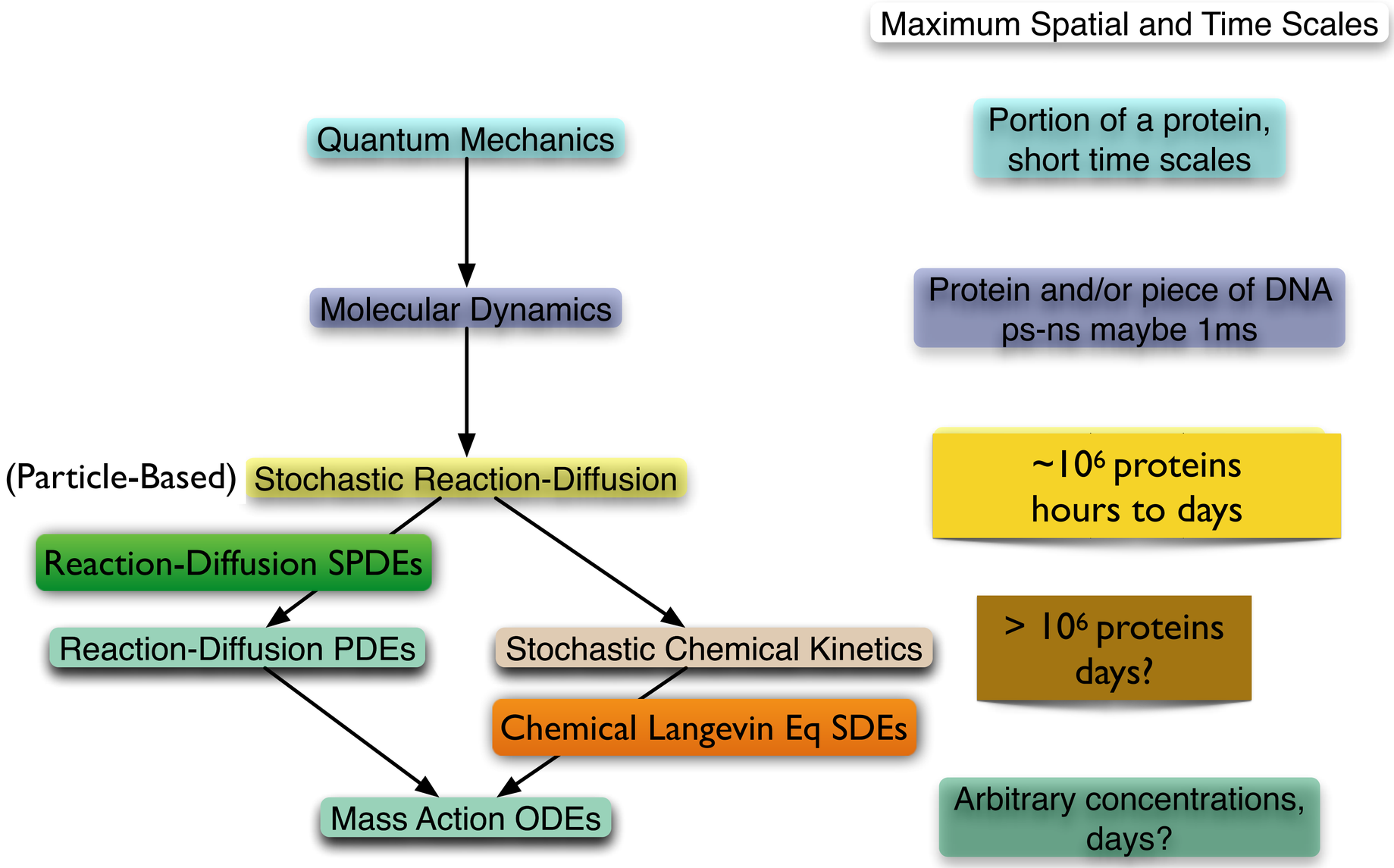
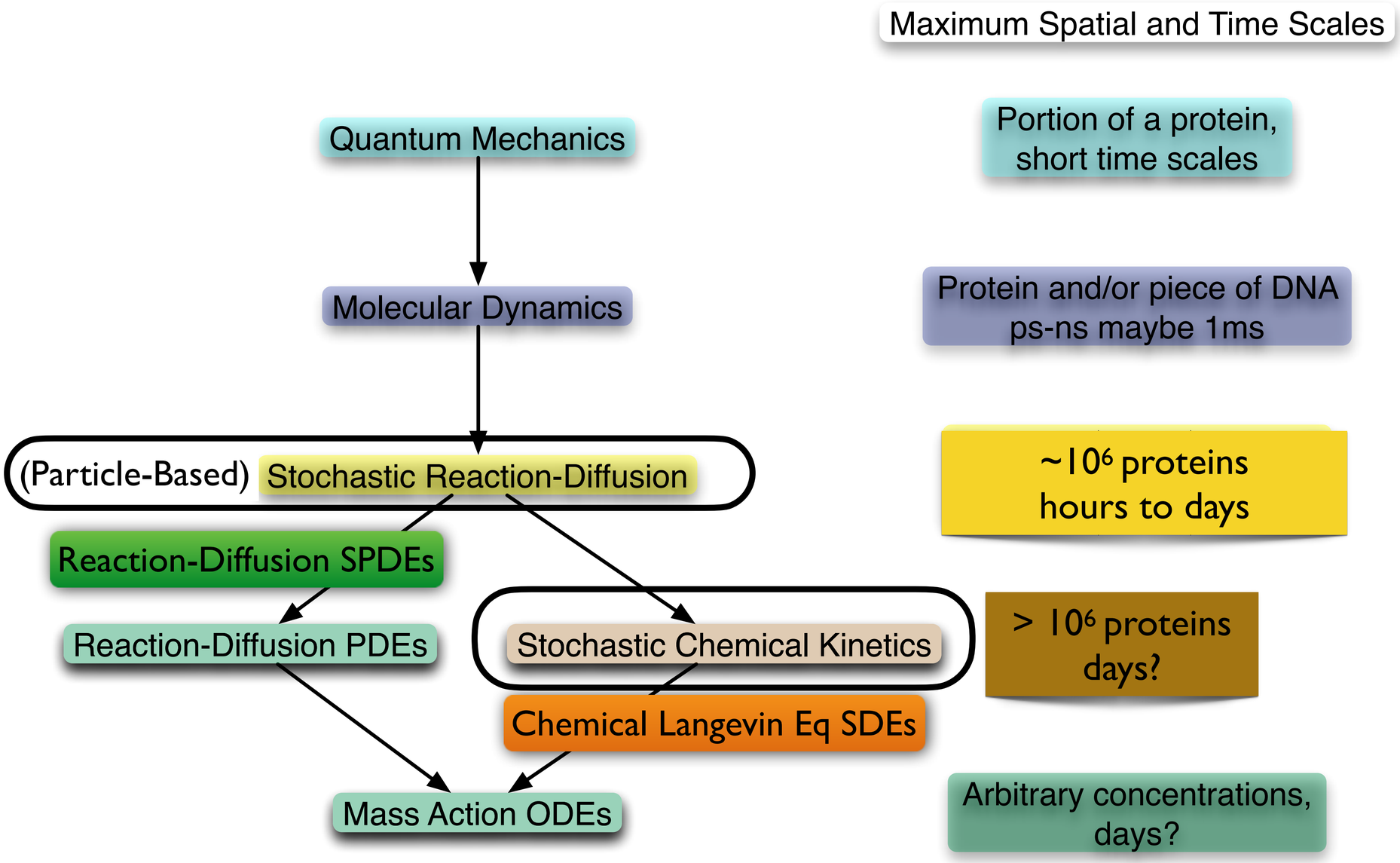
# Computational Methods for Stochastic Chemical Kinetics Models

