# Simulation and verification for computational modelling of signalling pathways

Marta Kwiatkowska


Computing Laboratory, University of Oxford
http://www.comlab.ox.ac.uk/people/marta.kwiatkowska.html

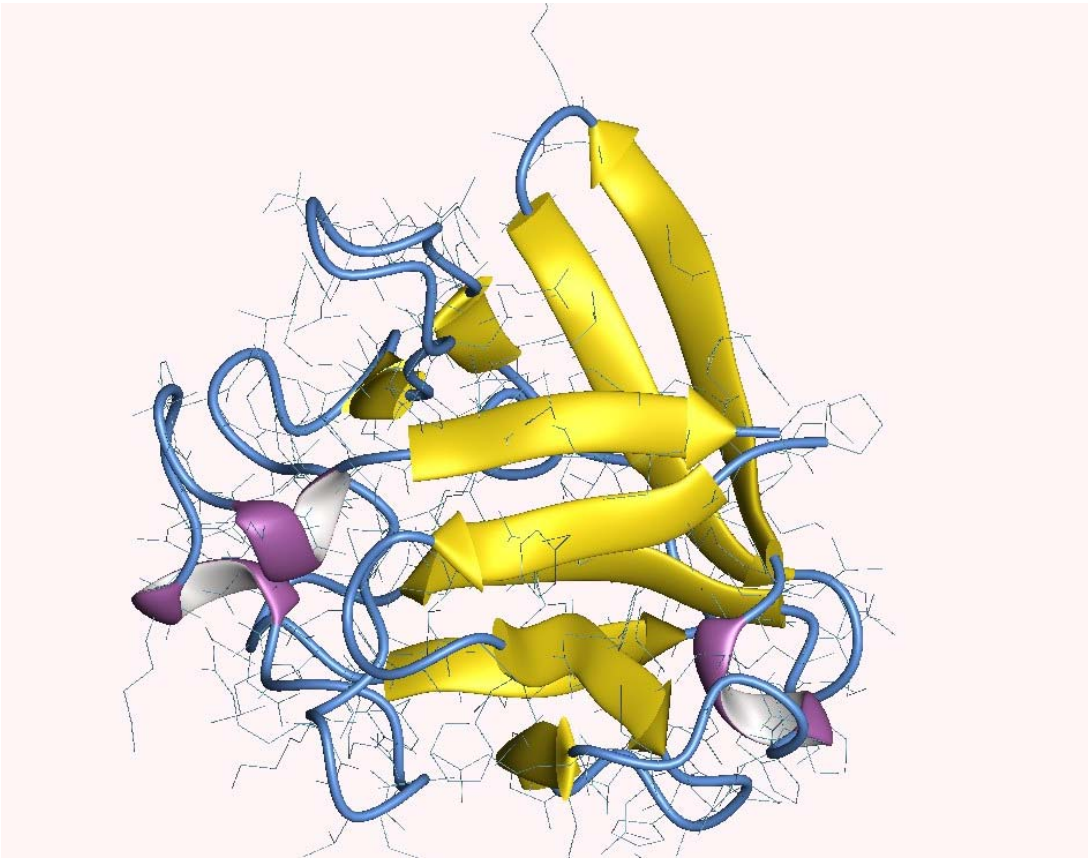
Algorithmic Bioprocesses, Leiden, 6th December 2007

# Overview

- Modelling frameworks for biological signalling
  - Continuous deterministic vs discrete stochastic approach
  - Process calculi as description formalisms
- Modelling formalisms and analysis techniques
  - Continuous time Markov chain models
  - The temporal logic CSL
  - Simulation vs verification
  - Probabilistic model checking
- Case study
  - FGF signalling: some results
- Future challenges

# Modelling signalling pathways

- Focus on
  - networks of molecules
  - interaction
  - continuous & discrete dynamics
- Rather than
  - geometry
  - structure
  - sequence



Google images: Human FGF, http://160.114.99.91/astrojan/prot1t.htm

# Why process calculus?

- Language for modelling networks of objects
  - Compact description for networks of interacting objects
  - Molecule-centred, but can be used at all levels
    - molecular, cellular, tissue
  - Ease of textual manipulation: add/remove/modify reaction
- Calculus and algebra for processes
  - Compositional models: networks built from molecules
  - Induce graph-theoretic representations
  - Stochastic variants generate continuous time Markov chains
  - Proof rules for reasoning about process behaviour
  - Algebraic laws
- Support powerful analysis methods
  - Simulation, to obtain individual trajectories
  - Verification, reasoning about all possible behaviours

# Addressing a real need…

"We have no real 'algebra' for describing regulatory circuits across different systems..."

– T. F. Smith (TIG 14:291-293, 1998)

"The data are accumulating and the computers are humming, what we are lacking are the words, the grammar and the syntax of a new language…"

– D. Bray (TIBS 22:325-326, 1997)

- NB Regev, Shapiro, Priami, Cardelli, etc, propose that existing process calculi developed in computer science can be exploited in systems biology
- Specially developed formalisms (e.g. beta-binders, kappa) are similar rule-based formalisms inspired by biology

# Modelling frameworks

- Assume wish to model mixture of molecules
  - N different molecular species, interact through reactions
  - fixed volume V (spatially uniform), constant pressure and temperature
- Continuous deterministic approach
  - approximate the number of molecules in V at time t by a continuous function, if large numbers of molecules
  - obtain ODEs (ordinary differential equations)
  - not for individual runs, but average
- Discrete stochastic approach
  - discrete system evolution, via discrete events for reactions
  - obtain discrete-state stochastic process

# Discrete stochastic approach

- Work with discrete states as vectors $\underline{x}$ of molecule counts for each species
  - probability $P(\underline{x},t)$ that at time t there will be $\underline{x}_A$ of species A
- A useful fact
  - if constant state-dependent rates, obtain Continuous time Markov chain (CTMC)
  - therefore, can use stochastic process calculi (which induce CTMCs) as model description formalisms
- The stochastic approach admits
  - discrete event simulation
  - and is realistic for a single time course evolution, not just average
- Can we achieve more with techniques developed in computer science?
  - model reductions, temporal logic, model checking…

# Typical modelling/analysis approach…

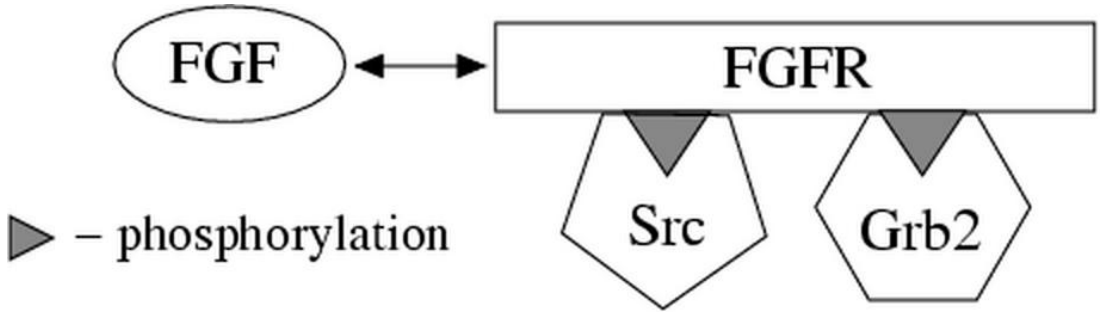
- Use biochemical reactions as basis
  - Write in SBML (Systems Biology Mark Up Language)
  - Share models on WWW, access a range of solvers
    - Analytical solutions rarely feasible
    - ODE and stochastic models generated automatically
    - e.g. Gillespie, for population-based discrete stochastic models
- Choose a formalism, model directly
  - Discrete models: graphical notations (Petri nets), textual (process calculi, rewrite systems), or their stochastic extension…
  - Continuous: ODEs, PDEs
- Choose an analysis method
  - Simulation, if focus is on time course trajectories
  - Otherwise, more powerful methods needed (bifurcation, etc)

# Our approach

- Consider a hypothesis about interaction between molecular species in a signalling pathway
  - write down the set of reactions, which could be hypothetical
  - obtain a set of ODEs from reactions, plot time trajectories for average concentrations (e.g. Cellerator)
  - model using process calculi, simulate to obtain individual time trajectories (e.g. BioSPI, SPiM for stochastic pi-calculus)
- Analyse via (probabilistic) model checking, in addition to simulation
  - detailed exhaustive analysis, i.e. for all configurations
  - can definitively establish causal relationships
  - wide range of quantitative properties
- Trade-offs
  - suffers from exponential state explosion problems
  - in comparison, ODE approaches may exhibit exponential growth in number of equations

