# Models and Languages for Computational Systems Biology

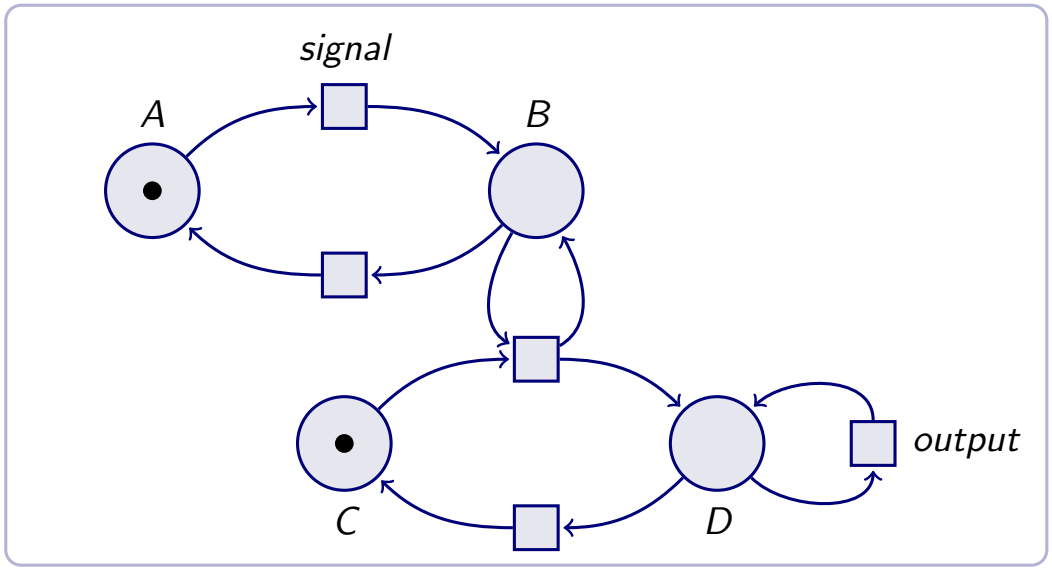
## Lecture 11: Markov Chains and Stochastic Logics

Ian Stark

School of Informatics
The University of Edinburgh

Thursday 18 February 2010
Semester 2 Week 6

# Small cascade example

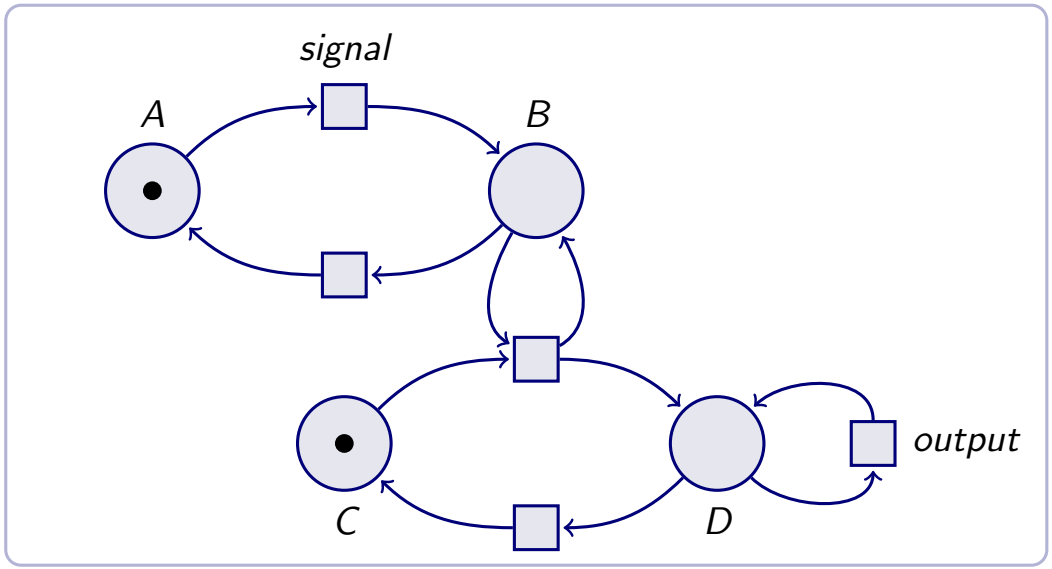


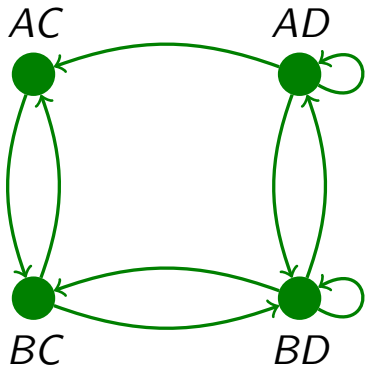

This Petri net represents a portion of a signal transduction cascade:

- An incoming signal promotes $A$ to state $B$;
- this $B$ catalyses conversion of $C$ into $D$;
- in turn, $D$ enables the output signal action on the far right.

Following activation, each component relaxes back to its inactivated form.

