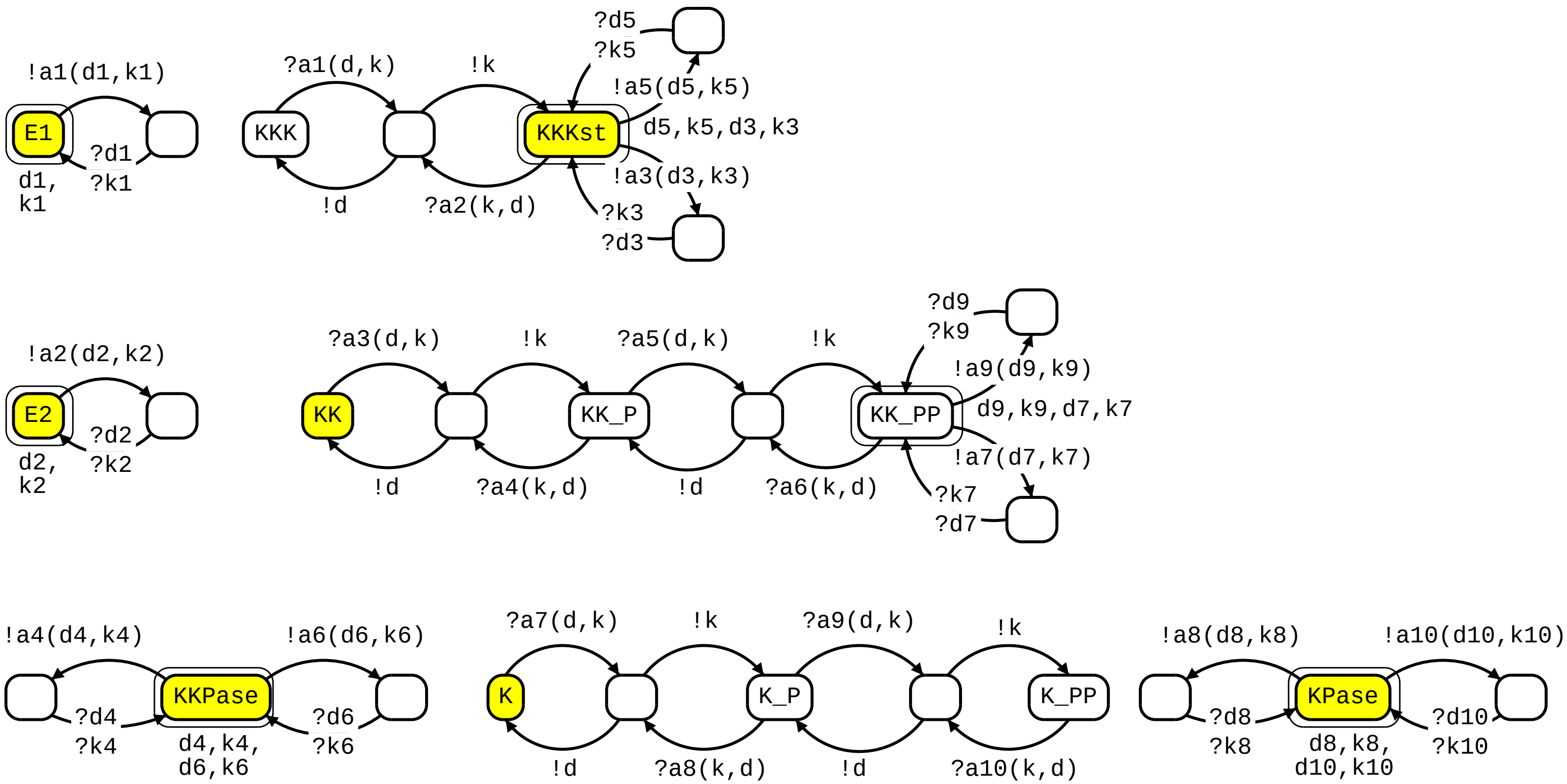
E_1 is bound to KKK by private channels d_1 and k_1

Mapk Cascade



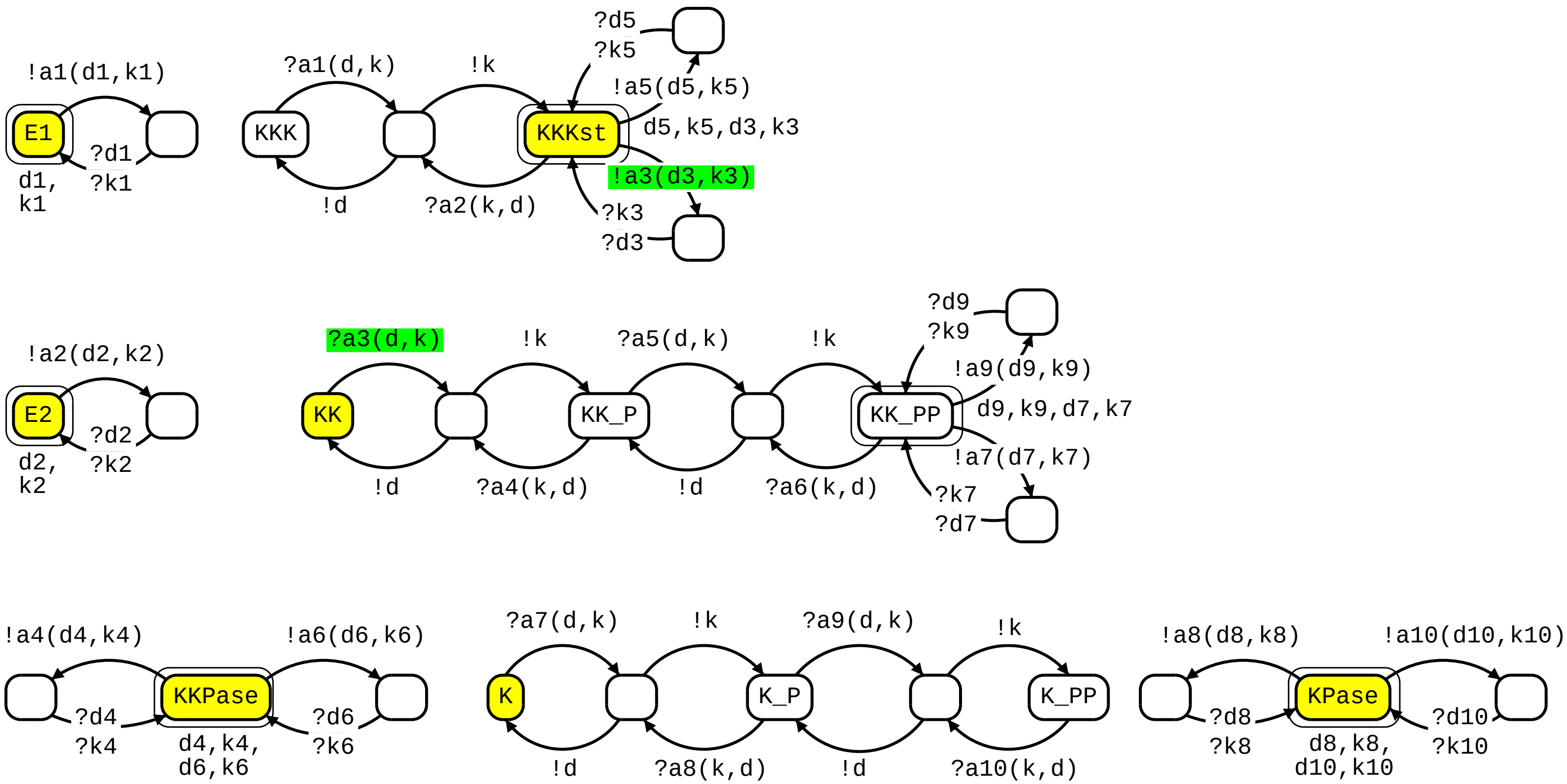
E_1 can react with KKK using channel k_1