# Case study: fragment of FGF pathway

- Fragment of Fibroblast Growth Factor (FGF) pathway
  - regulator of skeletal development, e.g. number of digits




- Biological challenges
  - unknown function of molecules, model different hypotheses
  - expensive experimental scenarios
- Aim to develop ODE and discrete stochastic models
  - ODE: use Cellarator & Mathematica
  - discrete: simulation (BioSPI, SPiM), verification (PRISM)

# The reactions

1: FGF binds/releases FGFR

$FGF + FGFR \rightarrow FGFR{:}FGF$ $\quad k_1 = 5e{+}8\ M^{-1}s^{-1}$

$FGF + FGFR \leftarrow FGFR{:}FGF$ $\quad k_2 = 0.002\ s^{-1}$

2: Phosphorylation of FGFR (whilst FGFR:FGF)

$FGFR \rightarrow FGFRP$ $\quad k_3 = 0.1\ s^{-1}$

3: Dephosphorylation of FGFR

$FGFRP \rightarrow FGFR$ $\quad k_4 = 0.01\ s^{-1}$

4: Relocation of FGFR (whilst FGFRP)

$FGFR \rightarrow relocFGFR$ $\quad k_5 = 1/60\ min^{-1}$

# PRISM language fragment

```
module fgfr

    fgfr  : [0..1] init 0; // 0 – free, 1 – bound
    phos : [0..1] init 0; // 0 – unphosphorylated, 1 – phosphorylated
    reloc : [0..1] init 0; // 0 – not relocated, 1– relocated

    [bnd] reloc=0 ∧ fgfr=0 → k1 : (fgfr'=1); // FGF and FGFR bind
    [rel]   reloc=0 ∧ fgfr=1 → k2 : (fgfr'=0); // FGF and FGFR release
    [] reloc=0 ∧ fgfr =1 ∧ phos =0 → k3 : (phos'=1); // FGFR phosphor.
    [] reloc=0 ∧ phos=1     → k4 : (phos'=0); // FGFR dephosphorylates
    [] reloc=0 ∧ phos=1     → k5 : (reloc'=1); // FGFR relocates

endmodule
```

# The induced CTMC

The reactions

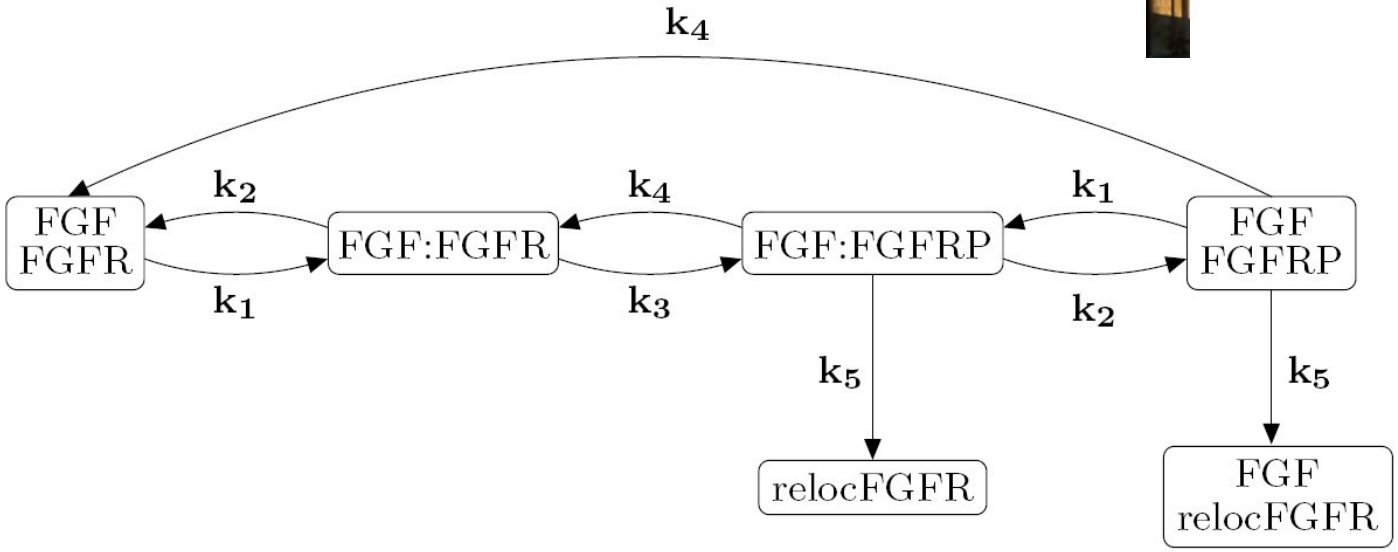
1: FGF binds/releases FGFR
FGF + FGFR → FGFR:FGF $k_1$=5e+8 $M^{-1}s^{-1}$
FGF + FGFR ← FGFR:FGF $k_2$=0.002 $s^{-1}$

2: Phosphorylation of FGFR (whilst FGFR:FGF)
FGFR → FGFRP $k_3$=0.1 $s^{-1}$

3: Dephosphorylation of FGFR
FGFRP → FGFR $k_4$=0.01 $s^{-1}$

4: Relocation of FGFR (whilst FGFRP)
FGFR → relocFGFR $k_5$=1/60 $min^{-1}$

- Individual-based model




- Stochastic rates $k_i$ derived from kinetic rates as usual
- Can decorate states/transitions with costs/rewards

# Fragment of ODEs

$$
\begin{aligned}
Fgfr'(t) &= -\,bind \cdot Fgf(t) \cdot Fgfr(t) \\
&\quad +\,rel \cdot Fgfr\_Fgf(t) \\
&\quad +\,dph \cdot FgfrP\,(t) \\
FgfrP'(t) &= -\,bind \cdot Fgf(t) \cdot FgfrP(t) \\
&\quad +\,rel \cdot FgfrP\_Fgf(t) \\
&\quad -\,dph \cdot FgfrP\,(t) \\
&\quad +\,reloc \cdot FgfrP(t) \\
&\quad +\,reloc \cdot FgfrP\_Fgf(t) \\
Fgfr\_Fgf'(t) &= -\,rel \cdot Fgfr\_Fgf(t) \\
&\quad +\,bind \cdot Fgf(t) \cdot Fgfr(t) \\
&\quad -\,ph \cdot Fgfr\_Fgf\,(t) \\
&\quad +\,dph \cdot FgfrP\_Fgf\,(t) \\
&\ldots
\end{aligned}
$$