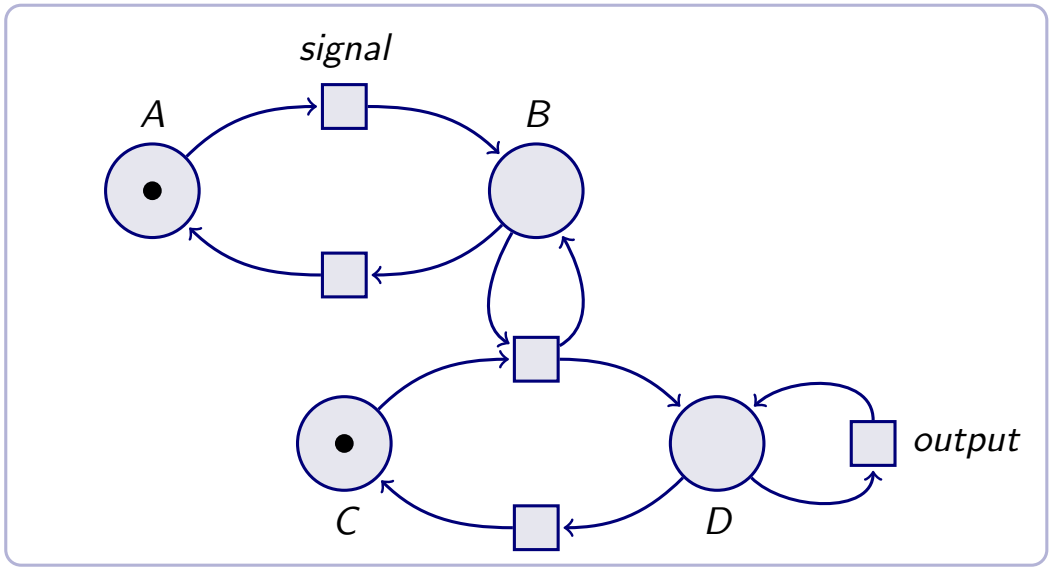
# Small cascade example

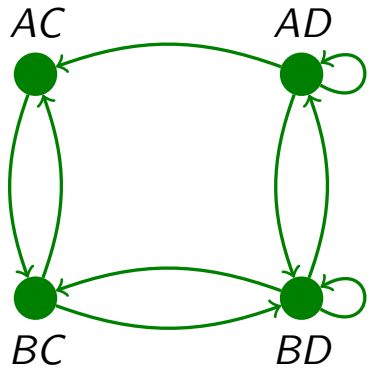



From a Petri net, we can extract a transition system. In this case, the system has a finite set of reachable states, connected as shown.

# Small cascade example

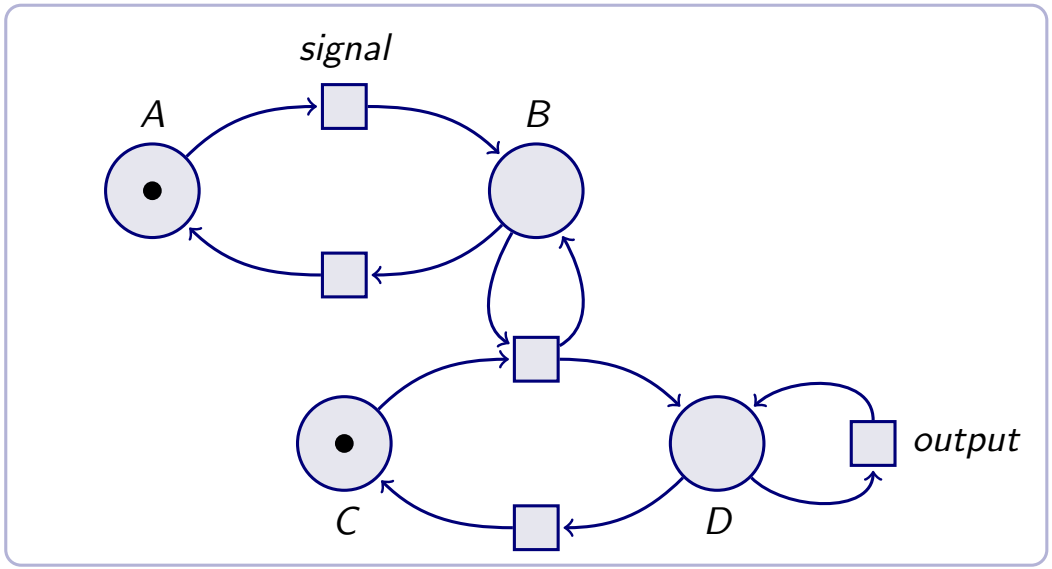


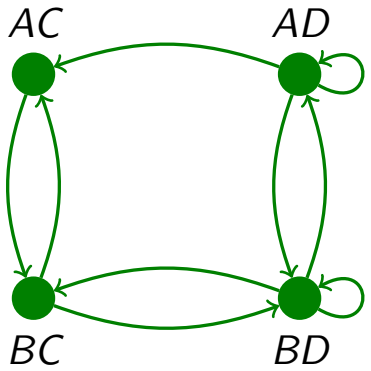




For a transition system, we can evaluate queries about runs of the system:

| | |
|---|---|
| $\mathbf{GF}(\#D > 0)$ | Will the final action be repeatedly enabled? |
| $\mathbf{FG}(\#C = 0)$ | Might $C$ disappear for ever? |

