Exploring Variation in Biochemical Pathways with the Continuous π-Calculus

Ian Stark and Marek Kwiatkowski

Laboratory for Foundations of Computer Science
School of Informatics
The University of Edinburgh

Evolutionary Ecology
Institute of Integrative Biology
Department of Environmental Sciences
EAWAG & ETH Zürich

Friday 16 December 2010

http://homepages.ed.ac.uk/stark http://mareklab.org

Summary

The *continuous* π*-calculus* ($c\pi$) is a process algebra for modelling behaviour and variation in molecular systems.

It has a structured operational semantics that captures system behaviour as trajectories through a continuous process space, by generating familiar differential-equation models.

We have existing biochemical systems expressed in $c\pi$; in particular, a standard setting of the MAPK cascade.

By systematically exploring neighbourhoods of this basic pathway model, we have been able to identify the robustness and evolvability of its individual components.

Contents

- Development and evolution
- The continuous π-calculus
- Compilation and execution
- Variation operators
- Experiments on MAPK cascade

Development and Evolution

Development is the process by which genetic information (genotype) is translated to a functional biological object (phenotype).

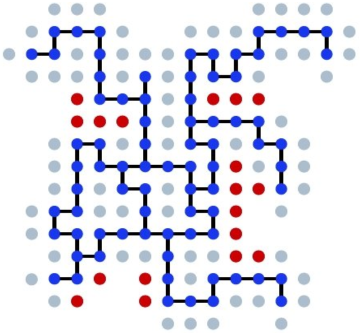
In most settings of interest, development is notoriously complex. For example, an embryo becoming an organism or a peptide chain folding into a protein.

Evolutionary developmental biology (evo-devo) is concerned with evolution-related properties of development, such as *evolvability*, *robustness*, *canalisation* and *plasticity*.

Mathematical abstractions and simple instances of development help to illuminate generic features of this process.

Neutral Spaces and Neighbours

The neutral space of a phenotype is the collection of all genotypes giving rise to that phenotype.



- ✓ robustness
- ✓ evolvability
- ✓ neutral evolution

- ? recombination
- ? horizontal gene transfer

- ✗ phenotype plasticity
- ✗ variable development

A. Wagner Robustness and Evolvability in Living Systems Princeton University Press, 2005

The Continuous π-Calculus

The *continuous π-calculus* ($\mathrm{c}\pi$) is a name-passing process algebra that generates system behaviours as trajectories over time through a real-valued vector space.

The intended application is modelling behaviour and variation in biomolecular systems, where the vector space is a *phase space* of chemical concentrations.

Marek Kwiatkowski and Ian Stark.
The Continuous π-Calculus: A Process Algebra for Biochemical Modelling. In
Computational Methods in Systems Biology: Proc. CMSB 2008
Lecture Notes in Computer Science 5307, pages 103–122. Springer 2008

Marek Kwiatkowski.
A Formal Computational Framework for the Study of Molecular Evolution
PhD Dissertation, University of Edinburgh, December 2010.

The Continuous π-Calculus

Formality: Unambiguous description

Parsimony: Few primitives

Compositionality: The behaviour of a whole arises entirely from the behaviour of its parts.

Abstraction: System description distinct from system dynamics

Intermediation: Potentially many analysis techniques for a single description

- *Continuous* rather than discrete amounts of agents
- *Flexible* interaction structure of names

Basics of $c\pi$

Continuous π has two levels of system description:

- Species
  - Individual molecules (proteins)
  - Transition system semantics
- Processes
  - Bulk population (concentration)
  - Differential equations

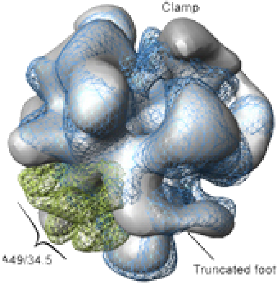
Process space arises as a real-valued vector space over species, with each point the state of a system and behaviours as trajectories through that.

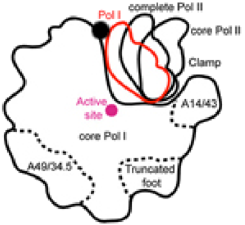
Names in $c\pi$

As in standard π-calculus, *names* indicate a potential for interaction: for example, the docking sites on an enzyme where other molecules may attach.

These sites may interact with many different other sites, to different degrees.

This variation is captured by an *affinity network*: a graph setting out the interaction potential between different names.