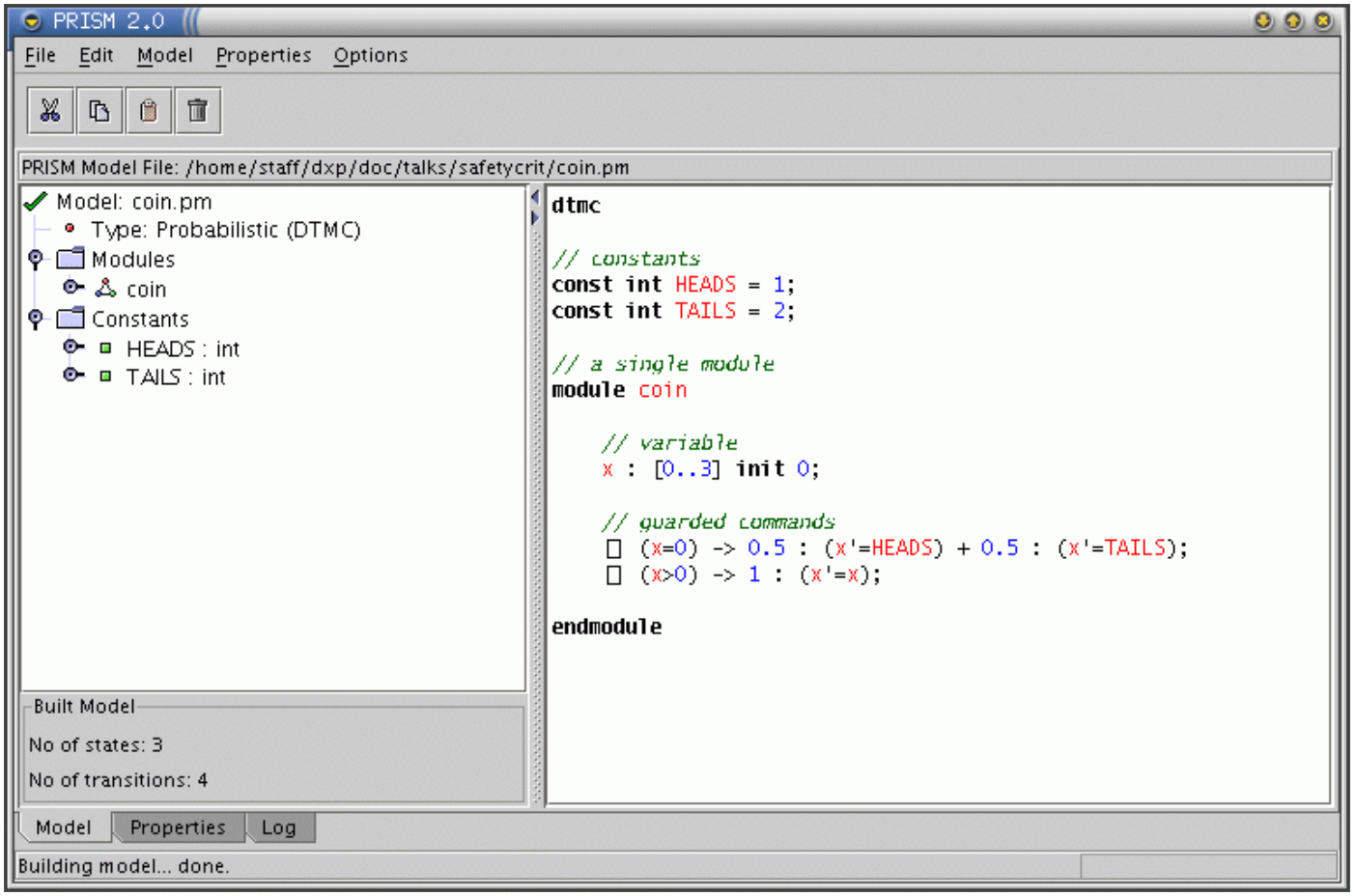
# SBML code fragment

```
<listOfSpecies>
    <species id="FGFR" initialConcentration="1" ... />
    <species id="FGF" initialConcentration="1" ... /> ...
</listOfSpecies>
<reaction id="Reaction1" reversible="true">
    <listOfReactants>
        <speciesReference species="FGFR" />...
    </listOfReactants>
    <listOfProducts>
        <speciesReference species="FGFR_FGF" /> ...
    </listOfProducts>
    <kineticLaw><math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply> <times/>
            <ci>k1</ci> <ci>FGFR</ci> <ci>FGF</ci>
        </apply>
    </math></kineticLaw>
</reaction>
```

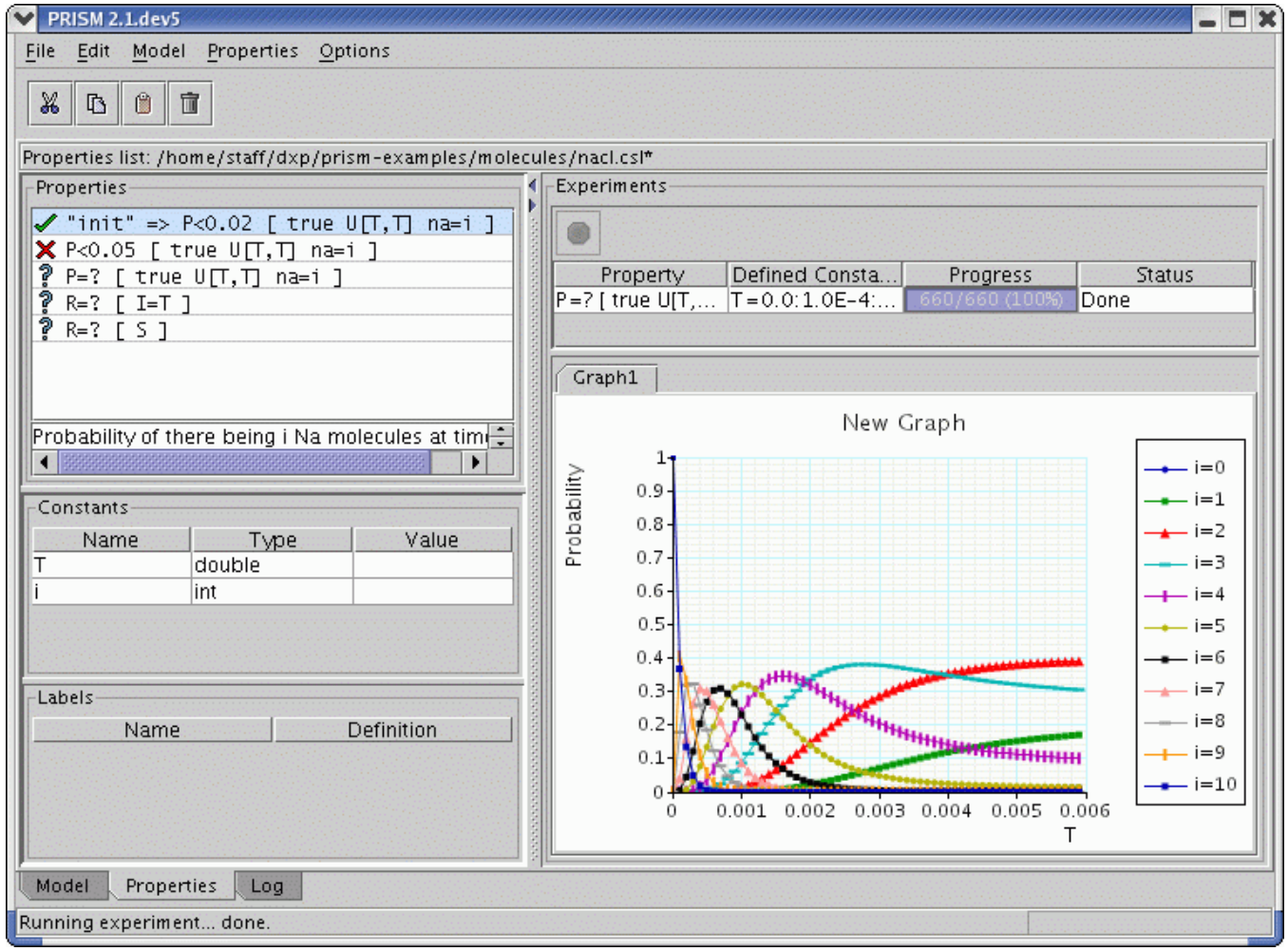
# The PRISM model checker: Overview

- Simple, discrete stochastic modelling language
  - networks formed from interacting modules
  - interactions are associated with state-dependent rates
  - similar expressive power to variants of stochastic pi-calculus
- Specifications given in temporal logic:
  - what is the probability that concentration is less than min at time 10?
    $\mathbf{P}_{=?}$ [true $\mathbf{U}^{[10,10]}$ c<min]
  - what is the probability the concentration reaches min?
    $\mathbf{P}_{=?}$ [true $\mathbf{U}$ c≥min]
  - in the long run, what is the probability that the concentration remains stable between min and max
    $\mathbf{S}_{=?}$ [(c≥min)∧(c≤max)]
- Numerous case studies and errors detected in computer network protocols
- SBML enabled
- See www.prismmodelchecker.org

# Screenshot: Text editor



# Screenshot: Graphs



# How does PRISM work?

- Constructs a CTMC model from a description:
  - Identify (discrete) states and transition rates
    - Individual-based: vectors of (discrete) states of individual molecules
    - Population-based: vectors of numbers of molecules in the respective (discrete) states
  - Assign costs/rewards to states and/or transitions
- Can run simulations directly from syntactic description
  - Obtain individual trajectories, cf Gillespie
- For more powerful analysis, represent the model in a data structure
  - Typically a matrix
  - Can be very large (grows exponentially)
    - 2 molecules of 10 states each can result in $10^2$ states
  - Apply probabilistic model checking (also called verification)

# Simulation vs Verification

- Probabilistic verification exact and detailed
  - exhaustive traversal of the state space
  - can definitively establish causal relationships
  - compute for range of parameters: quantitative trends
  - able to identify best/worst case scenarios
  - but suffers from state explosion problems
- Must consider all possible executions – often not feasible mechanically!
  - [NB a challenging problem in computer science…]
- Simulation approximate
  - but OK for averages over large numbers of runs
  - generally greater scalability