# Small cascade example

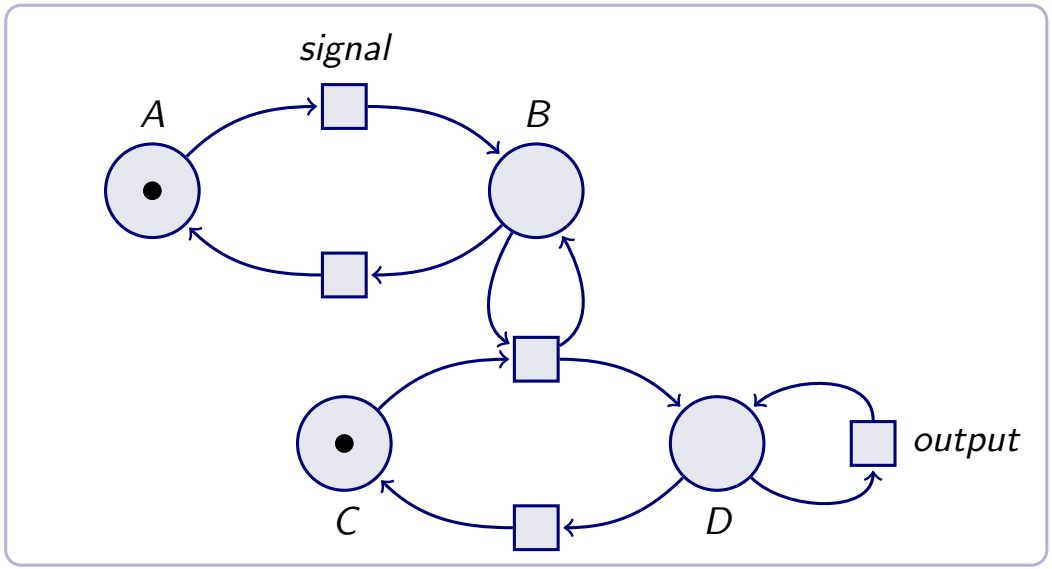


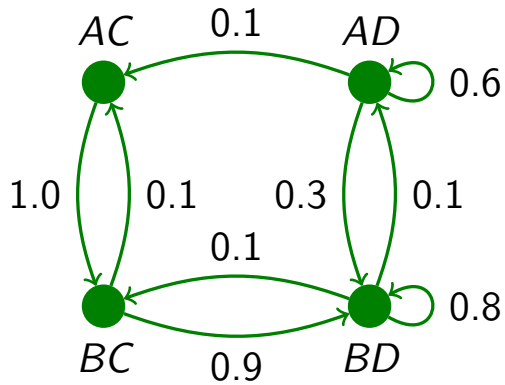




We can also ask whether these hold for all, none, or some paths.

| | |
|---|---|
| $\mathbf{GF}(\#D > 0)$ | Will the final action be repeatedly enabled? |
| $\mathbf{FG}(\#C = 0)$ | Might $C$ disappear for ever? |

# Small cascade example



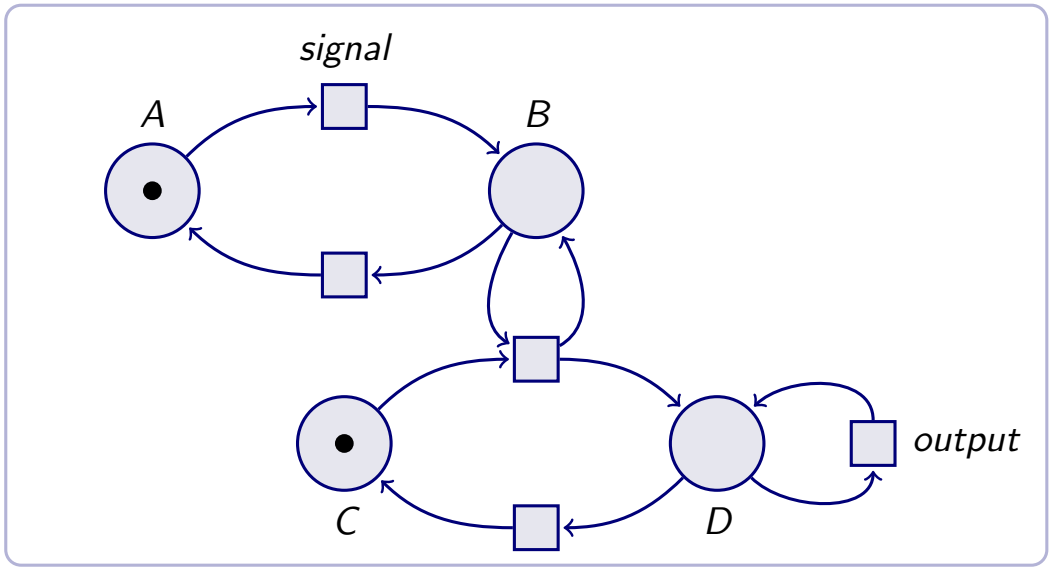



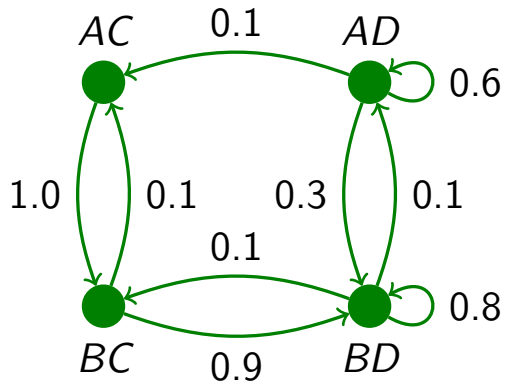

We can also ask whether these hold for all, none, or some paths.

| | |
|---|---|
| $\mathbf{GF}(\#D > 0)$ | Will the final action be repeatedly enabled? |
| $\mathbf{FG}(\#C = 0)$ | Might $C$ disappear for ever? |

As this is a discrete nondeterministic presentation, taking no account of rates or probabilities, answers may sometimes be counter-intuitive.