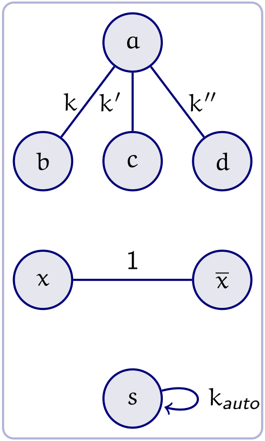


Names in $c\pi$

As in standard π-calculus, *names* indicate a potential for interaction: for example, the docking sites on an enzyme where other molecules may attach.

These sites may interact with many different other sites, to different degrees.

This variation is captured by an *affinity network*: a graph setting out the interaction potential between different names.

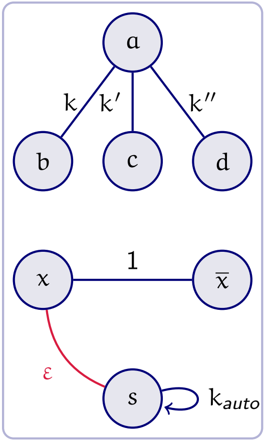


Names in $c\pi$

As in standard π-calculus, *names* indicate a potential for interaction: for example, the docking sites on an enzyme where other molecules may attach.

These sites may interact with many different other sites, to different degrees.

This variation is captured by an *affinity network*: a graph setting out the interaction potential between different names.



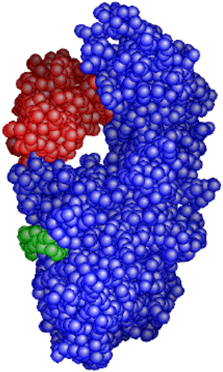
Restriction in $c\pi$

Name restriction $\nu x(A \mid B)$ captures molecular *complexes*, with local name x mediating further internal modification, or decomplexation.

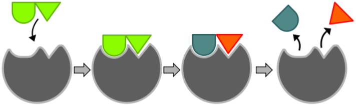
The binder can be a single local name $(\nu x.-)$, or several names with their own affinity network $(\nu M.-)$.

As in the classic π-calculus “cocktail party” model, interacting names can communicate further names, allowing further interactions.

In particular, we use name *extrusion* to model complex formation.



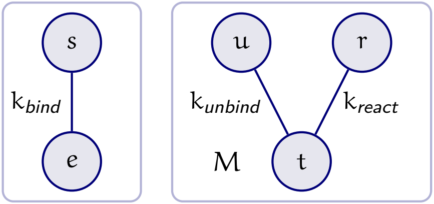
Example Species: Enzyme Catalysis

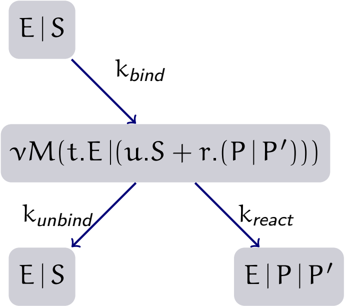


$$S = s(x, y).(x.S + y.(P|P'))$$

$$E = \nu(u, r, t{:}M).(e\langle u, r\rangle.t.E)$$

$$P = P' = \tau@k_{degrade}.0$$





$E\,|\,S$

k_{bind}

$\nu M(t.E\,|\,(u.S + r.(P\,|\,P')))$

k_{unbind}

k_{react}

$E\,|\,S$

$E\,|\,P\,|\,P'$

Formalities: Species and Processes

Species $A, B ::= \ldots$

Processes $P, Q ::= 0 \quad | \quad c \cdot A \quad | \quad P \parallel Q$

Component $c \cdot A$ of species A at concentration $c \in \mathbb{R}_{\geqslant 0}$.

Mixture of processes $P \parallel Q$.

Set $\mathcal{S}$ of species up to structural congruence, and $\mathcal{S}^{\#}$ of *prime* species.

We can identify processes, up to structural congruence, with elements of *process space* $\mathcal{P} = \mathbb{R}^{\mathcal{S}^{\#}}$.

Species embed in process space $\langle - \rangle : \mathcal{S} \to \mathcal{P}$ at unit concentration.