Mapk Cascade



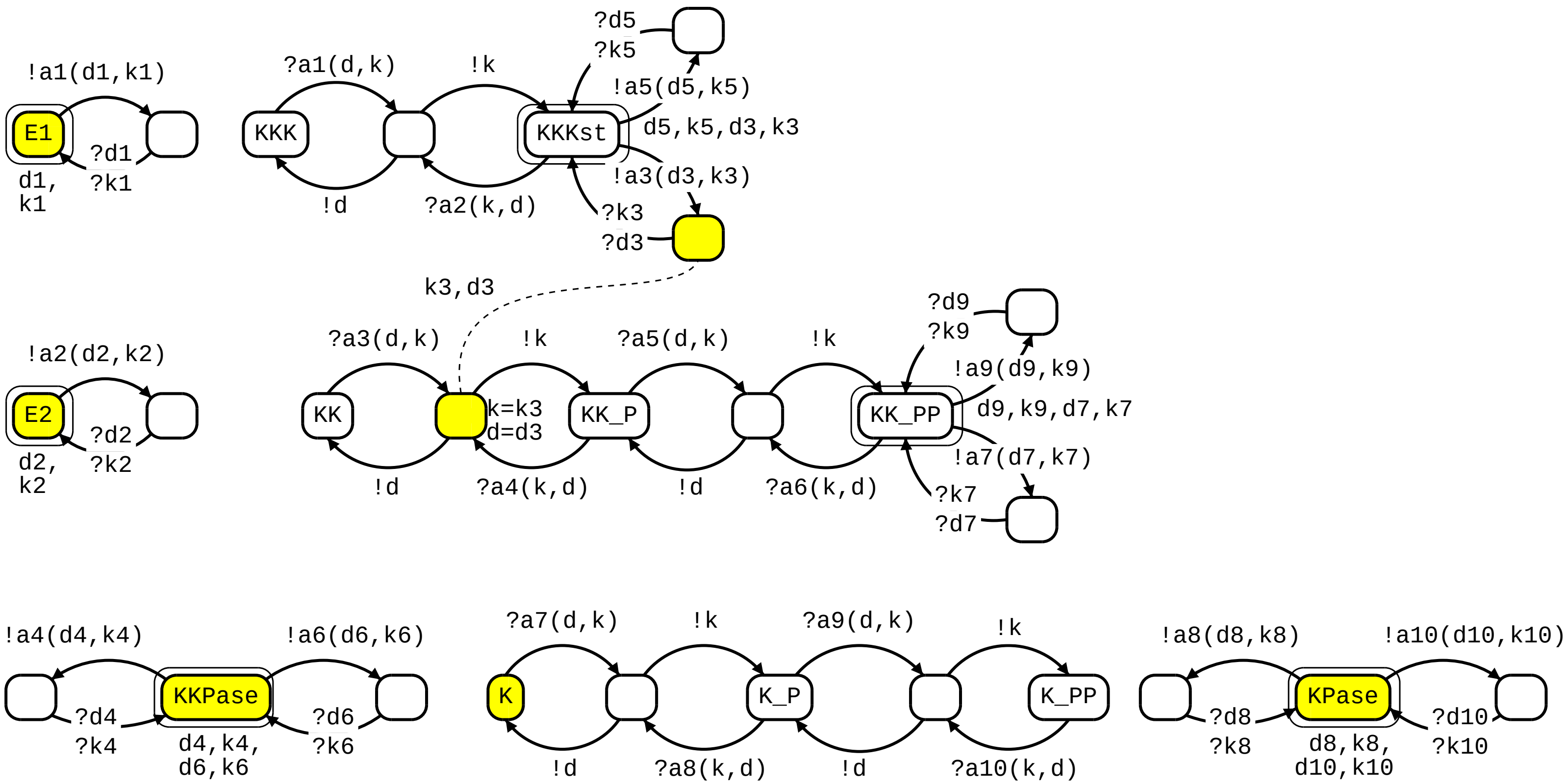
KKK is transformed to KKK*

Mapk Cascade



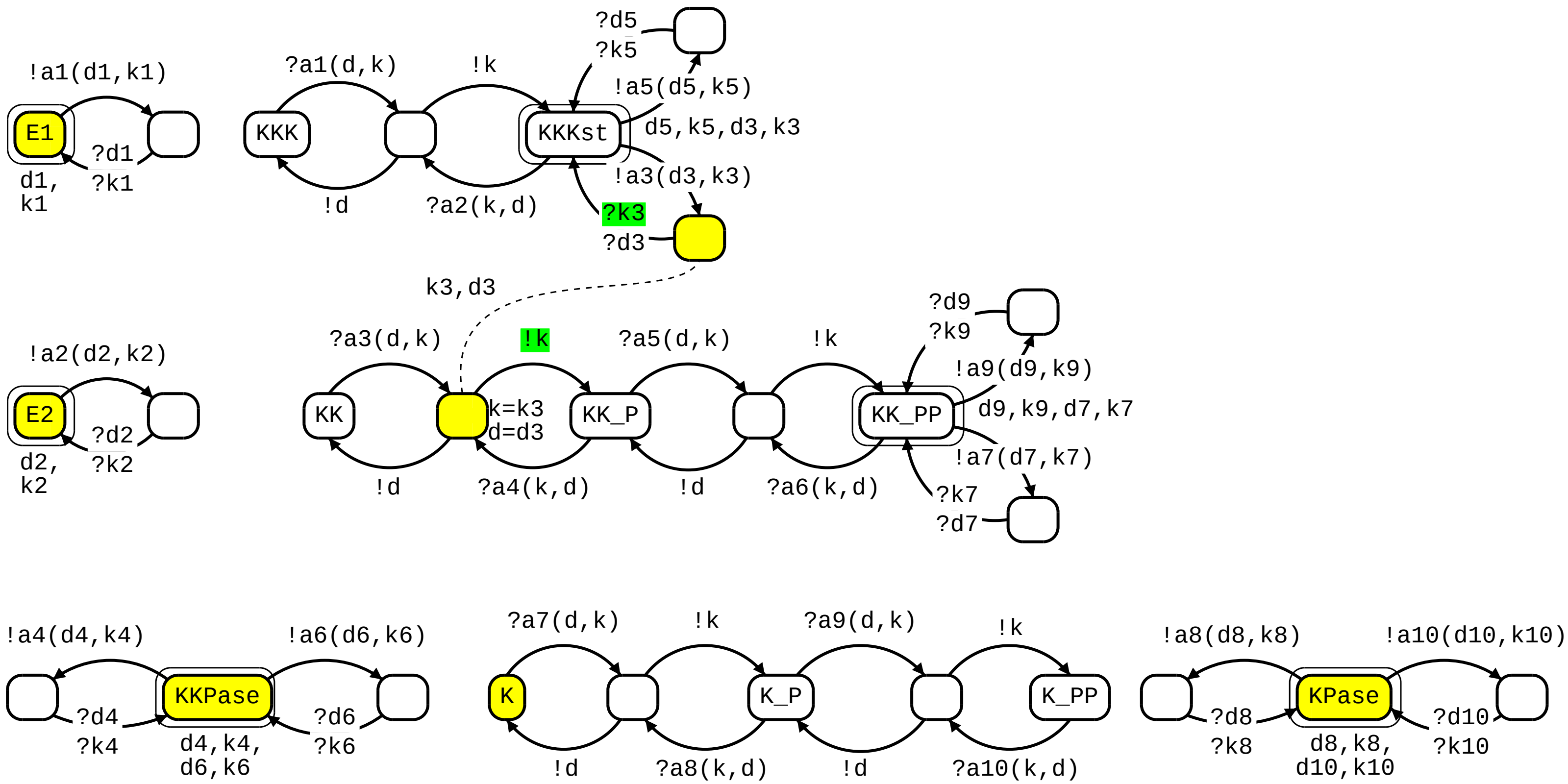
KKK* can bind with KK using channel a_3

Mapk Cascade



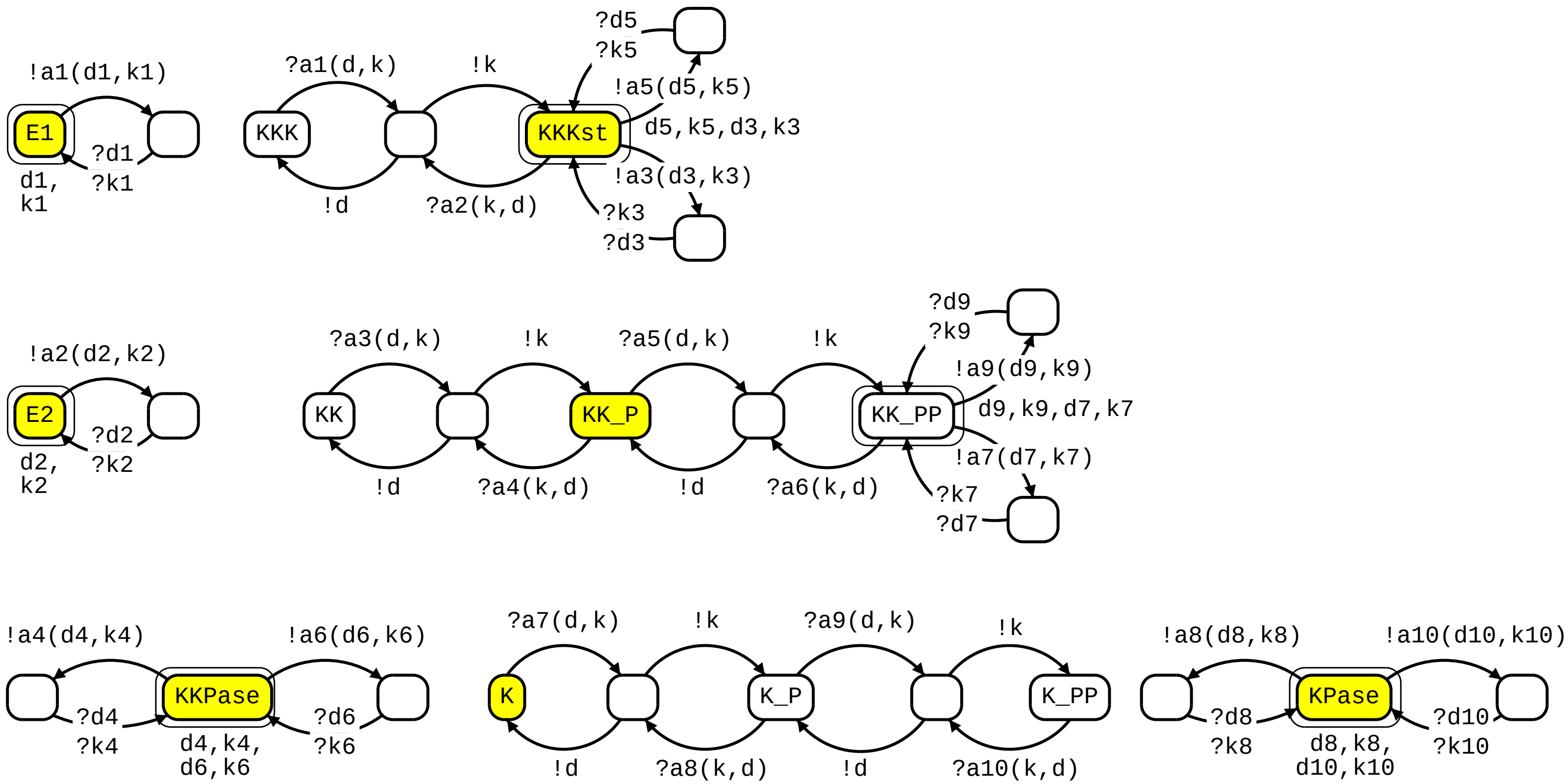
KKK* is bound to KK by private channels d_3 and k_3