# Continuous-time Markov chains

- Continuous-time Markov chains (CTMCs)
  - labelled transition systems augmented with rates
  - discrete states, continuous time-steps
  - delays exponentially distributed

- Formally, a CTMC C is a tuple $(S, s_{init}, \mathbf{R}, L)$ where:
  - S is a finite set of states ("state space")
  - $s_{init} \in S$ is the initial state
  - $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ is the transition rate matrix
  - $L : S \to 2^{AP}$ is a labelling with atomic propositions

- Transition rates
  - transition between s and s' when $\mathbf{R}(s,s') > 0$
  - probability triggered before t time units $1 - e^{-\mathbf{R}(s,s') \cdot t}$

# Alternative CTMC representations

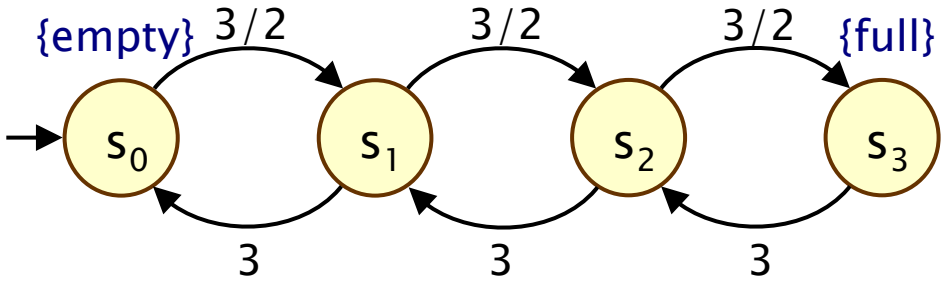
- Alternative definition:
  - a family of random variables { X(t) | t $\in \mathbb{R}_{\geq 0}$ }
  - X(t) are observation made at time instant t
  - i.e. X(t) is the state of the system at time instant t

- Infinitesimal generator matrix

$$Q(s,s') = \begin{cases} -R(s,s') & s \neq s' \\ \sum_{s \neq s'} R(s,s') & \text{otherwise} \end{cases}$$

# Simple CTMC example

- Modelling a queue of jobs
  - jobs arrive with rate 3/2, are served with rate 3
  - maximum size of the queue is 3




- Infinite path ω is a sequence $s_0t_0s_1t_1s_2t_2\ldots$ such that
  - $R(s_i,s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$ for all $i \in \mathbb{N}$
  - paths denoted ω, ω(i) is $i^{th}$ state, ω@t state at time t
- Define probability space via cylinder construction
  - Sample space Ω = Path(s) (infinite and finite paths)
  - Events, least σ-algebra on Path(s)
  - Probability measure $Pr_s$ extends uniquely from probability defined over cylinders

# Embedded DTMC

- Can determine the probability of each transition occurring
  - independent of the time at which it occurs
- Embedded DTMC: $emb(C)=(S,s_{init},\mathbf{P}^{emb(C)},L)$
  - state space, initial state and labelling as for the CTMC
  - let exit rate be $E(s) = \sum_{s' \in S} R(s,s')$
  - for any $s,s' \in S$

$$\mathbf{P}^{emb(C)}(s,s') = \begin{cases} R(s,s')/E(s) & \text{if } E(s) > 0 \\ 1 & \text{if } E(s) = 0 \text{ and } s = s' \\ 0 & \text{otherwise} \end{cases}$$

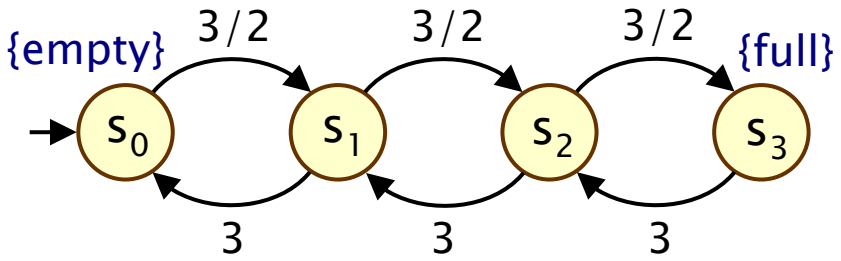
- Alternative convenient characterisation of the behaviour:
  - remain in s for delay exponentially distributed with rate E(s)
  - probability next state is s' is given by $\mathbf{P}^{emb(C)}(s,s')$

# Simple CTMC example

$C = ( S, s_{init}, R, L )$

$S = \{s_0, s_1, s_2, s_3\}$

$s_{init} = s_0$




$AP = \{empty, full\}$

$L(s_0)=\{empty\}$ $L(s_1)=L(s_2)=\varnothing$ and $L(s_3)=\{full\}$

$$R = \begin{bmatrix} 0 & 3/2 & 0 & 0 \\ 3 & 0 & 3/2 & 0 \\ 0 & 3 & 0 & 3/2 \\ 0 & 0 & 3 & 0 \end{bmatrix}$$

transition rate matrix

$$Q = \begin{bmatrix} -3/2 & 3/2 & 0 & 0 \\ 3 & -9/2 & 3/2 & 0 \\ 0 & 3 & -9/2 & 3/2 \\ 0 & 0 & 3 & -3 \end{bmatrix}$$

infinitesimal generator matrix

# Transient and steady-state behaviour

- Transient behaviour
  - state of the model at a particular time instant
  - $\underline{\pi}_{s,t}(s')$ is probability of, having started in state s, being in state s' at time t
  - $\underline{\pi}_{s,t}(s') = \Pr_s\{ \omega \in \text{Path}(s) \mid \omega@t=s' \}$
- Steady-state behaviour
  - state of the model in the long-run
  - $\underline{\pi}_{s}(s')$ is probability of, having started in state s, being in state s' in the long run
  - $\underline{\pi}_{s}(s') = \lim_{t\to\infty} \underline{\pi}_{s,t}(s')$
  - the percentage of time, in long run, spent in each state
- Can compute these numerically, from rates matrix **R**
  - e.g. embedded/uniformised DTMC

# Computing transient probabilities

- $\Pi_t$ – matrix of transient probabilities
  - $\Pi_t(s,s') = \underline{\pi}_{s,t}(s')$
- $\Pi_t$ solution of the differential equation: $\Pi_t' = \Pi_t \cdot Q$
  - Q infinitesimal generator matrix
- Can be expressed as a matrix exponential and therefore evaluated as a power series

$$\Pi_t = e^{Q \cdot t} = \sum_{i=0}^{\infty} (Q \cdot t)^i / i!$$

  - computation potentially unstable
  - probabilities instead computed using the uniformised DTMC (details omitted)

# Continuous Stochastic Logic (CSL)

- Temporal logic for describing properties of CTMCs
  - CSL = Continuous Stochastic Logic
  - extension of (non-probabilistic) temporal logic CTL
- Key additions:
  - probabilistic operator P
  - steady state operator S
- Example: down → $P_{>0.75}$ [ ¬down $U^{[1,2]}$ up ]
  - when the signal goes down, the probability of it going up between 1 and 2 minutes without further decrease is greater than 0.75
- Example: $S_{<0.1}$[a]
  - in the long run, the chance that molecule a is present is less than 0.1

# CSL syntax

- CSL syntax:
  - $\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\sim p}[\psi] \mid S_{\sim p}[\phi]$ (state formulas)
  - $\psi ::= \phi\ U^{I}\ \phi$ (path formulas)
  - where a is an atomic proposition, I interval of $\mathbb{R}_{\geq 0}$ and $p \in [0,1]$, $\sim \in \{<,>,\leq,\geq\}$

$\psi$ is true with probability ~p

"time bounded until"

in the "long run" $\phi$ is true with probability ~p

- Meaning of $\phi\ U^{I}\ \phi$ is standard, i.e. $\omega \models \phi_1\ U^{I}\ \phi_2$
  - there exists a time instant in the interval I where $\phi_2$ is true and $\phi_1$ is true at all preceding time instants