Formalities: Process Semantics

$\frac{dP}{dt}$: Immediate behaviour

- Vector field over process space $\mathcal{P}$
- Equivalent to an ODE system

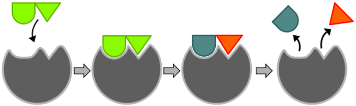
∂P: Interaction potential

- Captures response to available sites
- Rank 3 tensor field over $\mathcal{P}$

$$\frac{d(P \parallel Q)}{dt} = \frac{dP}{dt} + \frac{dQ}{dt} + \partial P \obar \partial Q$$

$$\partial(P \parallel Q) = \partial P + \partial Q$$

Example Process: Enzyme Catalysis



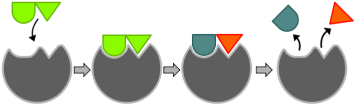
$$S = s(x, y).(x.S + y.(P|P'))$$

$$E = \nu(u, r, t{:}M).(e\langle u, r\rangle.t.E)$$

$$P = P' = \tau@k_{\mathit{degrade}}.0$$

$$c_S \cdot S \parallel c_E \cdot E$$

Example Process: Enzyme Catalysis



$$S = s(x, y).(x.S + y.(P|P'))$$

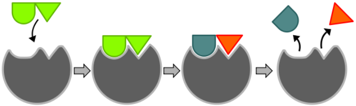
$$E = \nu(u, r, t{:}M).(e\langle u, r\rangle.t.E)$$

$$P = P' = \tau@k_{degrade}.0$$

$$c_S \cdot S \parallel c_E \cdot E$$

```
enzyme.cpi
...
species E() = {
  site t, u, r;
...
```

Example Process: Enzyme Catalysis

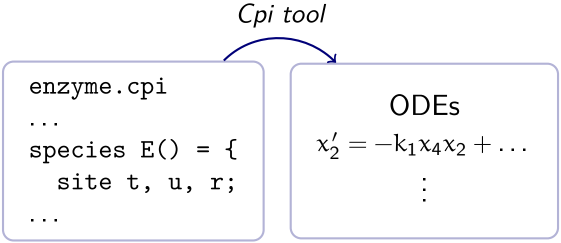


$$S = s(x,y).(x.S + y.(P|P'))$$

$$E = \nu(u,r,t{:}M).(e\langle u,r\rangle.t.E)$$

$$P = P' = \tau@k_{degrade}.0$$

$$c_S \cdot S \parallel c_E \cdot E$$



Cpi tool

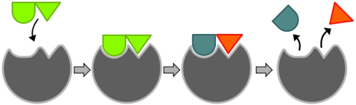
```
enzyme.cpi
...
species E() = {
  site t, u, r;
...
```

ODEs

$$x_2' = -k_1 x_4 x_2 + \ldots$$

$$\vdots$$

Example Process: Enzyme Catalysis

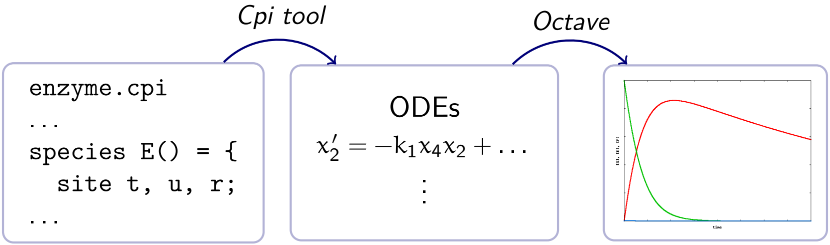


$$S = s(x,y).(x.S + y.(P|P'))$$

$$E = \nu(u,r,t{:}M).(e\langle u,r\rangle.t.E)$$

$$P = P' = \tau@k_{degrade}.0$$

$$c_S \cdot S \parallel c_E \cdot E$$



```
enzyme.cpi
...
species E() = {
  site t, u, r;
...
```