# Small cascade example

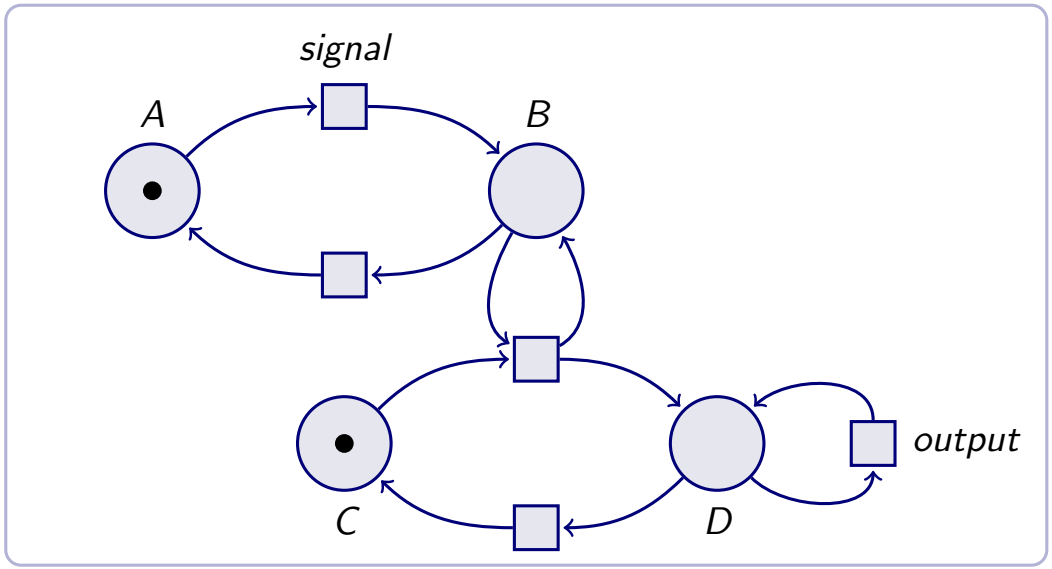


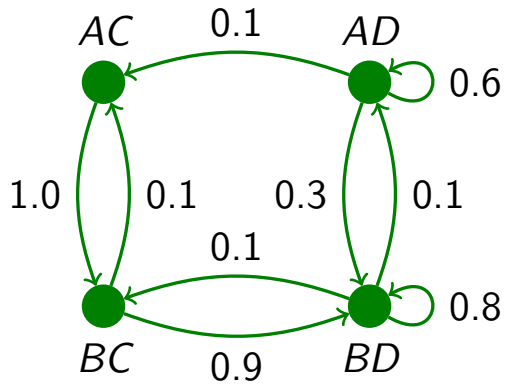




If we add probabilities to transitions, we obtain a (discrete time) Markov chain (DTMC), which can be described by its stochastic matrix.

$$\begin{pmatrix} 0 & 0.1 & 0 & 0.1 \\ 1.0 & 0 & 0.1 & 0 \\ 0 & 0.9 & 0.8 & 0.3 \\ 0 & 0 & 0.1 & 0.6 \end{pmatrix}$$

# Small cascade example



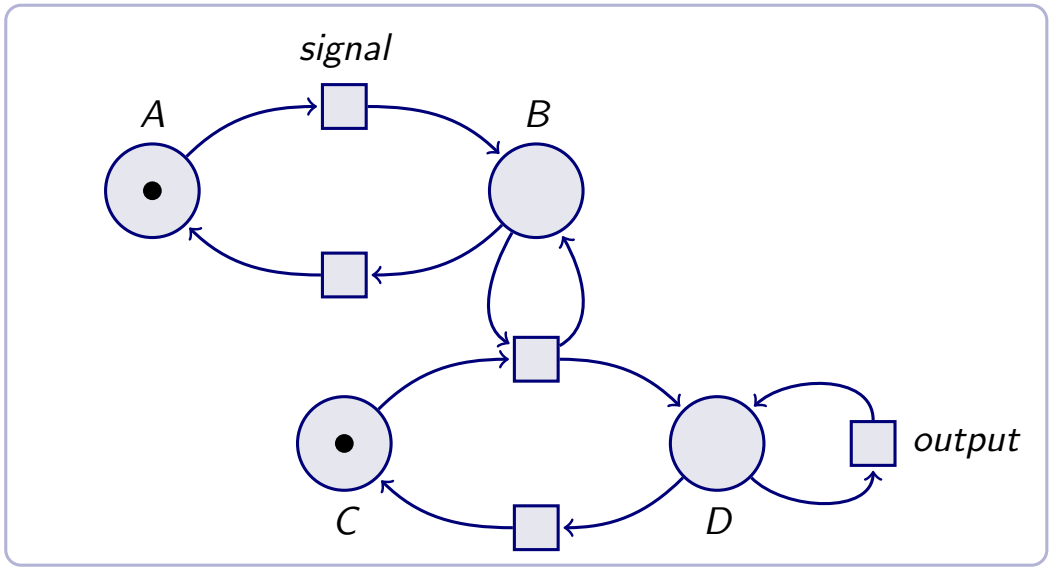

In this case, the system is ergodic and the equilibrium steady-state probabilities are given by the unique eigenvector with eigenvalue 1.

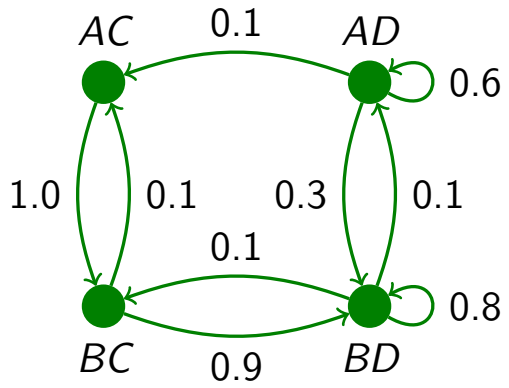
$$\begin{pmatrix} 0 & 0.1 & 0 & 0.1 \\ 1.0 & 0 & 0.1 & 0 \\ 0 & 0.9 & 0.8 & 0.3 \\ 0 & 0 & 0.1 & 0.6 \end{pmatrix}$$

Eigenvalues
$\{1, 0.438,$
$0.332, -0.371\}$

Steady-state $\begin{pmatrix} 0.027 \\ 0.097 \\ 0.701 \\ 0.175 \end{pmatrix}$

# Small cascade example




The set of possible runs is now a *probability space*, where we can measure the probability of certain system behaviours.

| | |
|---|---|
| $\mathbf{GF}(\#D > 0)$ | Will the final action be repeatedly enabled? |
| $\mathbf{FG}(\#C = 0)$ | Might $C$ disappear for ever? |

Instead of true or false, we can now assign these events probabilities. Probabilities of 1 and 0 indicate *almost always* and *almost never*.