# CSL semantics for CTMCs

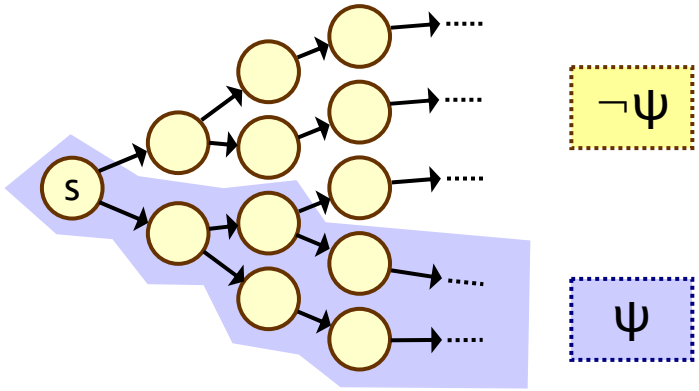
- CSL formulas interpreted over states of a CTMC
  - $s \models \phi$ denotes $\phi$ is "true in state s" or "satisfied in state s"
- Semantics of state formulas standard except for:
  - for a state s of the CTMC $(S, s_{init}, \mathbf{R}, L)$:
  - $s \models P_{\sim p}[\psi] \quad \Leftrightarrow \quad Prob(s, \psi) \sim p$
  - $s \models S_{\sim p}[\phi] \quad \Leftrightarrow \quad \Sigma_{s' \models \phi} \underline{\pi}_s(s') \sim p$

Probability of, starting in state s, satisfying the path formula $\psi$

Probability of, starting in state s, being in state s' in the long run

# CSL semantics

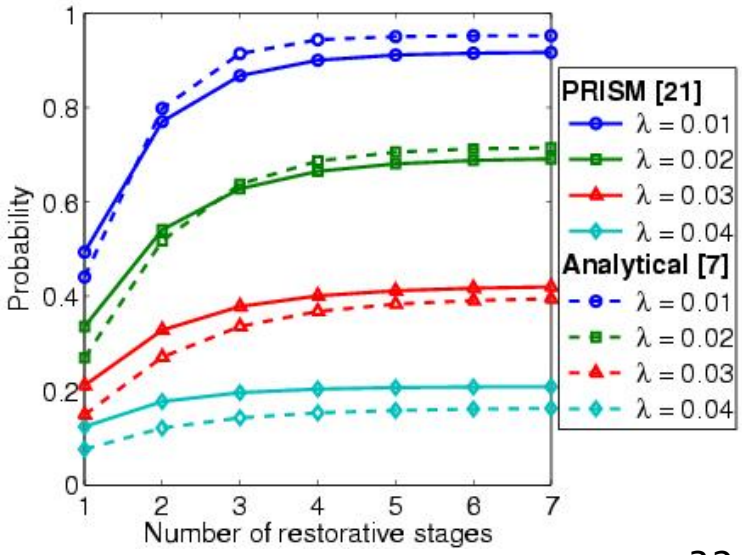
- Semantics of the probabilistic operator P
  - informal definition:
  - $s \models P_{\sim p}[\ \psi\ ]$ means that "the probability, from state s, that $\psi$ is true for an outgoing path satisfies $\sim p$"
  - formally: $s \models P_{\sim p}[\psi] \Leftrightarrow Prob(s, \psi) \sim p$
  - where: $Prob(s, \psi) = Pr_s \{ \omega \in Path(s) \mid \omega \models \psi \}$




$Prob(s, \psi) \sim p$ ?

# Quantitative properties

- Consider a CSL formula $P_{\sim p}$ [ ψ ]
  - if the probability is unknown, how to choose the bound p?
- When the outermost operator of a PTCL formula is P
  - we allow the form $P_{=?}$ [ ψ ]
  - "what is the probability that path formula ψ is true?"
- Model checking is no harder: compute the values anyway
- Useful to spot patterns, trends
- Example (DTMC)
  - P=? [ F err/total>0.1 ]
  - "what is the probability that 10% of the NAND gate outputs are erroneous?"

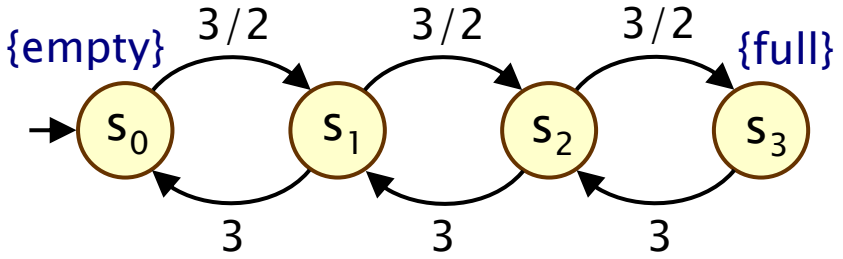


# Reward structures – Example

- Example: "number of requests served"

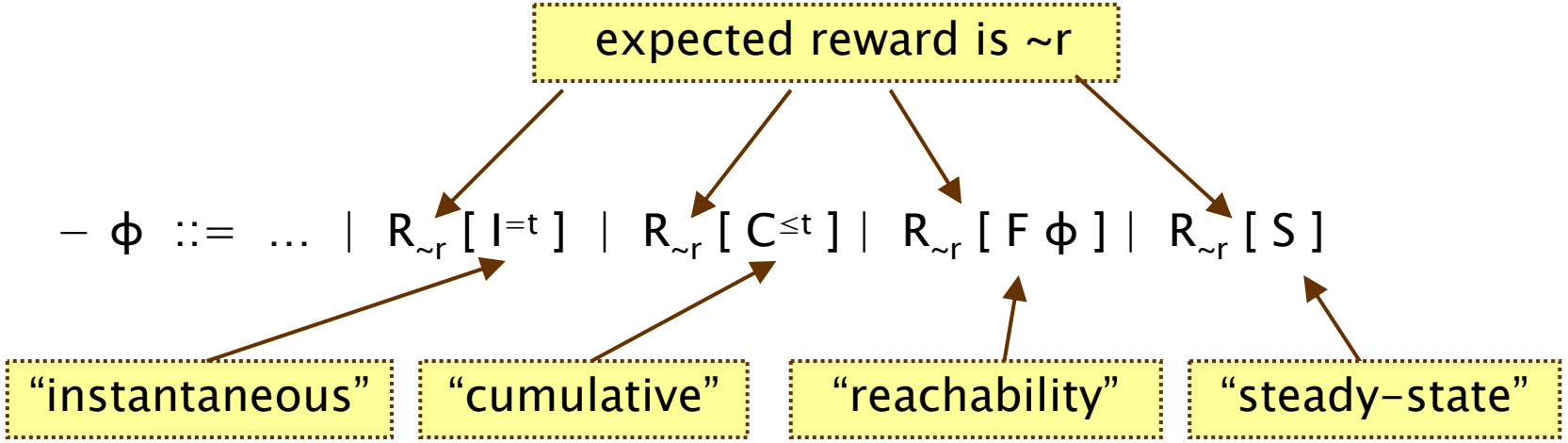
$$\rho = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \iota = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Example: "size of message queue"
  - $\underline{\rho}(s_i)=i$ and $\iota(s_i,s_j)=0$ for all states $s_i$ and $s_j$



# CSL and rewards

- Extend CSL to incorporate reward-based properties
  - add R operator, consider in quantitative form also




  - where r,t $\in \mathbb{R}_{\geq 0}$, ~ $\in \{<,>,\leq,\geq\}$

- $R_{\sim r}$ [ · ] means “the expected value of · satisfies ~r”

# Types of reward formulas

- Instantaneous: $R_{\sim r}$ [ $I^{=t}$ ]
  - the expected value of the reward at time-instant t is ~r
  - "the expected queue size after 6.7 seconds is at most 2"
- Cumulative: $R_{\sim r}$ [ $C^{\leq t}$ ]
  - the expected reward cumulated up to time-instant t is ~r
  - "the expected requests served within the first 4.5 seconds of operation is less than 10"
- Reachability: $R_{\sim r}$ [ F $\phi$ ]
  - the expected reward cumulated before reaching $\phi$ is ~r
  - "the expected requests served before the queue becomes full"
- Steady-state $R_{\sim r}$ [ S ]
  - the long-run average expected reward is ~r
  - "expected long-run queue size is at least 1.2"