Cpi tool

ODEs

$$x_2' = -k_1 x_4 x_2 + \dots$$

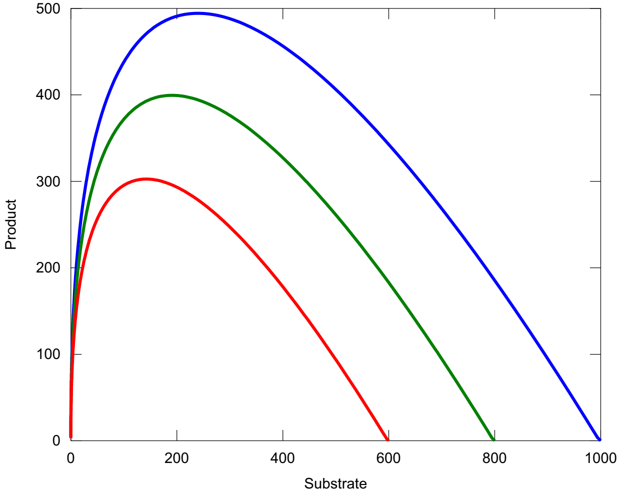
$$\vdots$$

Octave

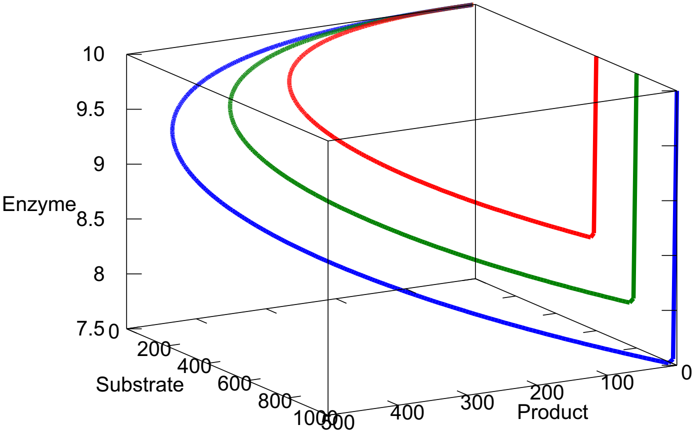
Tool Syntax

```
const kbind=1e−3;    const kreact=2.0;
const kunbind=1.0;   const kdegrade=3e−4;

site e,s; react (e,s)@kbind;

species S() = { body s(;x,y).(x(;).S() + y(;).P());   init 1000.0; }

species E() = { site u,r,t;
                react (u,t)@kunbind;
                react (r,t)@kreact;
                body e(u,r;).act(;).E();
                init 10.0; }

species P() = { body tau<kdegrade>.0; init 0.0; }
```

Process Space: Substrate & Product



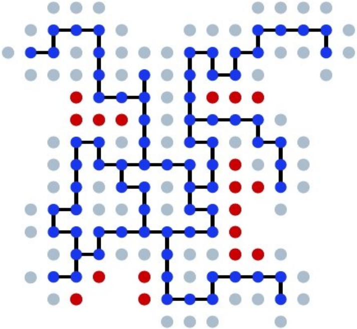
Process Space: Substrate & Product & Enzyme



Examples

- Enzyme catalysis
- Competitive and noncompetitive inhibition
- KaiABC circadian cycle in the blue-green algae Synechococcus Elongatus
- MAPK signalling cascade

Remember Neutral Spaces?



We need:

1. genotype space
   (done: $c\pi$ models)
2. phenotype space
   (done: model dynamics)
3. a mapping between the two
   (done: ODE extraction)
4. accessibility relation

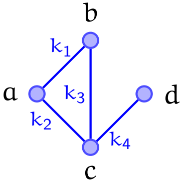
Variation Operators

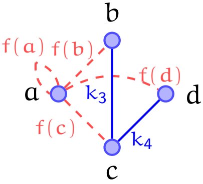
Variation operators are transformations of $c\pi$models which correspond to evolutionary events.

Variation Operators

Variation operators are transformations of $c\pi$models which correspond to evolutionary events.

Example: site reconfiguration

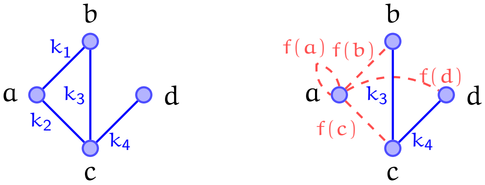




Variation Operators

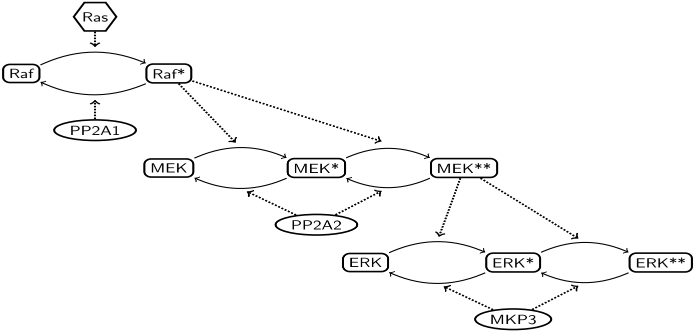
Variation operators are transformations of cπmodels which correspond to evolutionary events.