Mapk Cascade



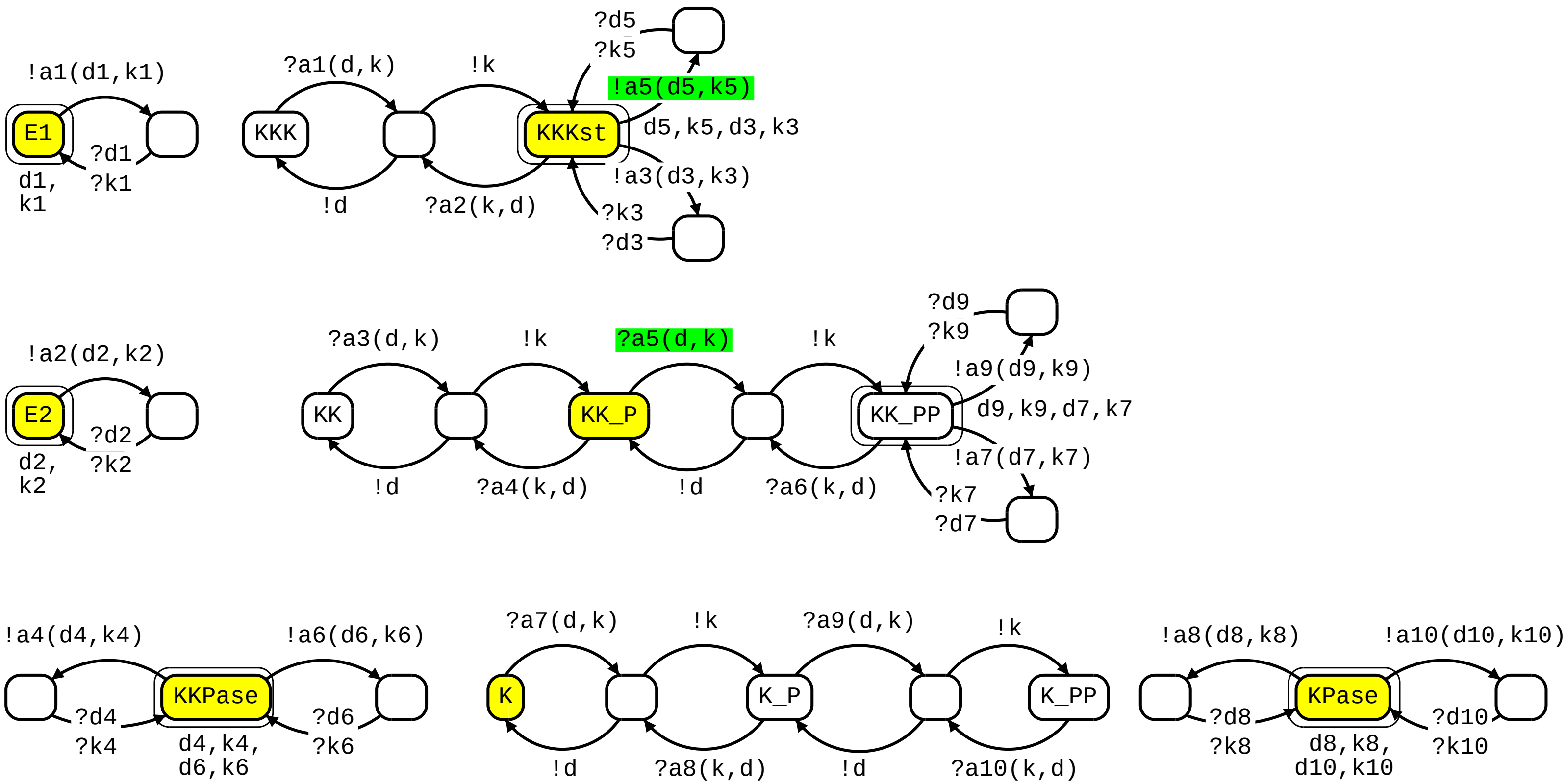
KKK* can react with KK using channel k_3