# PCTL Syntax

## Probabilistic Computational Tree Logic

$$p ::= \mathrm{Atom} \mid p \wedge q \mid p \vee q \mid p \Rightarrow q \mid \neg p$$
$$\mid \mathbf{F}_{\sim x}(p) \mid \mathbf{G}_{\sim x}(p) \mid p\,\mathbf{U}_{\sim x}\,q \mid p\,\mathbf{R}_{\sim x}\,q$$

Here $\sim \in \{<, \leqslant, \geqslant, >\}$ and $x \in [0, 1]$. Alternatively, write $\mathcal{P}_{\sim x}(p\,\mathbf{U}\,q)$ etc.

H. Hansson and B. Jonsson.
A logic for reasoning about time and reliability.
*Formal Aspects of Computing*, 6(5):512–535, 1994.

# PCTL Syntax

## Probabilistic Computational Tree Logic

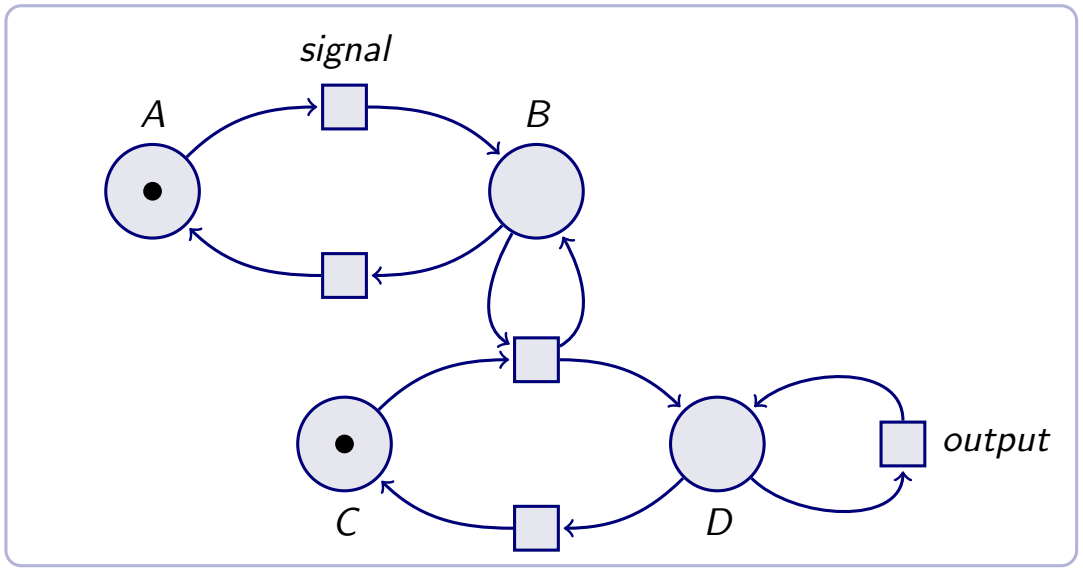
$$\mathrm{p} ::= \mathrm{Atom} \mid \mathrm{p} \wedge \mathrm{q} \mid \mathrm{p} \vee \mathrm{q} \mid \mathrm{p} \Rightarrow \mathrm{q} \mid \neg \mathrm{p}$$
$$\mid \mathbf{F}_{\sim \mathrm{x}}(\mathrm{p}) \mid \mathbf{G}_{\sim \mathrm{x}}(\mathrm{p}) \mid \mathrm{p}\, \mathbf{U}_{\sim \mathrm{x}}\, \mathrm{q} \mid \mathrm{p}\, \mathbf{R}_{\sim \mathrm{x}}\, \mathrm{q}$$

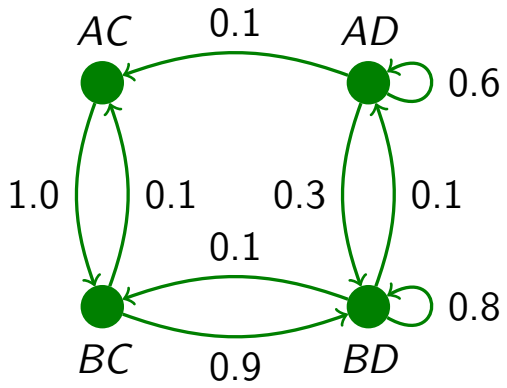
Here $\sim \in \{<, \leqslant, \geqslant, >\}$ and $\mathrm{x} \in [0, 1]$. Alternatively, write $\mathcal{P}_{\sim \mathrm{x}}(\mathrm{p}\, \mathbf{U}\, \mathrm{q})$ etc.

| | |
|---|---|
| $\mathrm{Atom}$ | Some basic predicate over system states |
| $\mathbf{F}_{>\mathrm{x}}(\mathrm{p})$ | $\mathrm{p}$ holds eventually, with probability at least $\mathrm{x}$ |
| $\mathbf{G}_{\leqslant \mathrm{x}}(\mathrm{p})$ | The probability $\mathrm{p}$ holds at all later states is no more than $\mathrm{x}$ |
| $(\mathrm{p}\, \mathbf{U}_{\geqslant 1}\, \mathrm{q})$ | Almost surely, $\mathrm{p}$ holds until finally $\mathrm{q}$ does |

We can recover **AG** with $\mathbf{G}_{\geqslant 1}$ and **EF** with $\mathbf{F}_{>0}$, etc.