Example: site reconfiguration



We have defined a dozen operators modelling gene duplications, gene knockouts, changes in expression levels, and more.

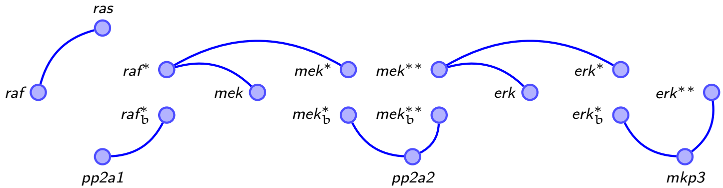
The MAPK Cascade



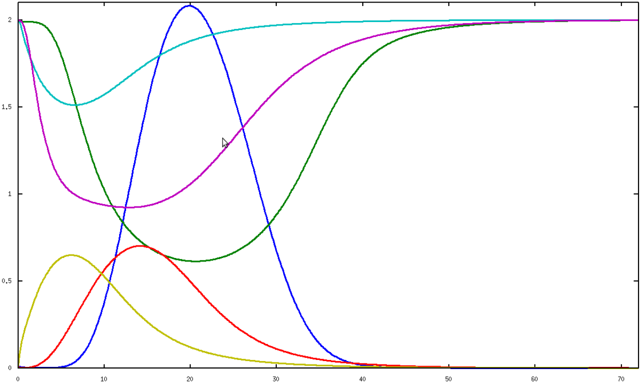
- Functionally conserved in most animals
- Crucial component of signal transduction pathways
- Relays and amplifies a signal
- Benchmark for new modelling techniques

MAPK in $c\pi$

$$
\begin{aligned}
\mathit{Ras} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{ras}(x;y).(\overline{x}.\mathit{Ras} + y.\mathit{Ras})\\
\mathit{Raf} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{raf}(x;y).(\overline{x}.\mathit{Raf} + y.\mathit{Raf}^*)\\
\mathit{Raf}^* &\triangleq (\boldsymbol{\nu} x—\overline{x})(\boldsymbol{\nu} z—\overline{z})(\mathit{raf}^*(x;y).(\overline{x}.\mathit{Raf}^* + y.\mathit{Raf}^*) + \mathit{raf}^*_{\mathrm{b}}(z;y).(\overline{z}.\mathit{Raf}^* + y.\mathit{Raf}))\\
\mathit{PP2A1} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{pp2a1}(x;y).(\overline{x}.\mathit{PP2A1} + y.\mathit{PP2A1})\\
\mathit{MEK} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{mek}(x;y).(\overline{x}.\mathit{MEK} + y.\mathit{MEK}^*)\\
\mathit{MEK}^* &\triangleq (\boldsymbol{\nu} x—\overline{x})(\boldsymbol{\nu} z—\overline{z})(\mathit{mek}^*(x;y).(\overline{x}.\mathit{MEK}^* + y.\mathit{MEK}^{**}) + \mathit{mek}^*_{\mathrm{b}}(z;y).(\overline{z}.\mathit{MEK}^{**} + y.\mathit{MEK}^*))\\
\mathit{MEK}^{**} &\triangleq (\boldsymbol{\nu} x—\overline{x})(\boldsymbol{\nu} z—\overline{z})(\mathit{mek}^{**}(x;y).(\overline{x}.\mathit{MEK}^{**} + y.\mathit{MEK}^{**}) + \mathit{mek}^{**}_{\mathrm{b}}(z;y).(\overline{z}.\mathit{MEK}^{**} + y.\mathit{MEK}^*))\\
\mathit{PP2A2} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,pp2a2(x;y).(\overline{x}.\mathit{PP2A2} + y.\mathit{PP2A2})\\
\mathit{ERK} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{erk}(x;y).(\overline{x}.\mathit{ERK} + y.\mathit{ERK}^*)\\
\mathit{ERK}^* &\triangleq (\boldsymbol{\nu} x—\overline{x})(\boldsymbol{\nu} z—\overline{z})(\mathit{erk}^*(x;y).(\overline{x}.\mathit{ERK}^* + y.\mathit{ERK}^{**}) + \mathit{erk}^*_{\mathrm{b}}(z;y).(\overline{z}.\mathit{ERK}^{**} + y.\mathit{ERK}^*))\\
\mathit{ERK}^{**} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{erk}^{**}_{\mathrm{b}}(x;y).(\overline{x}.\mathit{ERK}^{**} + y.\mathit{ERK}^*)\\
\mathit{MKP3} &\triangleq (\boldsymbol{\nu} x—\overline{x})\,\mathit{mkp3}(x;y).(\overline{x}.\mathit{MKP3} + y.\mathit{MKP3})\\
\boldsymbol{\Pi} &\overset{\acute{}}{=} \mathbf{c}_1 \cdot \boldsymbol{\mathit{Raf}} \parallel \mathbf{c}_2 \cdot \boldsymbol{\mathit{Ras}} \parallel \mathbf{c}_3 \cdot \boldsymbol{\mathit{MEK}} \parallel \mathbf{c}_4 \cdot \boldsymbol{\mathit{ERK}} \parallel \mathbf{c}_5 \cdot \boldsymbol{\mathit{PP2A1}} \parallel \mathbf{c}_6 \cdot \boldsymbol{\mathit{PP2A2}} \parallel \mathbf{c}_7 \cdot \boldsymbol{\mathit{MKP3}}
\end{aligned}
$$