# Reward formula semantics

- Formal semantics of the four reward operators:
  - $s \models R_{\sim r}[\, I^{=t} \,] \quad \Leftrightarrow \quad \mathrm{Exp}(s, X_{I=t}) \sim r$
  - $s \models R_{\sim r}[\, C^{\leq t} \,] \quad \Leftrightarrow \quad \mathrm{Exp}(s, X_{C\leq t}) \sim r$
  - $s \models R_{\sim r}[\, F\, \Phi \,] \quad \Leftrightarrow \quad \mathrm{Exp}(s, X_{F\Phi}) \sim r$
  - $s \models R_{\sim r}[\, S \,] \quad \Leftrightarrow \quad \lim_{t\to\infty}( 1/t \cdot \mathrm{Exp}(s, X_{C\leq t}) ) \sim r$
- where:
  - Exp(s, X) denotes the expectation of the random variable
    $X : \mathrm{Path}(s) \to \mathbb{R}_{\geq 0}$ with respect to the probability measure $\Pr_s$
- Details omitted

# Reward formula semantics

- Definition of random variables:
  - path $\omega = s_0 t_0 s_1 t_1 s_2 \ldots$

state of ω at

time spent in state $s_{jt}$ before t time units have elapsed

$$X_{l=k}(\omega) = \underline{\rho}(\omega @ t)$$

time spent in state $s_i$

$$X_{C \le t}(\omega) = \sum_{i=0}^{j_t - 1} \left( t_i \cdot \underline{\rho}(s_i) + \iota(s_i, s_{i+1}) \right) + \left( t - \sum_{i=0}^{j_t - 1} t_i \right) \cdot \underline{\rho}(s_{j_t})$$

$$X_{F\phi}(\omega) = \begin{cases} 0 & \text{if } s_0 \in \mathrm{Sat}(\phi) \\ \infty & \text{if } s_i \notin \mathrm{Sat}(\phi) \text{ for all } i \ge 0 \\ \sum_{i=0}^{k_\phi - 1} t_i \cdot \underline{\rho}(s_i) + \iota(s_i, s_{i+1}) & \text{otherwise} \end{cases}$$

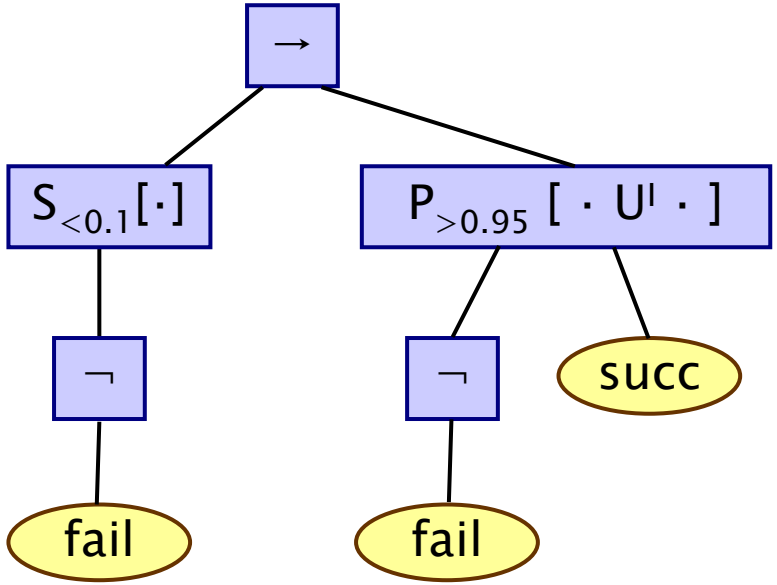
  - where $j_t = \min\{ j \mid \Sigma_{i \le j} t_i \ge t \}$ and $k_\phi = \min\{ i \mid s_i \models \phi \}$

# CSL model checking for CTMCs

- Algorithm for CSL model checking [Baier et al]
  - inputs: CTMC $C=(S,s_{init},\mathbf{R},L)$, CSL formula $\phi$
  - output: Sat($\phi$) = $\{ s \in S \mid s \models \phi \}$, the set of states satisfying $\phi$
- Proceeds by induction on formula, inputs data structure (e.g. matrix, possibly in BDD form) for C
- What does it mean for a CTMC C to satisfy a formula $\phi$?
  - check that $s \models \phi$ for all states $s \in S$, i.e. Sat($\phi$) = S
  - know if $s_{init} \models \phi$, i.e. if $s_{init} \in$ Sat($\phi$)
- Often, focus on quantitative results
  - e.g. compute result of P=? [true $U^{[0,13.5]}$ down ]
  - e.g. compute result of P=? [true $U^{[0,t]}$ down ] for $0 \leq t \leq 100$

# CSL model checking for CTMCs

- Basic algorithm proceeds by induction on parse tree of $\phi$
  - example: $\phi = S_{<0.9}[\neg \text{fail}] \rightarrow P_{>0.95}[\neg \text{fail } U^{I} \text{ succ}]$




- For the non-probabilistic operators:
  - Sat(true) = S
  - Sat(a) = { s ∈ S | a ∈ L(s) }
  - Sat($\neg\phi$) = S \ Sat($\phi$)
  - Sat($\phi_1 \wedge \phi_2$) = Sat($\phi_1$) ∩ Sat($\phi_2$)

# CSL model checking in brief

- The steady-state operator $S_{\sim p}[\phi]$
  - Reduces to solution of a linear equation system
- The time-bounded until $P_{\sim p}[\phi_1\ U^{[0,t]}\ \phi_2]$, remainder omitted
  - Numerical solution (exact, up to rounding error)
    - Reduces to transient analysis using uniformisation
    - Transform CTMC C to C' by removing all outgoing transitions from states satisfying $\phi_1$ or $\phi_2$

$$\text{Prob}(s,\phi_1\ U^{[0,t]}\ \phi_2) = \sum_{s' \in \text{Sat}(\phi_2)} \underline{\pi}^{C'}_{s,t}(s')$$

transient probability: starting in state the probability of being in state s' at time t

  - Sampling-based, statistical hypothesis testing
    - Reduces to simulation and estimation of satisfaction of time bound

# Model checking reward formulas

- Instantaneous: $R_{\sim r}$ [ $I^{=t}$ ]
  - reduces to transient analysis (state of the CTMC at time t)
  - use uniformisation
- Cumulative: $R_{\sim r}$ [ $C^{\leq t}$ ]
  - extends approach for time-bounded until [KNP06]
  - based on uniformisation
- Reachability: $R_{\sim r}$ [ F $\phi$ ]
  - can be computed on the embedded DTMC
  - reduces to solving a system of linear equation
- Steady-state: $R_{\sim r}$ [ S ]
  - similar to steady state formulae $S_{\sim r}$ [ $\phi$ ]
  - graph based analysis (compute BSCCs)
  - solve systems of linear equations (compute steady state probabilities of each BSCC)

# Probabilistic model checking involves…

- Construction of models:
  - DTMCs/CTMCs, MDPs, and PTAs (with digital clocks)
- Implementation of probabilistic model checking algorithms
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
    - qualitative model checking (for 0/1 probability)
  - numerical computation – iterative methods
    - quantitative model checking (plot probability values, expectations, rewards, steady-state, etc, for a range of parameters)
    - exhaustive
  - sampling-based – simulation
    - quantitative model checking as above, based on many simulation runs
    - approximate

# Model checking complexity

- For model checking of a CTMC, complexity is:
  - linear in $|\Phi|$ and polynomial in $|S|$
  - linear in $q \cdot t_{max}$ ($t_{max}$ is maximum finite bound in intervals)
- $P_{\sim p}[\Phi_1\ U^{[0,\infty)}\ \Phi_2]$, $S_{\sim p}[\Phi]$, $R_{\sim r}\ [F\ \Phi]$ and $R_{\sim r}\ [S]$
  - require solution of linear equation system of size $|S|$
  - can be solved with Gaussian elimination: cubic in $|S|$
  - precomputation algorithms (max $|S|$ steps)
- $P_{\sim p}[\Phi_1\ U^{I}\ \Phi_2]$, $R_{\sim r}\ [C^{\leq t}]$ and $R_{\sim r}\ [I^{=t}]$
  - at most two iterative sequences of matrix–vector product
  - operation is quadratic in the size of the matrix, i.e. $|S|$
  - total number of iterations bounded by Fox and Glynn
  - the bound is linear in the size of $q \cdot t$ ($q$ uniformisation rate)