# Small cascade example



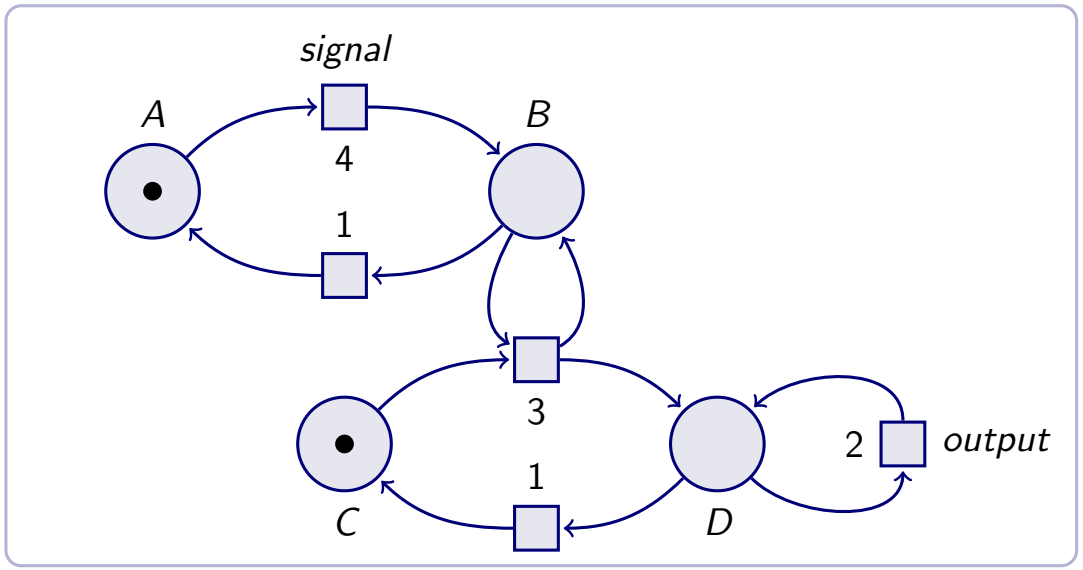

The original questions about runs can be rephrased in PCTL

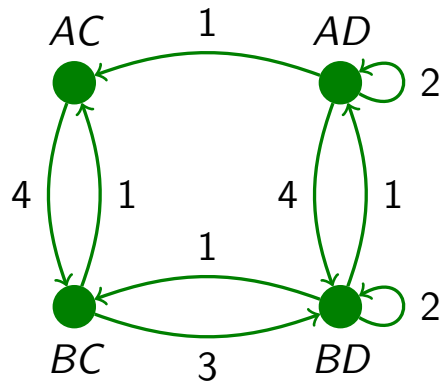
$\mathbf{G}_{\geqslant 1}\mathbf{F}_{\geqslant 1}(\#D > 0)$ The final action is almost sure to be repeatedly enabled

$\mathbf{F}_{\leqslant 0}\mathbf{G}_{>0}(\#C = 0)$ Almost never will $C$ disappear for ever.

However, this still takes no explicit account of the passage of time.

# Small cascade example

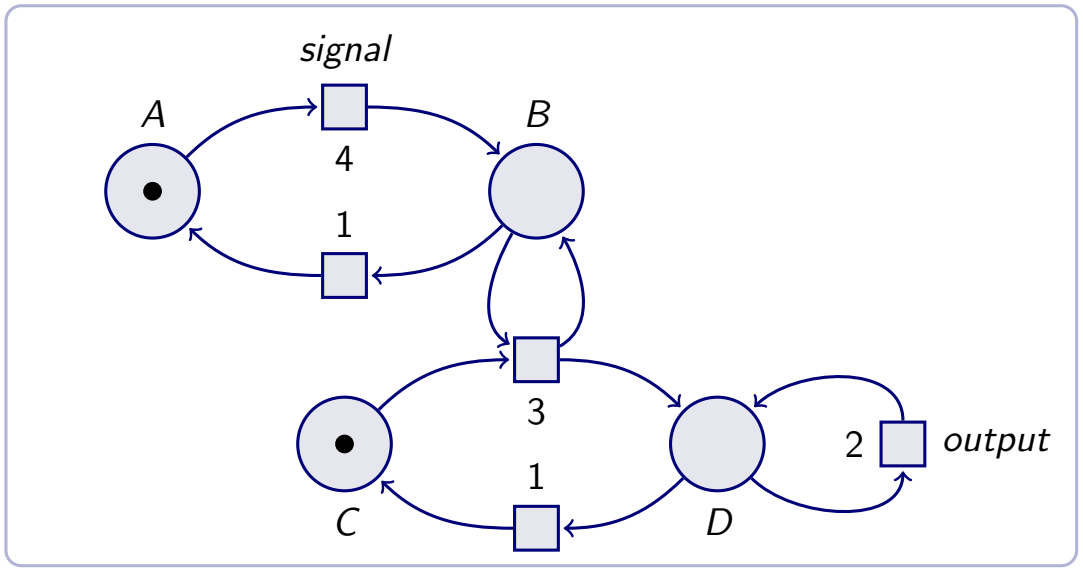


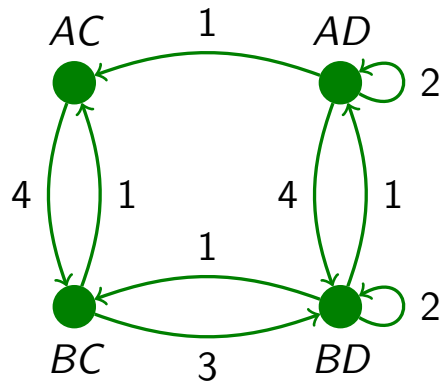




If we add rates to the transitions, to give a stochastic Petri net, we obtain a corresponding continuous-time Markov chain (CTMC), now described by a rate matrix.

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 4 & 0 & 1 & 0 \\ 0 & 3 & 2 & 4 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

# Small cascade example




From this it is possible to compute both transient and steady-state probabilities of system behaviour over time.