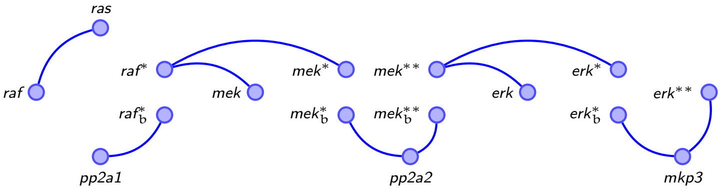


MAPK Behaviour



The tool compiles MAPK into 23 differential equations, which are then solved with Octave. Every reaction acquires emergent Michaelis-Menten kinetics.

Evolutionary Analysis of MAPK



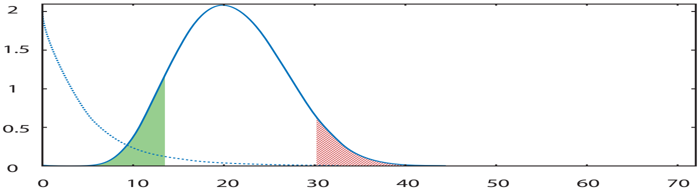
- Reconfigure every site in every way possible (ca. 1M variants)
- Determine the phenotype class of every variant using LTL checking
- Find evolutionarily fragile and robust sites
- Compute the fitness of every variant using signal integration
- Find the distribution of mutation effects on fitness

Phenotype Classes and Fitness

Phenotype classes

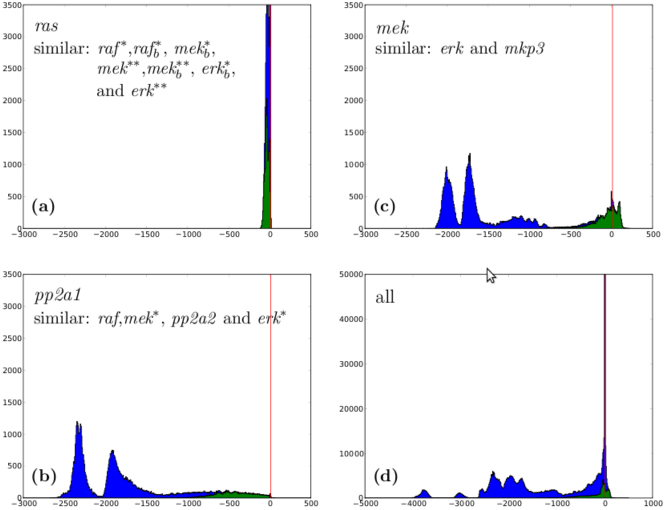
- Four categories: peak, switch, oscillatory, noise.
- Automatically identified using LTL checking.
- Results: peak 7.0%; switch 45.2%; oscillatory 0.0; noise 47.8%.