Mapk Cascade



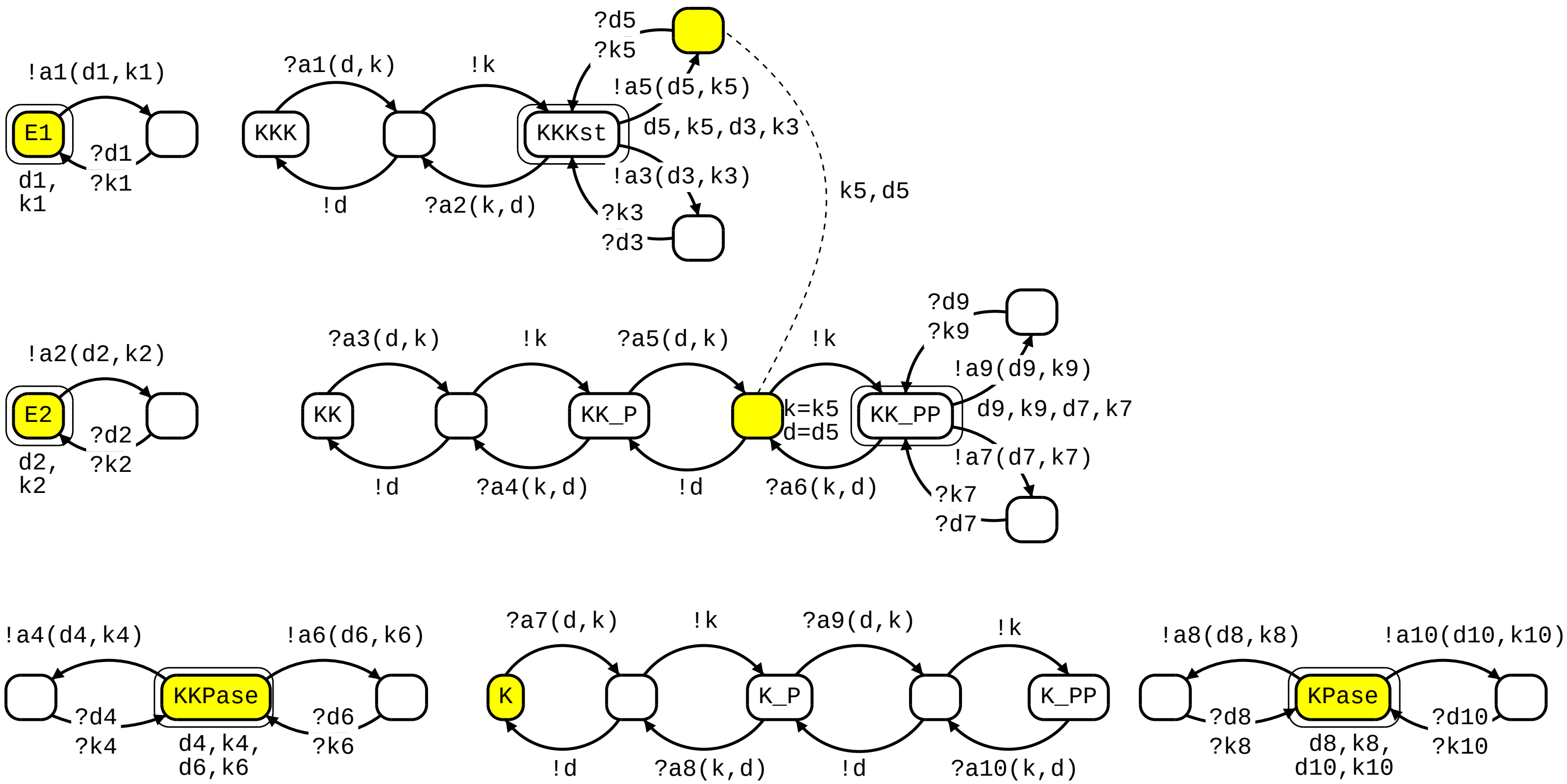
KK is transformed to KK-P

Mapk Cascade



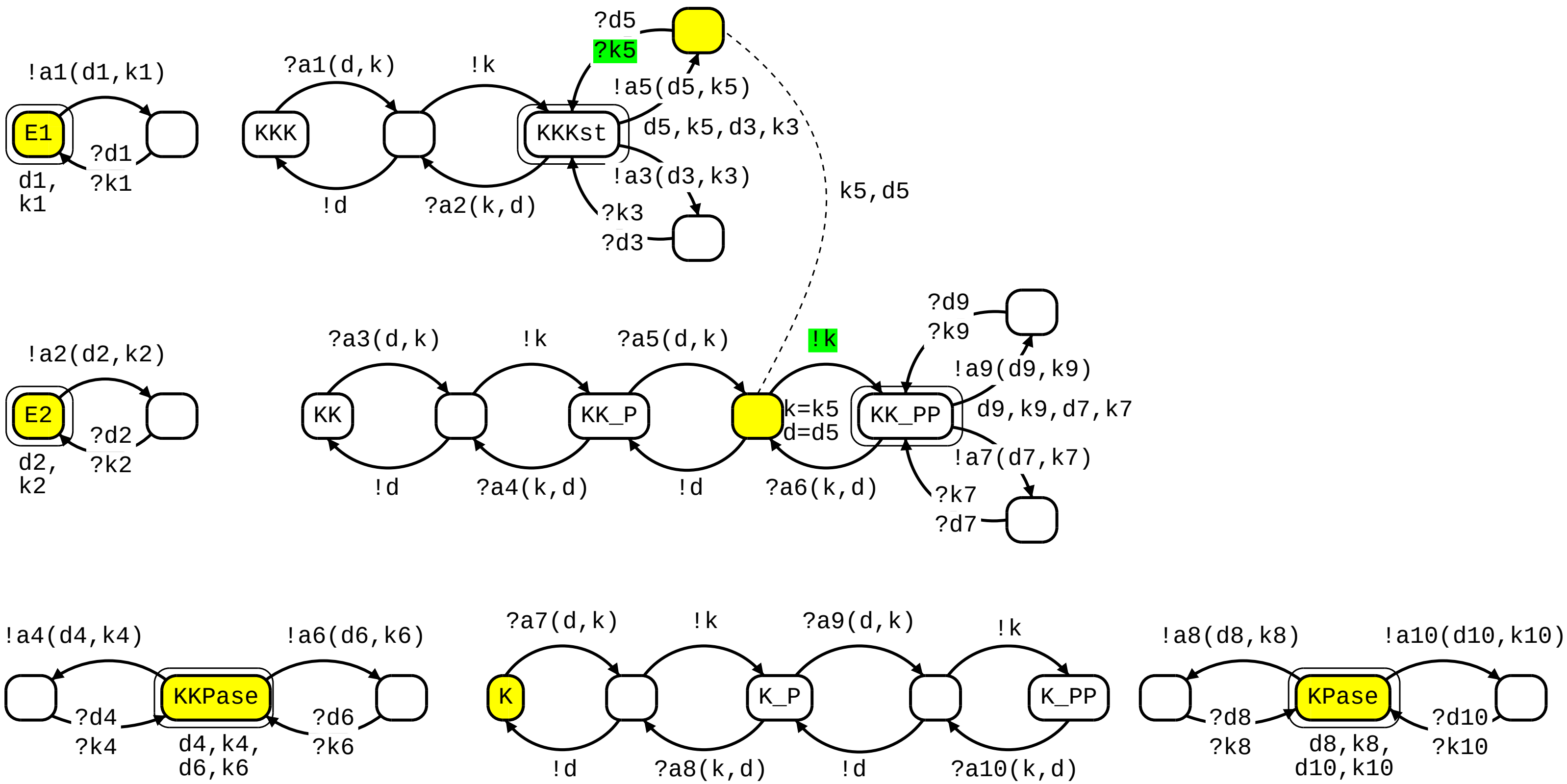
KKK* can bind to KK-P using channel a_5

Mapk Cascade



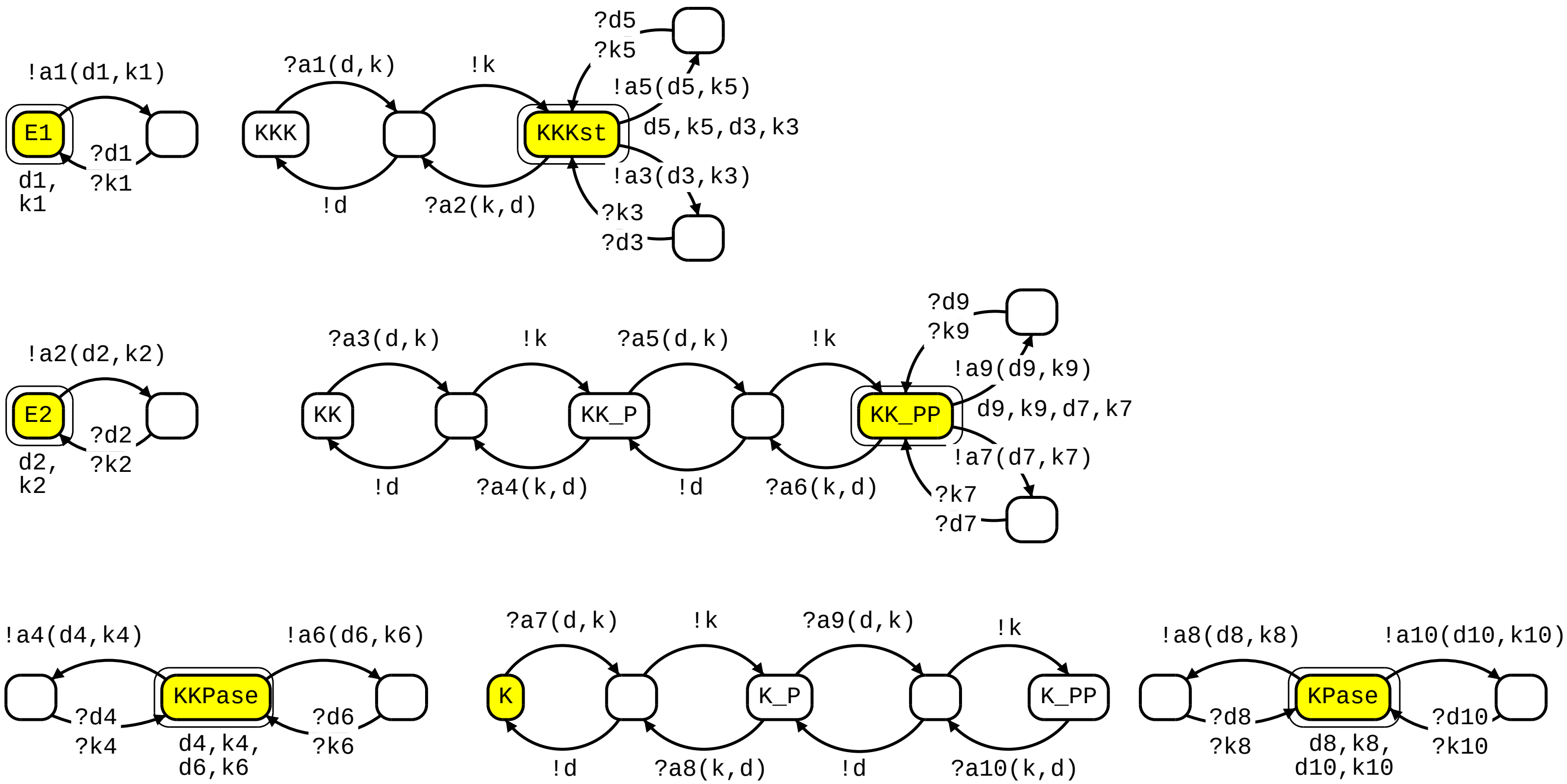
KKK* is bound to KK-P by channels d_5 and k_5

Mapk Cascade



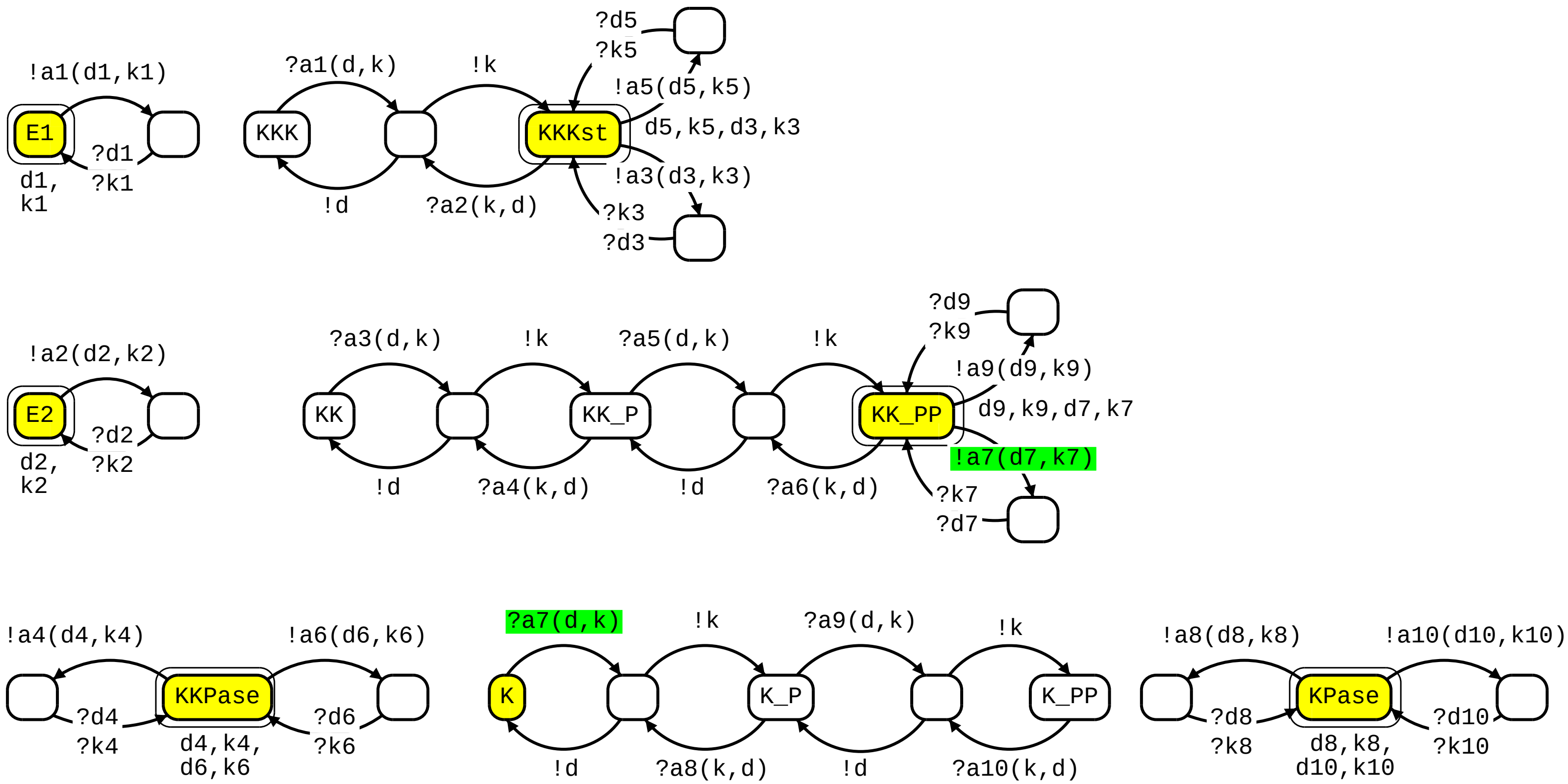
KKK* can react with KK-P using channel k_5