- Transient $\mathrm{p_t}(s, s')$ is probability of being in state $s'$ after time $\mathrm{t}$, having started in state $s$.
- Steady-state $\mathrm{p}(s, s')$ is the probability in the long run of being in state $s'$ having started in state $s$.

# CSL Syntax

## Continuous Stochastic Logic

$$p ::= \mathrm{Atom} \mid p \wedge q \mid p \vee q \mid p \Rightarrow q \mid \neg p$$
$$\mid \mathcal{P}_{\sim x}(p\ \mathbf{U}^{I}\ q) \mid \mathcal{S}_{\sim x}(p) \mid \ldots$$

Here $\sim \in \{<, \leqslant, \geqslant, >\}$ and $x \in [0,1]$ and interval $I \subseteq [0,\infty)$.

C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen.
Model-checking algorithms for continuous-time Markov chains.
*IEEE Transactions on Software Engineering*, 29(6):524–541, June 2003.

# CSL Syntax

## Continuous Stochastic Logic

$$\mathrm{p} ::= \mathrm{Atom} \mid \mathrm{p} \wedge \mathrm{q} \mid \mathrm{p} \vee \mathrm{q} \mid \mathrm{p} \Rightarrow \mathrm{q} \mid \neg \mathrm{p}$$
$$\mid \mathcal{P}_{\sim \mathrm{x}}(\mathrm{p}\ \mathbf{U}^{\mathrm{I}}\ \mathrm{q}) \mid \mathcal{S}_{\sim \mathrm{x}}(\mathrm{p}) \mid \ldots$$

Here $\sim \in \{<, \leqslant, \geqslant, >\}$ and $\mathrm{x} \in [0, 1]$ and interval $\mathrm{I} \subseteq [0, \infty)$.

| | |
|---|---|
| $\mathcal{P}_{>\mathrm{x}}(\mathrm{p}\ \mathbf{U}^{\mathrm{I}}\ \mathrm{q})$ | There is probability at least $\mathrm{x}$ that $\mathrm{p}$ holds until a point, during the interval $\mathrm{I}$, where $\mathrm{q}$ does. |
| $\mathcal{P}_{\geqslant 0.5}(\mathbf{F}^{[0,\mathrm{t}]}(\mathrm{p}))$ | There is a probability of no less than 0.5 that property $\mathrm{p}$ will hold within time $\mathrm{t}$. |
| $\mathcal{S}_{\leqslant 0.2}(\mathrm{p})$ | The probability that property $\mathrm{p}$ will hold in the steady-state is no more than 0.2. |

## PRISM: The Probabilistic Symbolic Model Checker

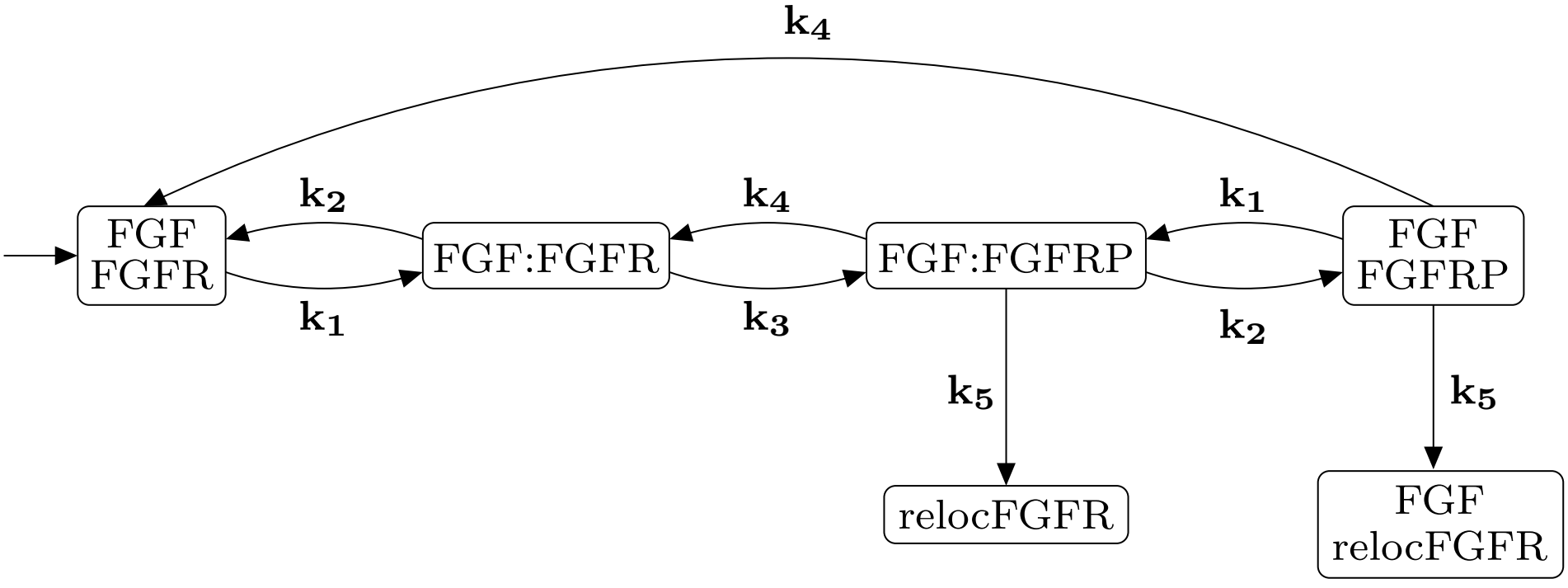
http://www.prismmodelchecker.org

> PRISM is a probabilistic model checker, a tool for formal modelling and analysis of systems which exhibit random or probabilistic behaviour. It supports three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs), plus extensions of these models with costs and rewards.
>
> PRISM has been used to analyse systems from a wide range of application domains, including communication and multimedia protocols, randomised distributed algorithms, security protocols, biological systems and many others.