# CSL examples – signalling dynamics

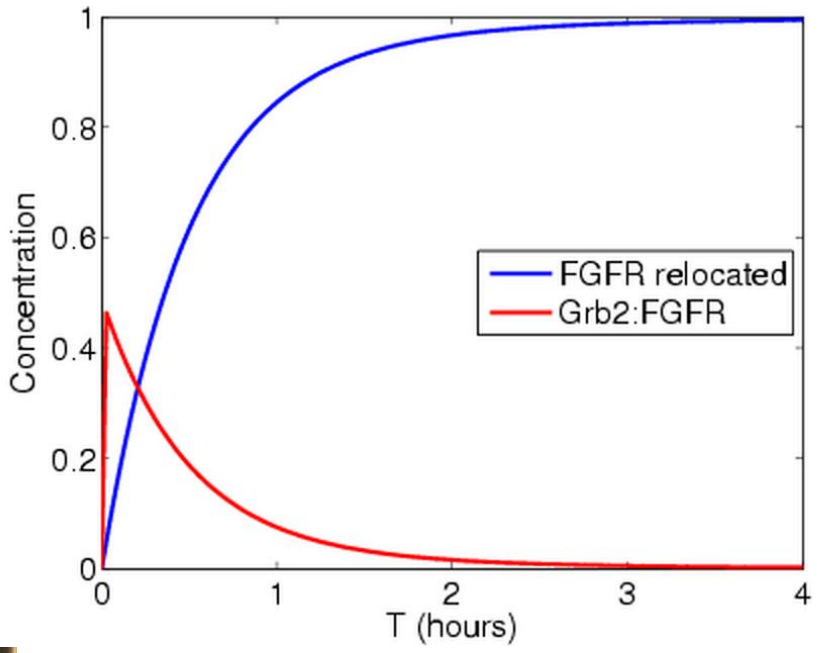
- $P_{=?}$[ true $U^{[T,T]}$ signal ]
  - the probability that signal is present at time T
- $P_{>0.9}$[ ¬A $U^{[0,T]}$ A ]
  - the probability that A causes relocation by time T is greater than 0.9
- $S_{=?}$[ Grb2:FRS2 ]
  - long-run probability that Grb2 bound to FRS2
- $R_{=?}$ [ C≤T ] (assign reward 1 to states where Grb2:FRS2)
  - expected time GRB2 bound to FRS2 within time T
- $R_{=?}$ [ C≤T ] (assign reward 1 to transitions binding Grb2 and FRS2)
  - expected number of times GRB2 & FRS2 bind by T

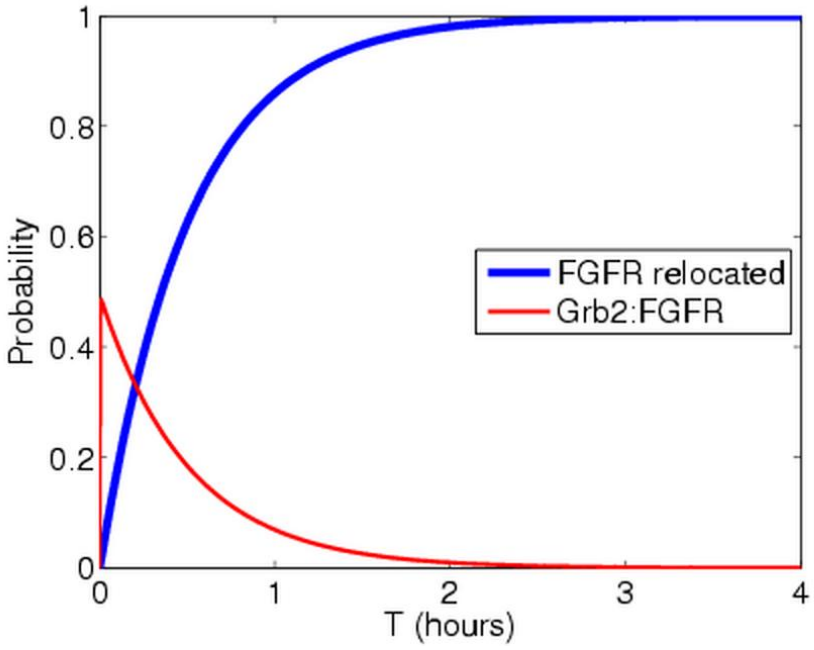
# FGF fragment – Results

Concentration/quantity of two forms of FGFR over time

ODEs

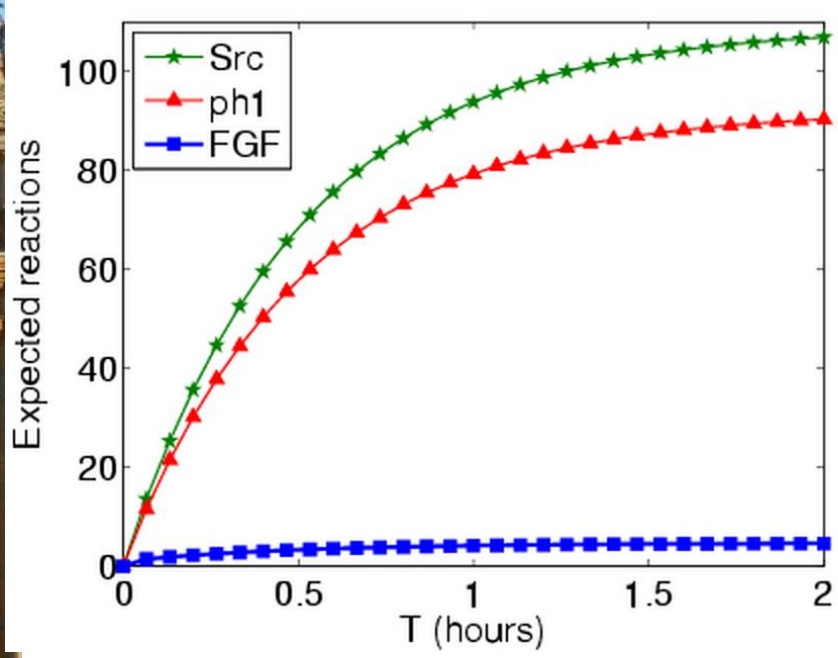
BioPEPA (30 runs)