Mapk Cascade



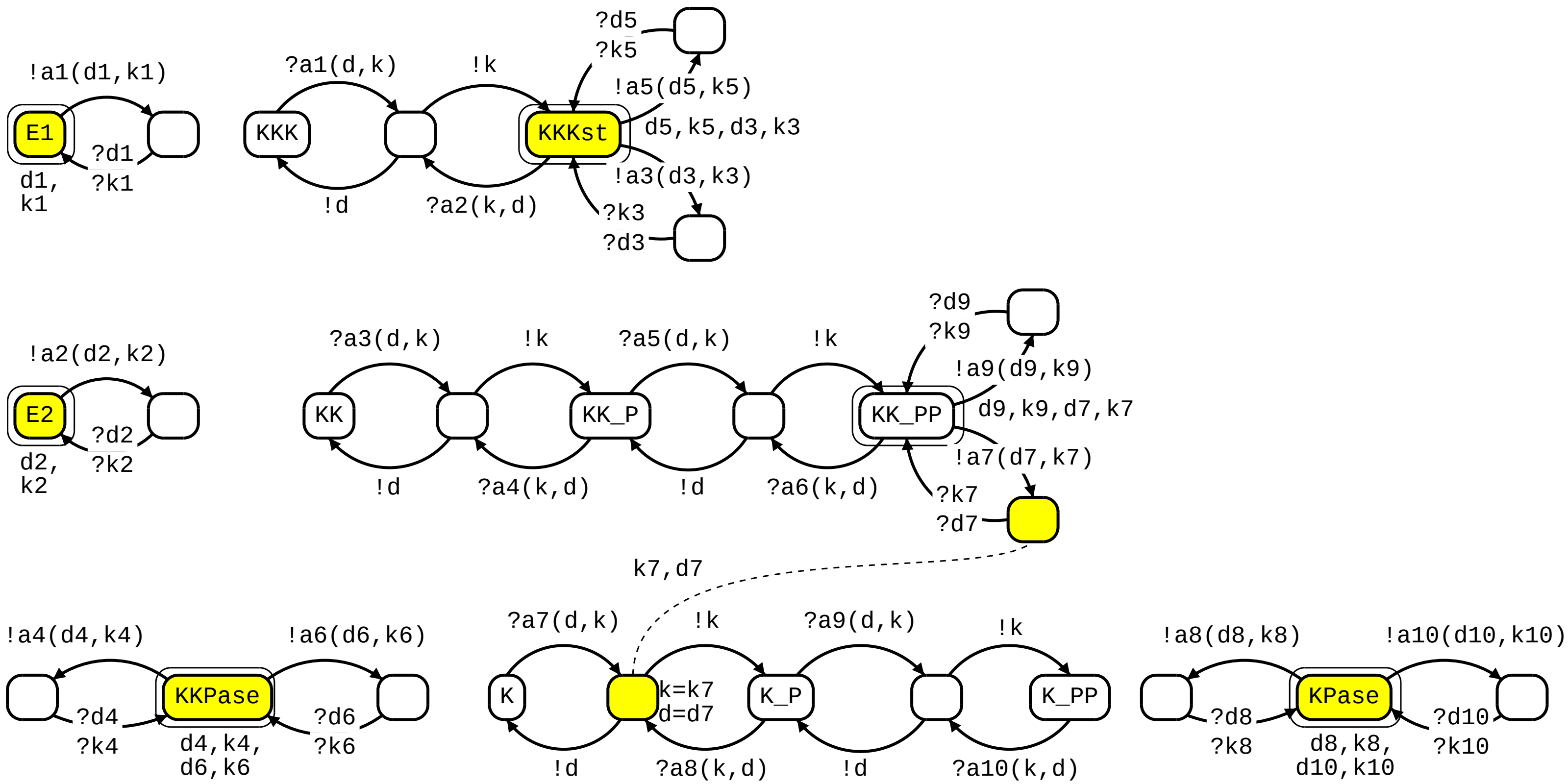
KK-P is transformed to KK-PP

Mapk Cascade



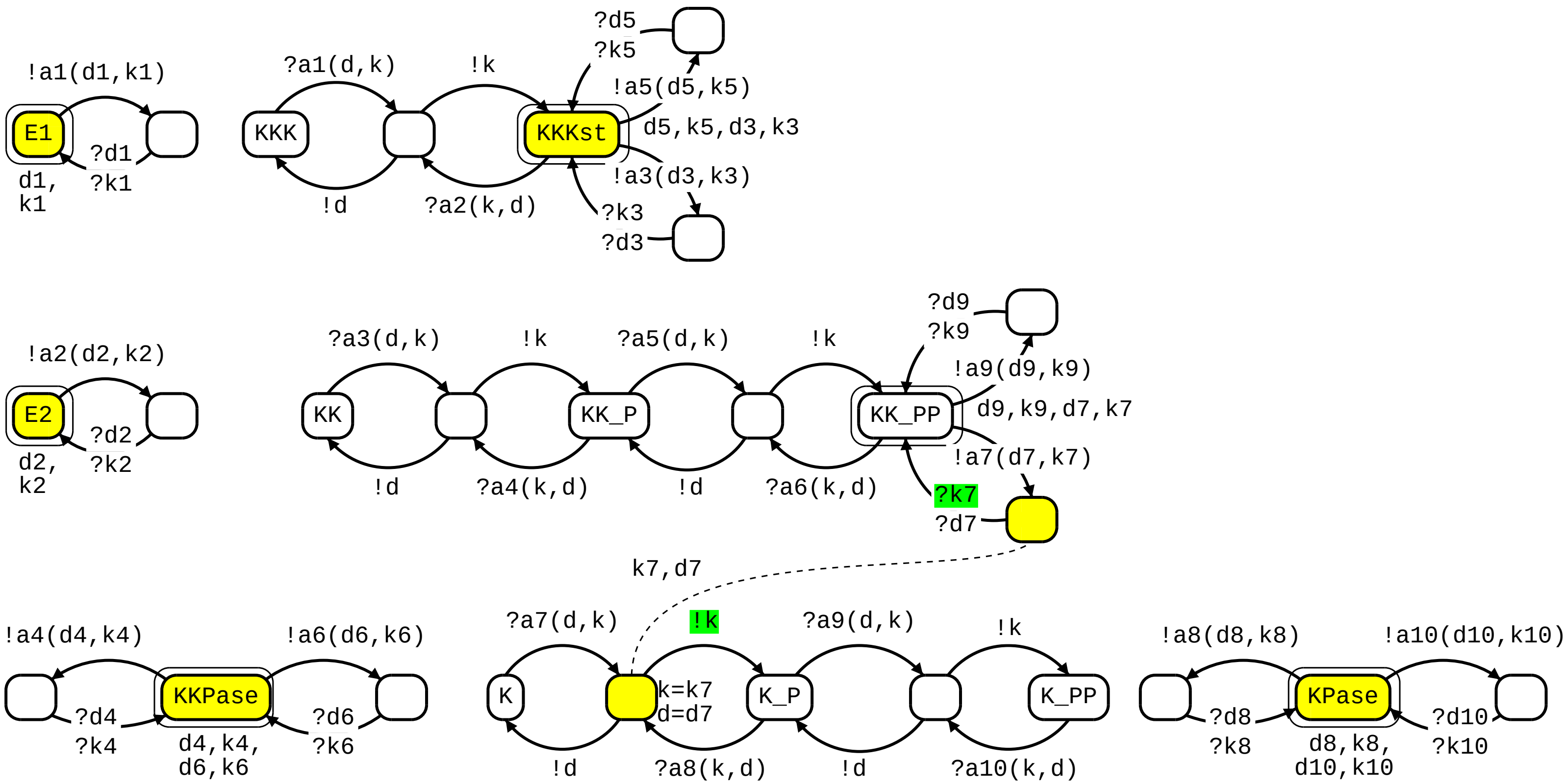
KK-PP can bind to K using channel a_7

Mapk Cascade



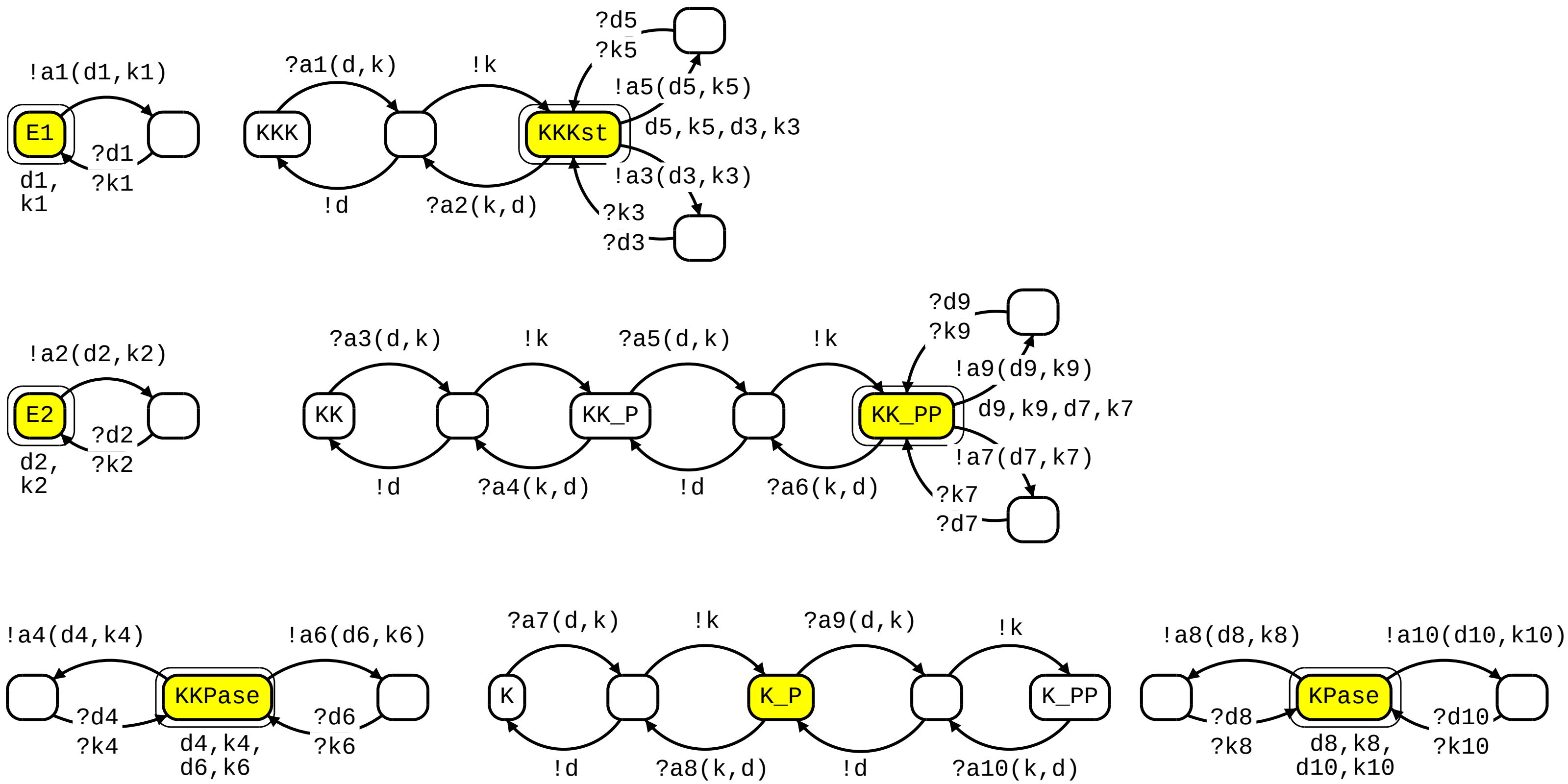
KK-PP is bound to K by channels d_7 and k_7