# PRISM in systems biology

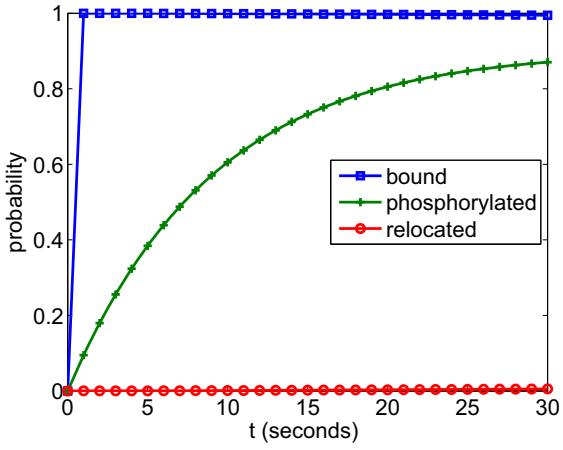
Excerpts from:

Marta Kwiatkowska, Gethin Norman, and David Parker.
Probabilistic model checking for systems biology.
In *Symbolic Systems Biology: Theory and Methods*. Jones and Bartlett, 2010. To appear.

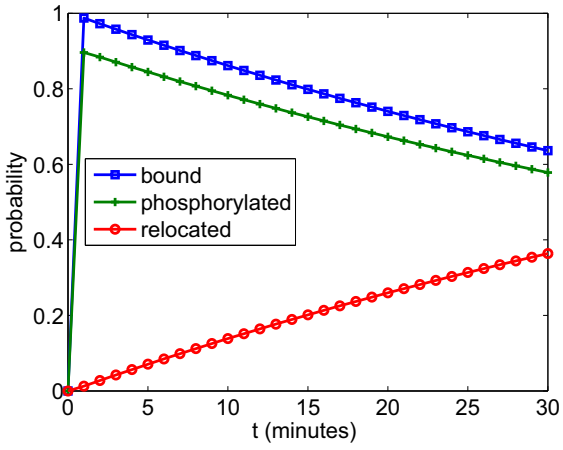

k4
k2
k4
k1
FGF
FGFR
FGF:FGFR
FGF:FGFRP
FGF
FGFRP
k1
k3
k2
k5
k5
relocFGFR
FGF
relocFGFR

```
const double k5 = 1/(60 * 60); // rate of relocation

module FGF

    fgf : [0..2] init 0; // 0 - free, 1 - bound, 2 - removed from system

    [bind] fgf=0 → (fgf'=1); // FGF and FGFR bind
    [rel] fgf=1 → (fgf'=0); // FGF and FGFR unbind
    [reloc] fgf=1 → (fgf'=2); // FGF disappears since bound when FGFR relocates

endmodule
```

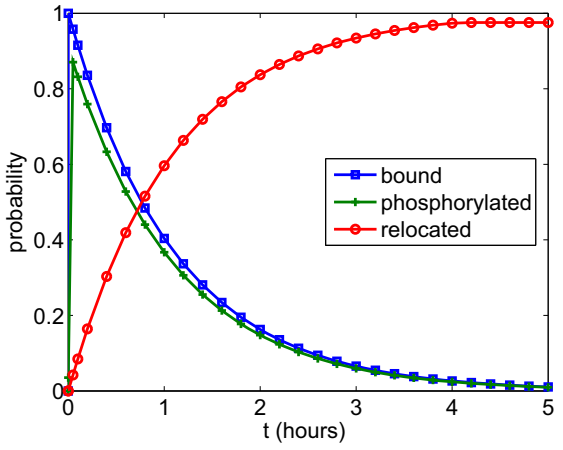
- $(phos{=}1) \Rightarrow \mathtt{P}_{>0.1}[\ \mathtt{true}\ \mathtt{U}^{[0,t]}\ (reloc{=}1)\ ]$ - 'if FGFR is currently phosphorylated, then the probability of it being relocated within the next $t$ seconds is greater than 0.1';
- $(phos{=}1 \wedge fgfr{=}0) \Rightarrow \mathtt{P}_{\leq 0.2}[\ (fgfr{=}0)\ \mathtt{U}\ (reloc{=}1)\ ]$ - 'if FGFR is phosphorylated and free, then the probability of it being relocated before binding to FGF is at most 0.2';
- $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}^{[t,t]}\ (fgf{=}1)\ ]$ - 'the probability that FGF is bound to FGFR at time instant $t$ (i.e. after exactly $t$ seconds)';
- $\mathtt{P}_{=?}[\ \mathtt{true}\ \mathtt{U}\ (reloc{=}1 \wedge fgf{=}2)\ ]$ - 'the probability that FGFR relocates and FGF is bound when relocation occurs';
- $\mathtt{S}_{=?}[\ (fgf{=}0)\ ]$ - 'the probability that, in the long run, FGF is free';




(a) Time scale seconds




(b) Time scale minutes




(c) Time scale hours

**Fig. 7.** Transient properties of FGFR for the model of Figure 4

## Homework

Read the remainder, §§4–6, of the tutorial article.

Marta Kwiatkowska, Gethin Norman, and David Parker.
Probabilistic model checking for systems biology.
In *Symbolic Systems Biology: Theory and Methods*. Jones and Bartlett, 2010. To appear.

Lucas Laursen.
Computational biology: Biological logic.
Nature 462:408–410, 2009.

> In the meantime, however, Fisher and her fellow executable-biology enthusiasts have a lot of convincing to do, says Stephen Oliver, a biologist at the University of Cambridge, UK. "Modelling in general is regarded sceptically by many biologists," he points out.

Do you think this is true? Can you find any evidence for or against?