Fitness

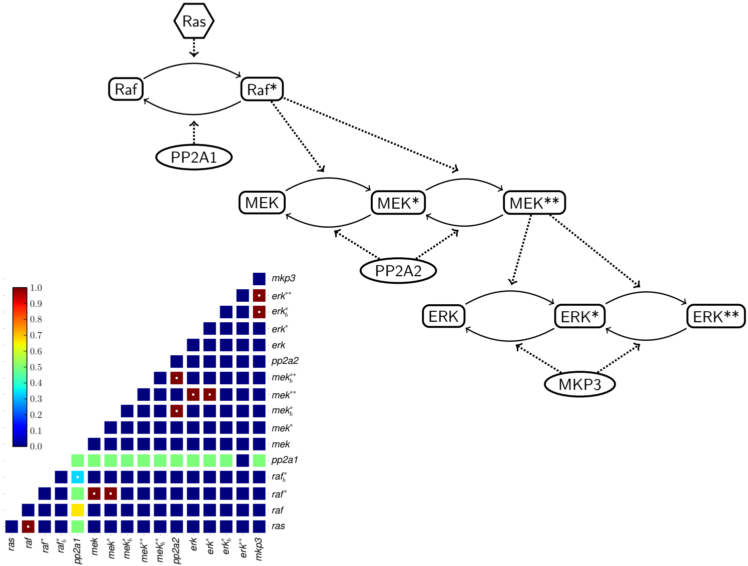


Fitness is the area marked green minus the area marked red.

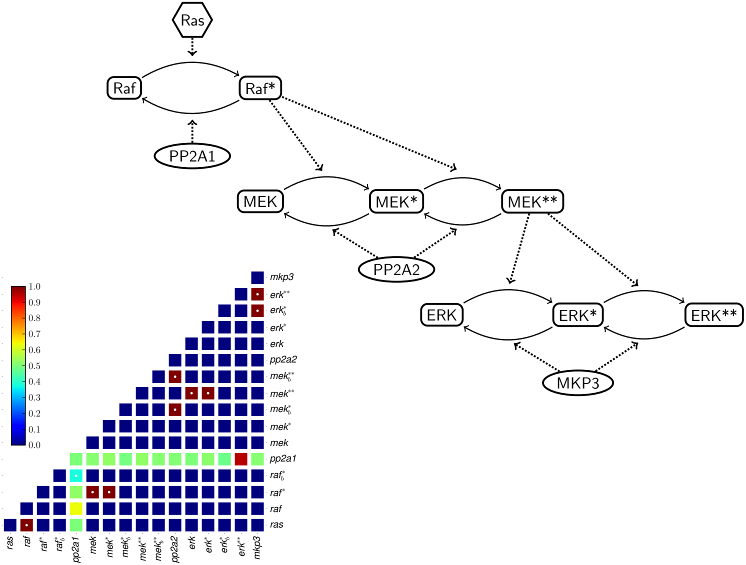
Fitness Distributions



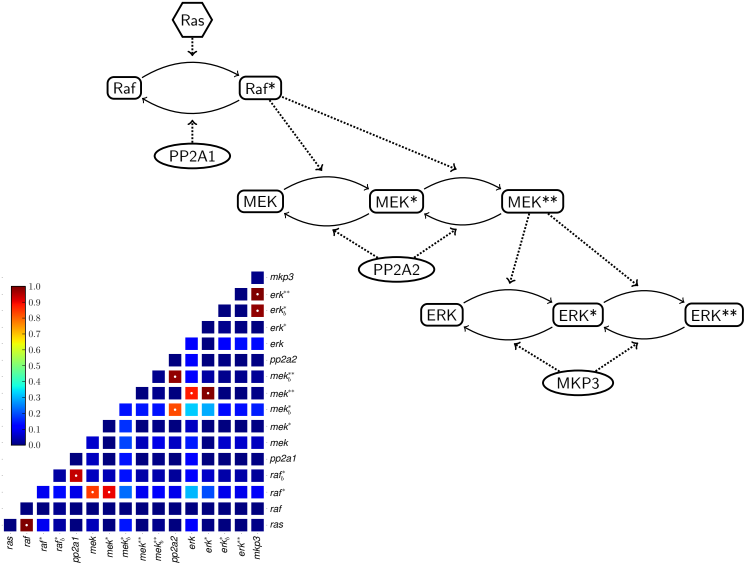
Less Fit Peaks (Left)



Less Fit Peaks (Right)



Advantageous Mutations



Summary

The *continuous π-calculus* ($c\pi$) is a process algebra for modelling behaviour and variation in molecular systems.

It has a structured operational semantics that captures system behaviour as trajectories through a continuous process space, by generating familiar differential-equation models.

We have existing biochemical systems expressed in $c\pi$; in particular, a standard setting of the MAPK cascade.

By systematically exploring neighbourhoods of this basic pathway model, we have been able to identify the robustness and evolvability of its individual components.