Mapk Cascade



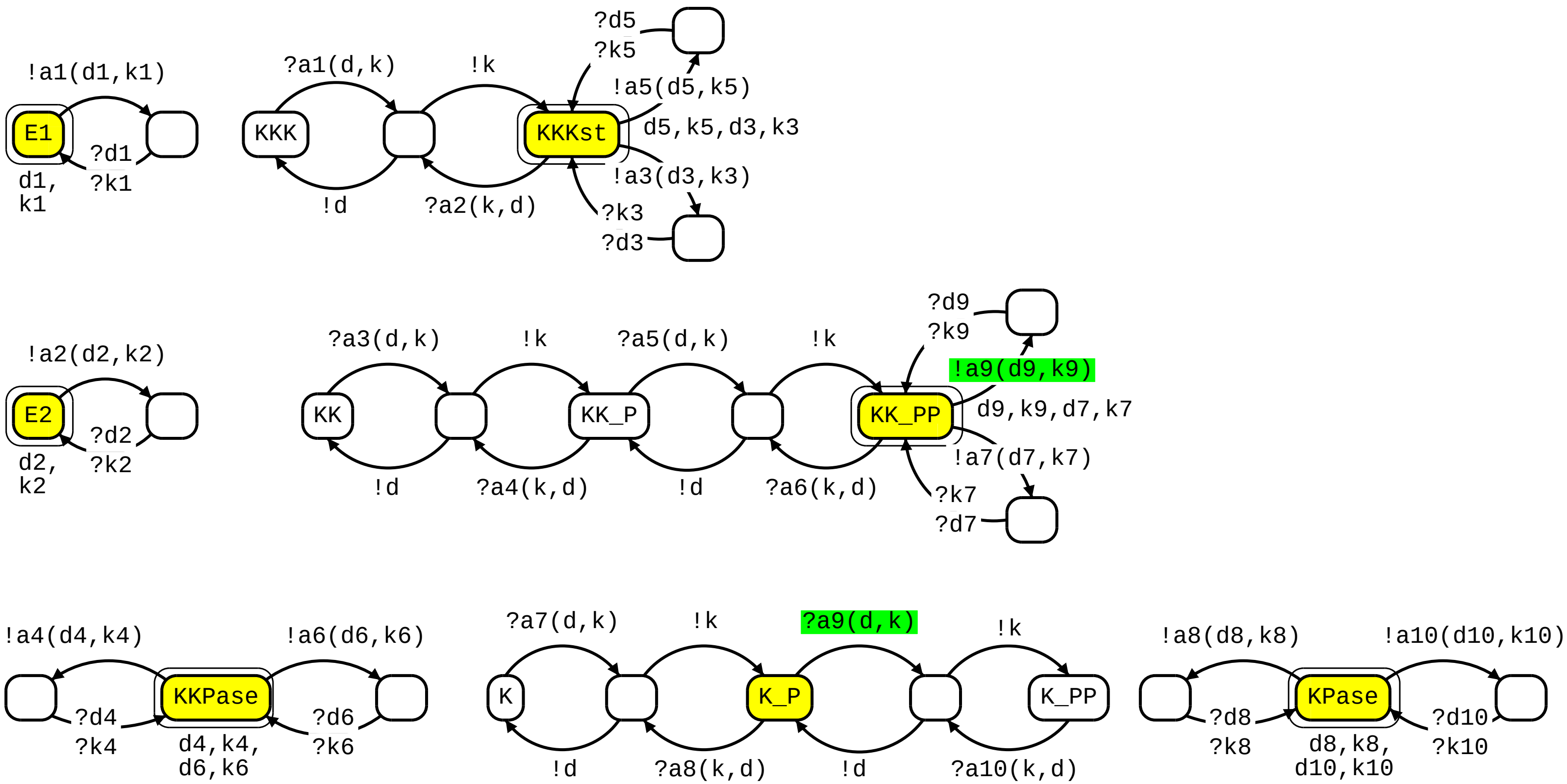
KK-PP can react with K using channel k_7