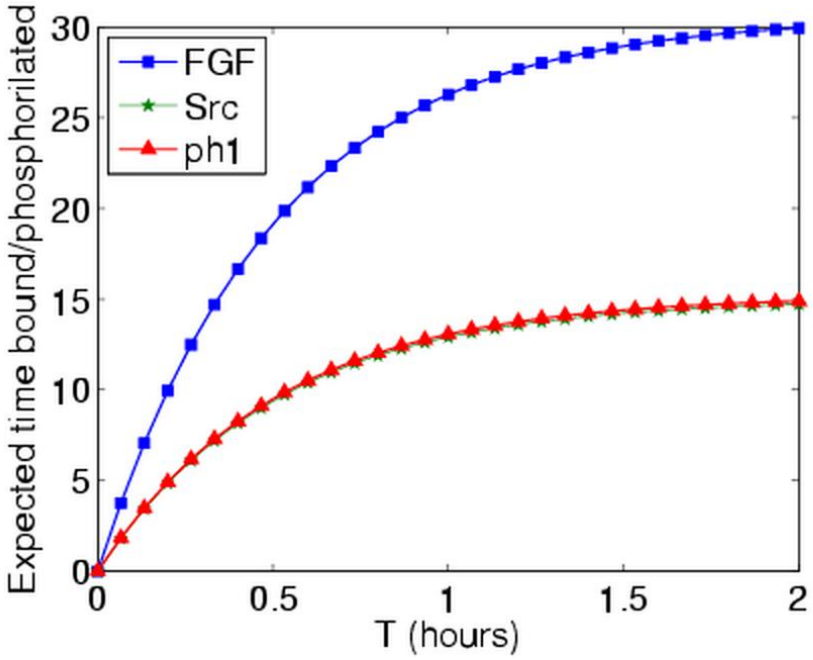



# FGF fragment – PRISM results $R_{=?}[C \leq T]$

Expected number of reactions by time T




Expected time complex spends bound by time T



# A variant of the FGF fragment

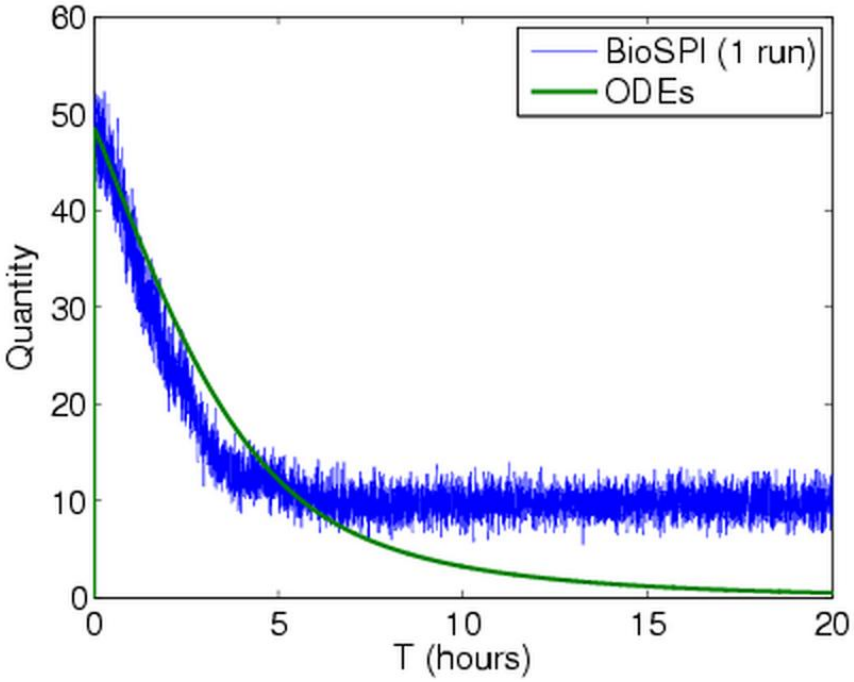
- Src positively regulates FGFR signalling by recruiting non-activated FGFR to the membrane, add reaction:

$$\text{FGFR:Src} \rightarrow \text{FGFR:Src} + \text{FGFR} + \text{Src}$$

Change initial amount of Src from 100 to 10 molecules, and similarly for ODEs

Difference between ODE and BioSPI caused by stochastic approach more accurate when number of molecules small

i.e. Src cannot be totally degraded

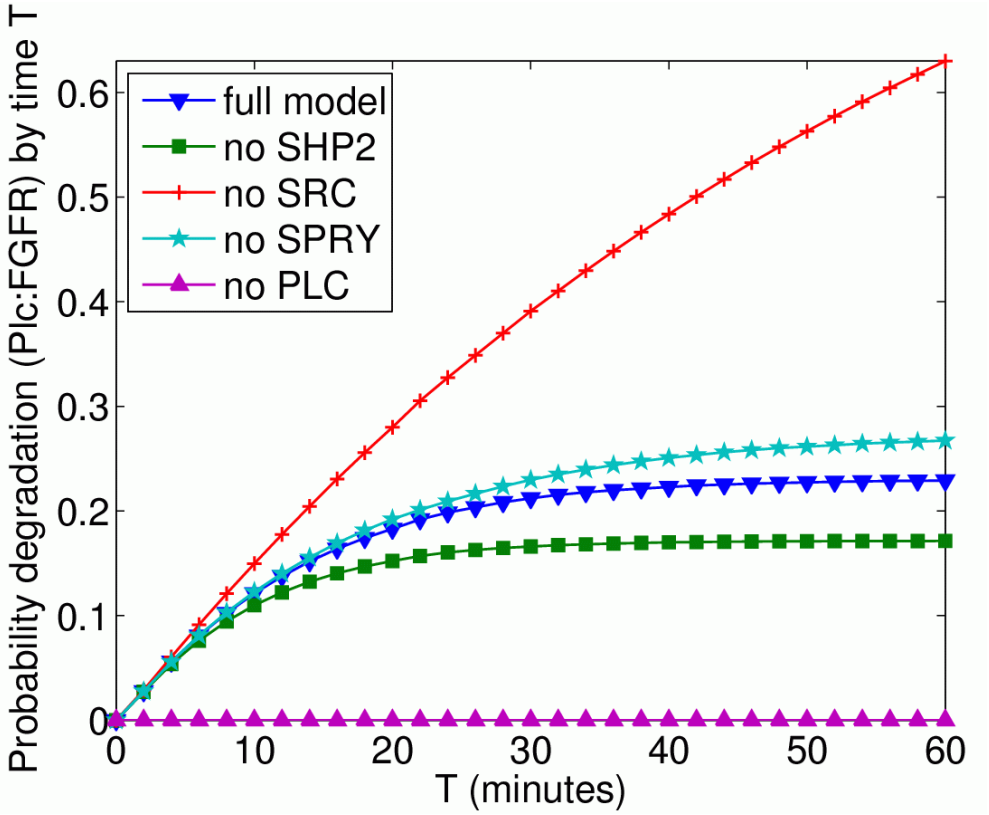


# PRISM model of full FGF pathway

- Biological Model
  - 12 elements
  - 14 phosphorylation sites
  - 14 sets of reaction rules (38 rules)
- PRISM model
  - one element of each type (10 modules and 26 variables)
  - relatively small state space
    (80,616 states and 560,520 transitions)
  - however, highly complex: large number of interactions
  - ODE model > 300 equations, need simplifications
- Predictions
  - known and new, experimentally validated

# FGF pathway – Model checking results

- Probability PLC causes degradation/relocation by T
  - $P_{=?}$ [ $\neg(a_{src} \vee a_{spry} \vee a_{plc})$ $U^{[0,T]}$ $a_{plc}$ ]




no PLC: PLC cannot cause degradation

no SRC: FRS2 not relocated, more chance of degradation by PLC

no SHP2: greater chance SRC bound to FRS2, increasing the possibility of FRS2 causing relocation

# Contrasting the approaches…

- SBML enables
  - Sharing of models
  - Automatic generation of models, both continuous and discrete
- But
  - SBML supports a single discrete state per molecule
  - If parallel molecular state changes, e.g. formation of complexes, observe exponential growth in number of ODEs
  - Discrete approaches more amenable to parallel changes but can result in exponential state explosion
- Also
  - Averages can be misleading if numbers of molecules small
  - Simulation approximate, cannot deal with causality…
- Can we tackle state explosion?

# First success: symmetry reduction

| (N,M) | individual model | | population model | |
|---|---|---|---|---|
| | states | transitions | states | transitions |
| (1,1) | 6 | 9 | 6 | 9 |
| (2,2) | 80 | 344 | 21 | 54 |
| (3,3) | 1,248 | 12,543 | 56 | 189 |
| (4,4) | 21,088 | 415,736 | 126 | 504 |
| (5,5) | 372,356 | 12,850,595 | 252 | 1,134 |
| (6,6) | 6,754,700 | 375,904,704 | 462 | 2,268 |
| (7,7) | 124,755,168 | 10,520,843,095 | 792 | 4,158 |
| (8,8) | 2,333,375,840 | 284,124,205,200 | 1,287 | 7,128 |
| (9,9) | 44,046,493,836 | 7,450,974,265,995 | 2,002 | 11,583 |
| (10,10) | 837,253,442,620 | 190,670,509,035,080 | 3,003 | 18,018 |

- Several orders of magnitude reduction in size of model
- Population-based automatically derived from individual-based model in PRISM
- Still, small numbers of molecules

# Related work

- Predictive analysis of signalling pathways with PRISM, Microsoft Research Cambridge
  - SBML-enabled probabilistic model checking with PRISM
  - FGF as a case study
- Other pathways and systems
  - RKIP inhibited ERK pathway, using PEPA
  - Wnt, multi-scale model coupling Wnt with cell-level decision making, using BioSPI
  - Gene networks, Rho GTP-binding Proteins, using SPiM
- Many more modelling formalisms, simulation and model checking frameworks, etc
  - Pathway Logic, beta-binders, kappa, P-systems, etc

# Future work

- Scalability
- Exploiting structure
  - abstraction/refinement
  - model reductions (symmetry, etc)
  - decomposition…
- Compositional reasoning
- Approximation methods
- Inter-translation between different methodologies
- SBML level 3
- Model extraction from data
- More real pathway case studies

# Acknowledgements


- Joint (inter-disciplinary) work with
  - John Heath (Biosciences, Birmingham)
  - Oksana Tymchyshyn (Computer Science), Gethin Norman, Dave Parker (Computing Laboratory)
  - Eamonn Gaffney (Centre for Mathematical Biology)
- Funding
  - Microsoft Research Cambridge project on Predictive modelling of signalling pathways via probabilistic model checking with PRISM
  - EPSRC, via the Integrative Biology project, BBSRC and CRUK


- More information on PRISM
  - See www.prismmodelchecker.com
  - Case studies, software, statistics, group publications
  - Download, version 3.1 (8000+ downloads)
  - Unix/Linux, Windows, Apple platforms