Mapk Cascade



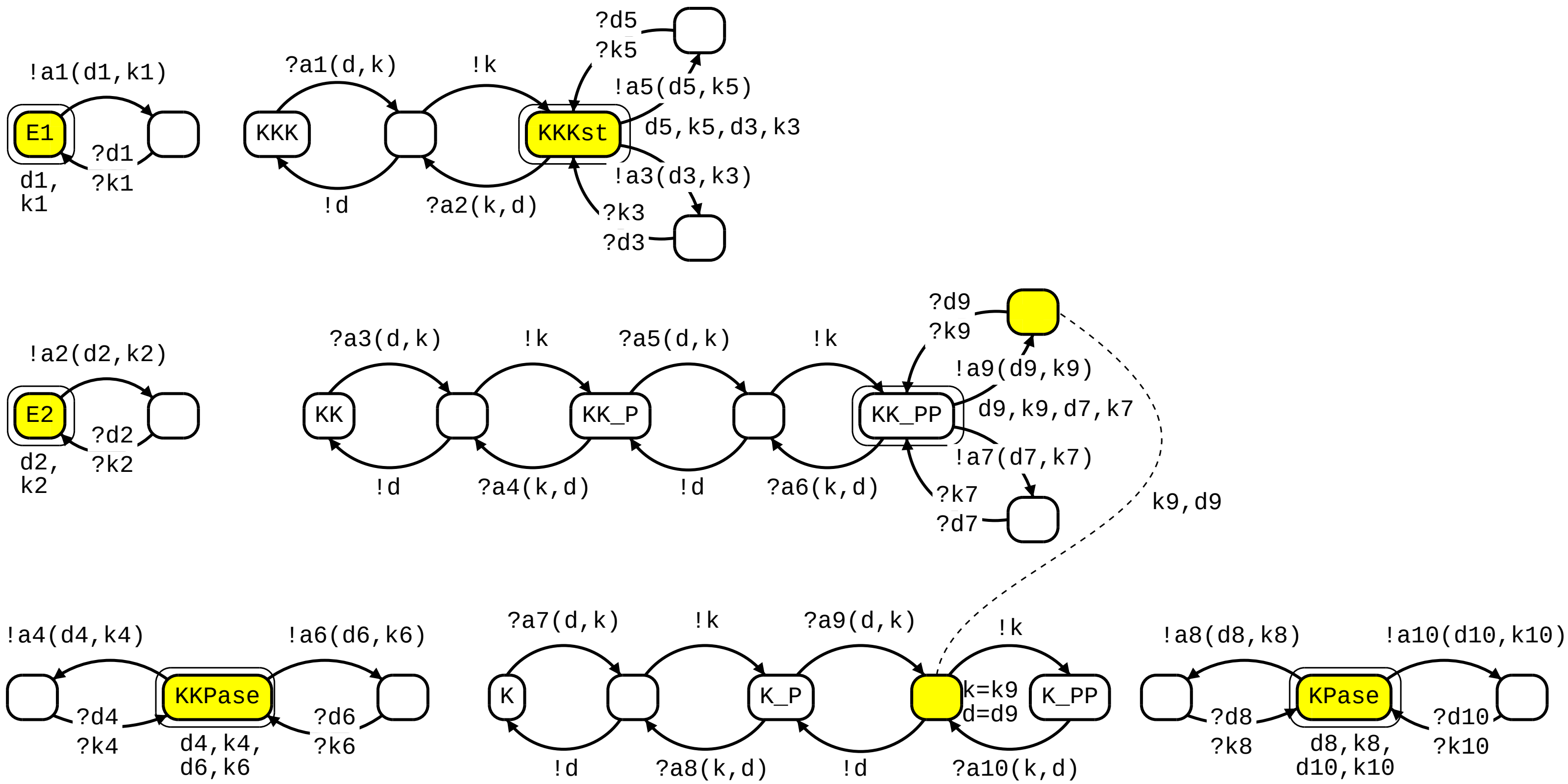
K is transformed to KK-P

Mapk Cascade



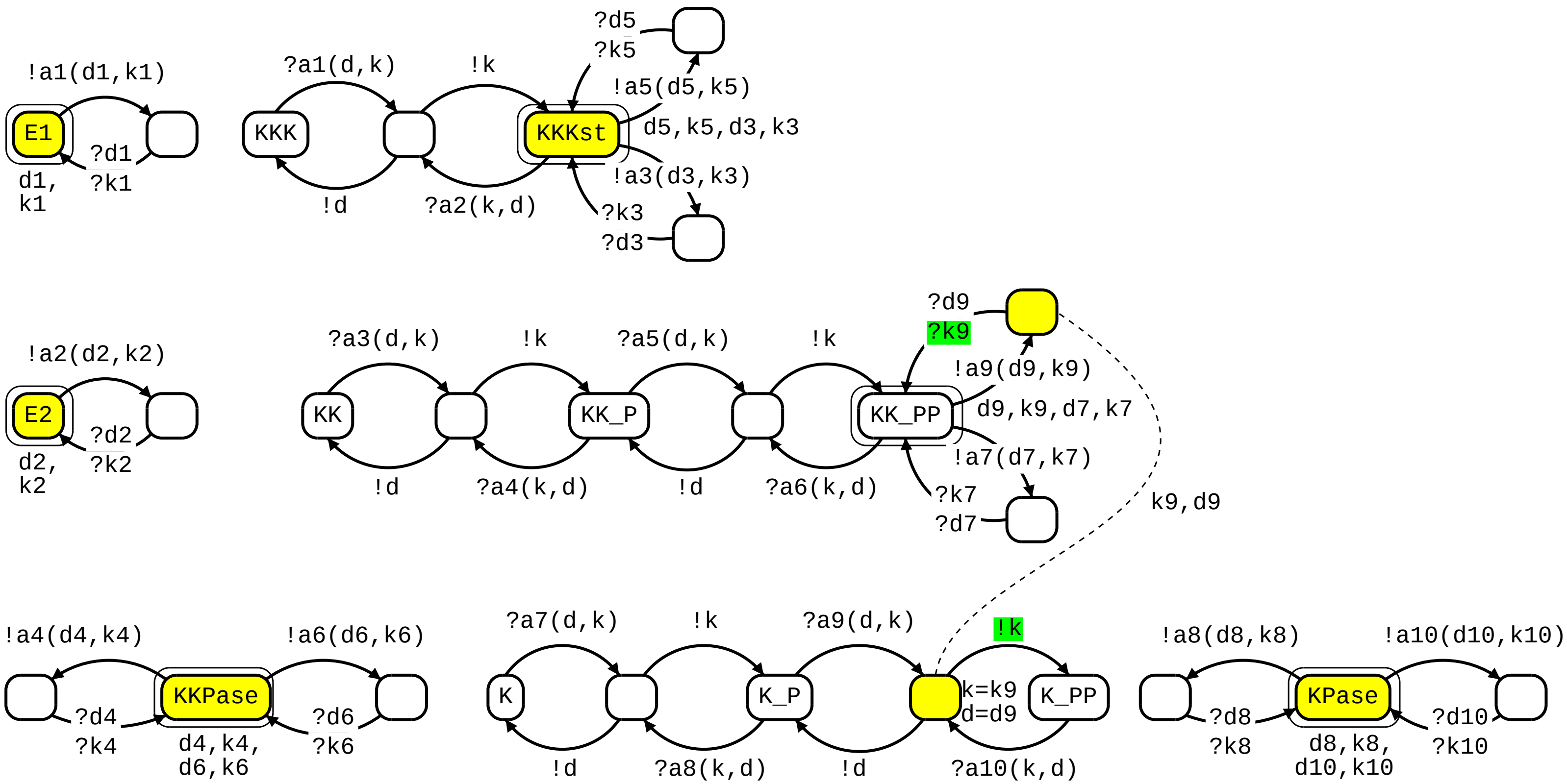
KK-PP can bind to KK-P using channel a_9

Mapk Cascade



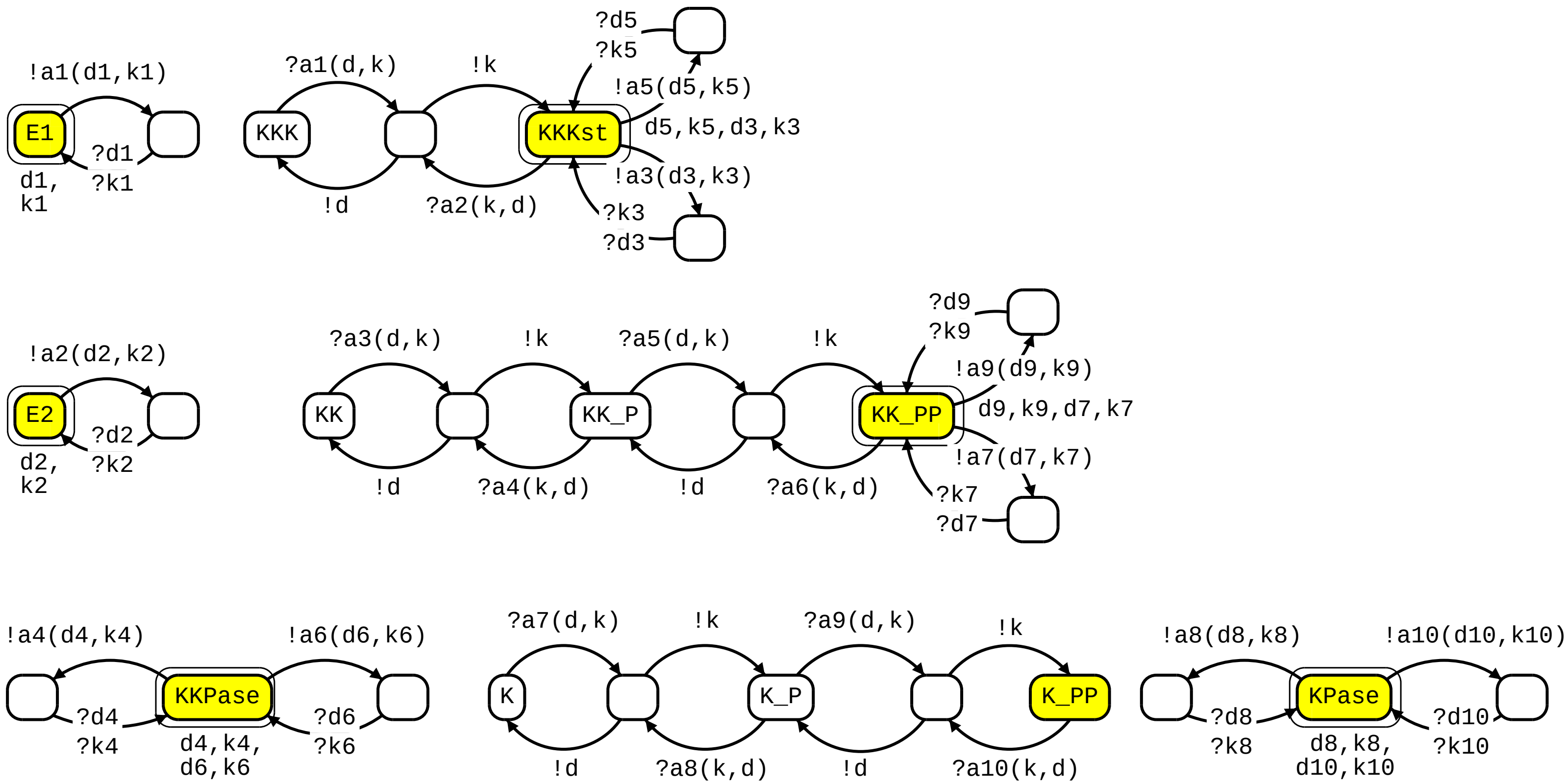
KK-PP is bound to KK-P by channels d_9 and k_9

Mapk Cascade



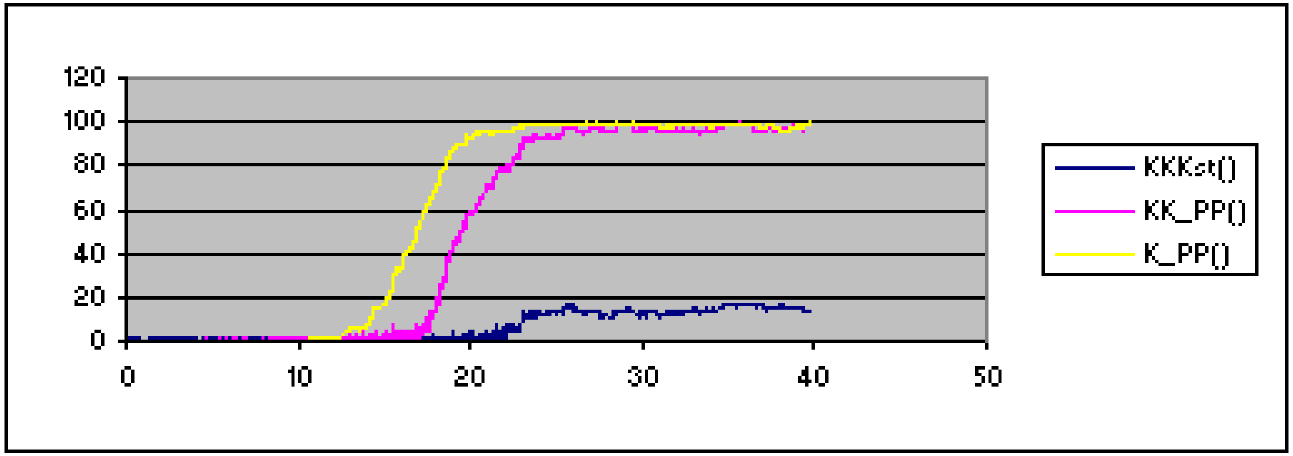
KK-PP can react with K-P using channel k_9

Mapk Cascade



K-P is transformed to K-PP, completing the cascade

Mapk Cascade: Results



Open Graph Syntax (DOT)

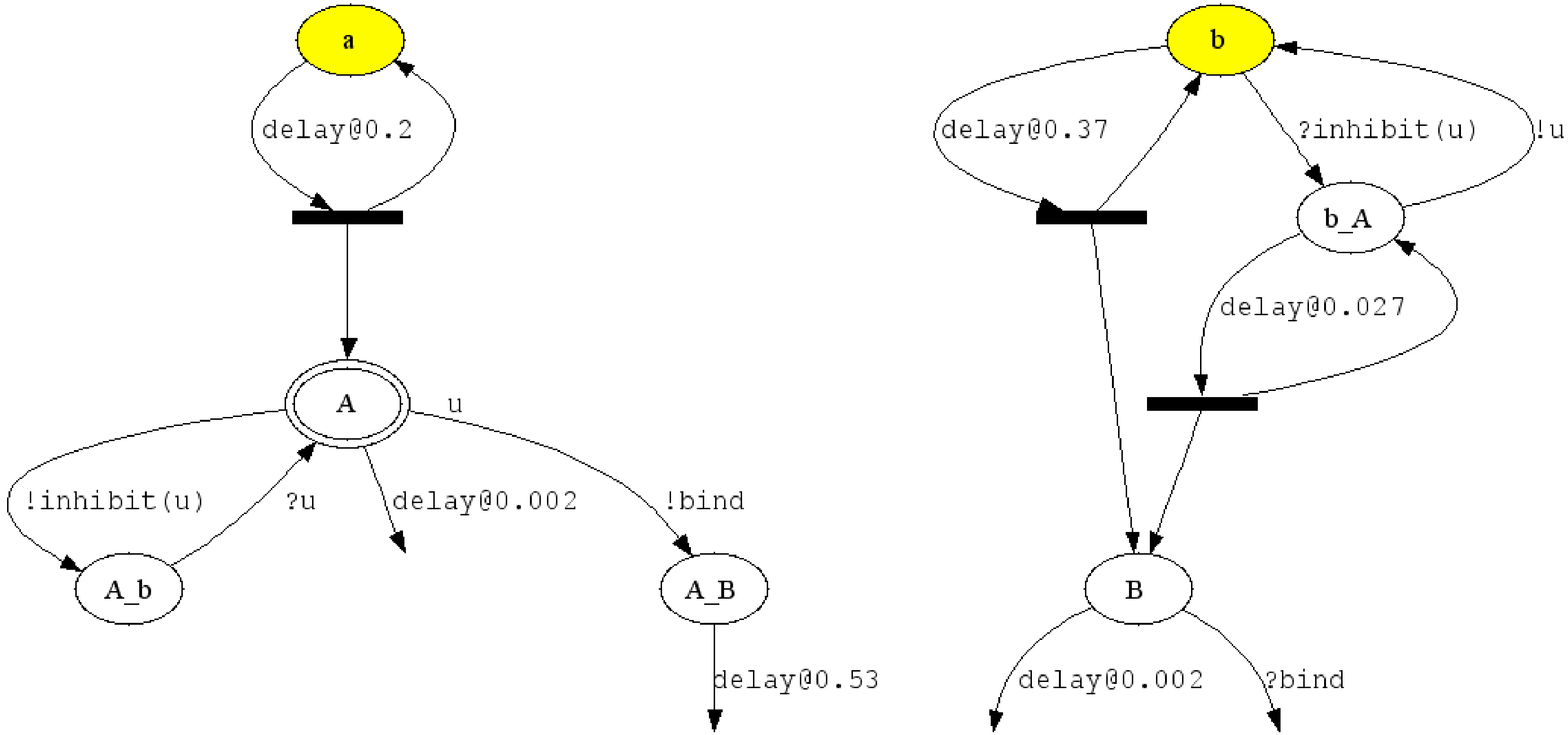
$$
\begin{array}{rcl}
G ::= & I[label] & \text{Labelled node with id } I \\
\mid & I \xrightarrow{label}_{label} J & \text{Labelled edge from node } I \text{ to node } J \\
\mid & G; G' & \text{Sequence of graph declarations}
\end{array}
$$

Graph Generation

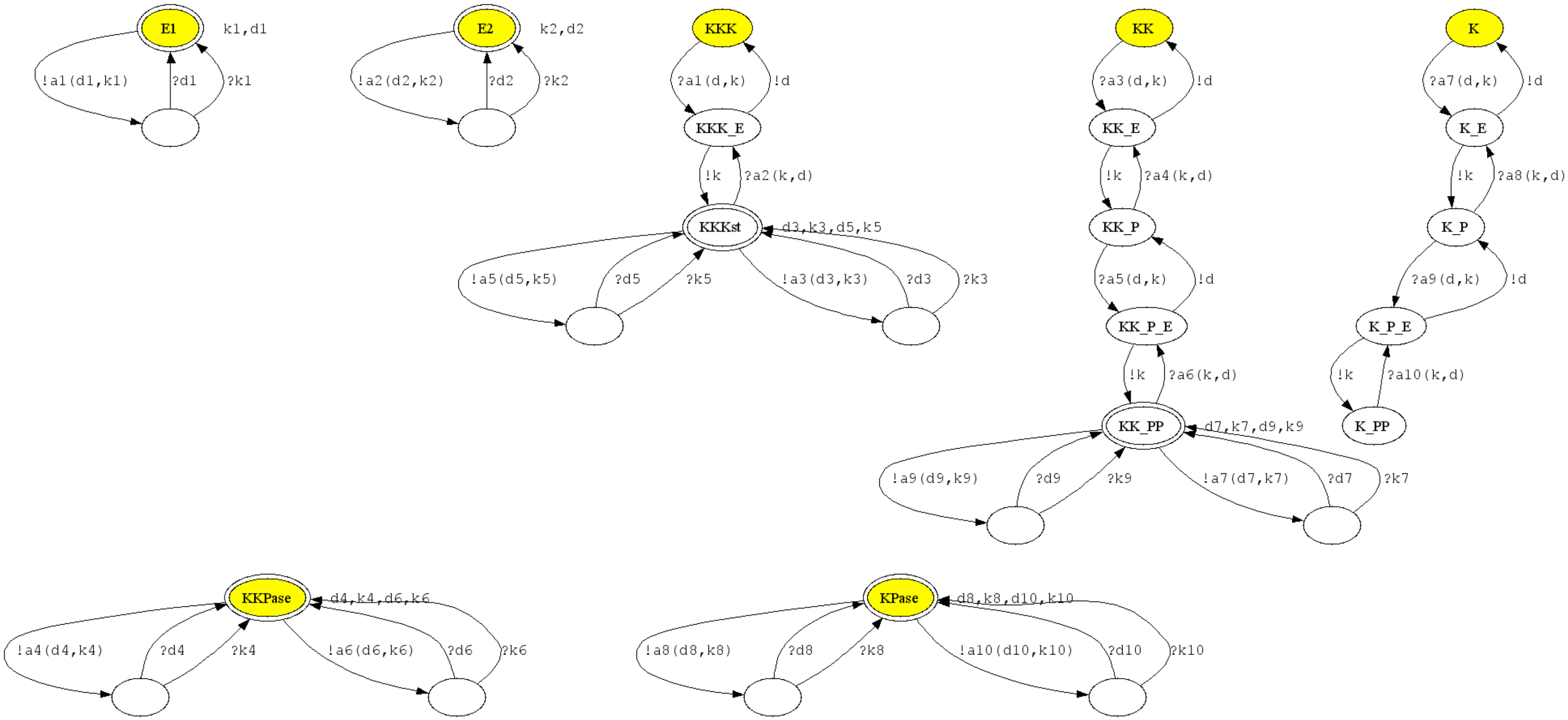
Proposition 2. $\forall \Gamma . \Gamma \in \mathrm{GSPi}_\Gamma \Rightarrow (|\Gamma|)_I \in \mathrm{DOT}$

$$
\begin{array}{rcrcl}
 & & (|\emptyset|)_I & \triangleq & \emptyset \\
 & & (|X(\vec{m}) \triangleq C, \Gamma|)_I & \triangleq & (|C|)_X ; (|\Gamma|)_I \\
 & & (|\nu x_1 \ldots \nu x_N \, C|)_I & \triangleq & I[x_1, \ldots, x_N]; (|C|)_I \\
X(\vec{m}) \triangleq C & \Rightarrow & (|X(\vec{n})|)_I & \triangleq & I \longrightarrow_{\{\vec{n}/\vec{m}\}} X \\
X(\vec{m}) \triangleq C & \Rightarrow & (|X(\vec{n}) \mid \Pi|)_I & \triangleq & I \longrightarrow_{\{\vec{n}/\vec{m}\}} X ; (|\Pi|)_I \\
 & & (|\mathbf{0}|)_I & \triangleq & \emptyset \\
X(\vec{m}) \triangleq C & \Rightarrow & (|\pi . X(\vec{n}) + \Sigma|)_I & \triangleq & I \xrightarrow{\pi}_{\{\vec{n}/\vec{m}\}} X ; (|\Sigma|)_I
\end{array}
$$

Generated Graphs: Evolved Network



Generated Graph: Mapk Cascade



Related Work

➢ Statecharts [Harel, 1987] highlighted the need for a scalable, self-contained graphical representation of concurrent systems.

➢ Synchronous variant to Statecharts allows concurrent processes to synchronise on shared labels [Andre, 1995].

➢ Foundational graphical representations for pi-calculus use elaborate graph rewriting rules [Milner, 1994].

➢ More recently, HDA [Montanari and Pistore, 2005] describes an automata-based representation for the pi-calculus.

➢ Preliminary informal ideas on a graphical representation for the stochastic pi-calculus in [Phillips and Cardelli, 2004].

Conclusion

- Presented a graphical representation for the stochastic pi-calculus.
- Used to model a Mapk signalling cascade and an evolved gene network.
- Highlights the existence of cycles, which are key to many biological systems.
- Able to animate interactions between biological system components.
- Able to clarify the overall system function and to debug changes in the system.

Future Work

- ➢ Observation: within a collection of mutually recursive definitions, applied arguments often the same as the formal parameters.
- ➢ Investigate additional design patterns to improve modelling and visualisation techniques.
- ➢ Use graph generation tool to implement a graphical debugger for SPiM.
- ➢ Develop a tool for drawing graphical models, which automatically generates SPiM code.
- ➢ Make modelling and simulation of biological systems more accessible to non computer scientists.

References

[Andre, 1995] Andre, C. (1995). Synccharts: A visual representation of reactive behaviors. research report tr95-52, University of Nice, Sophia Antipolis.

[Francois and Hakim, 2004] Francois, P. and Hakim, V. (2004). Design of genetic networks with specified functions by evolution in silico. volume 101, pages 580–585.

[Harel, 1987] Harel, D. (1987). Statecharts : A Visual Formalism for Complex Systems. *Sci. Comput. Prog.*, 8:231–274.

[Huang and Ferrel, 1996] Huang, C.-Y. F. and Ferrel, J. E. (1996). Ultrasensitivity of the mitogen-activated protein cascade. volume 93, pages 10078–10083.

[Lecca and Priami, 2003] Lecca, P. and Priami, C. (2003). Cell cycle control in eukaryotes: a biospi model. In *BioConcur'03*. ENTCS.

[Milner, 1994] Milner, R. (1994). Pi-nets: A graphical form of π-calculus. In *ESOP'94*, pages 26–42.

[Montanari and Pistore, 2005] Montanari, U. and Pistore, M. (2005). History-dependent automata: An introduction. volume 3465.

[Phillips and Cardelli, 2004] Phillips, A. and Cardelli, L. (2004). A correct abstract machine for the stochastic pi-calculus. In *Bioconcur'04*. ENTCS.

[Priami et al., 2001] Priami, C., Regev, A., Shapiro, E., and Silverman, W. (2001). Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*. In press.

[Regev et al., 2001] Regev, A., Silverman, W., and Shapiro, E. (2001). Representation and simulation of biochemical processes using the pi- calculus process algebra. In Altman, R. B., Dunker, A. K., Hunter, L., and Klein, T. E., editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470, Singapore. World Scientific Press.