Samuel Isaacson
Department of Mathematics and Statistics
Boston University
isaacson@math.bu.edu
http://math.bu.edu/people/isaacson/

# How might we model biochemical processes within cells?

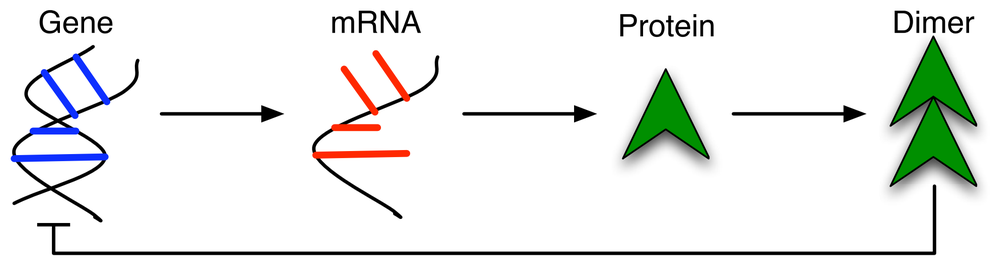


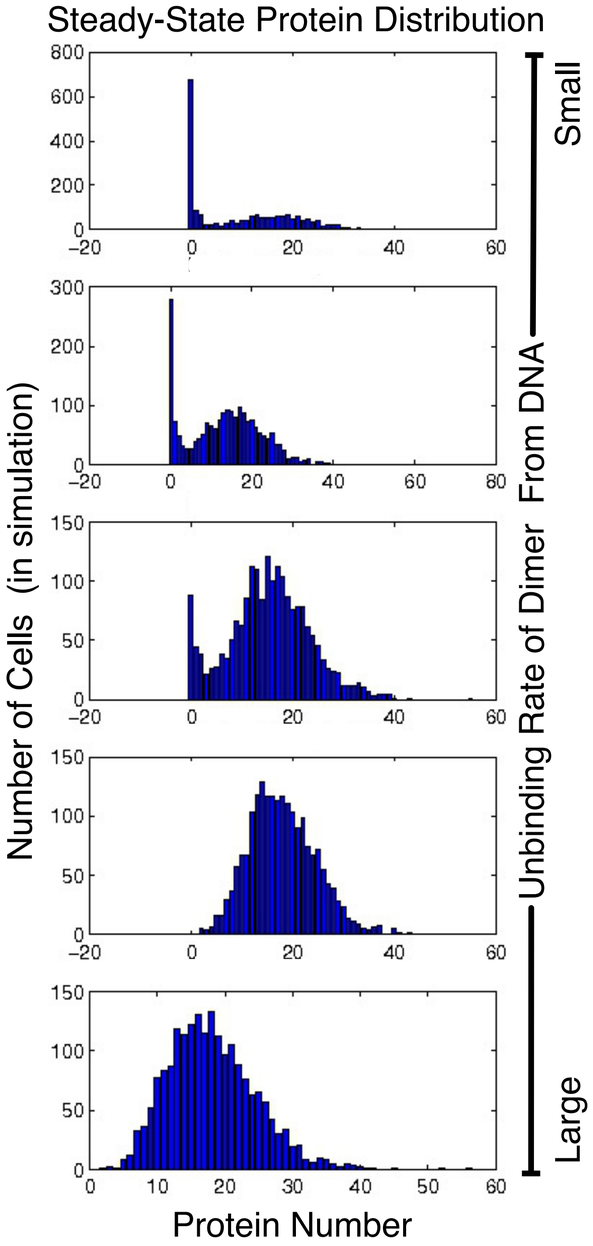
# How might we model biochemical processes within cells?



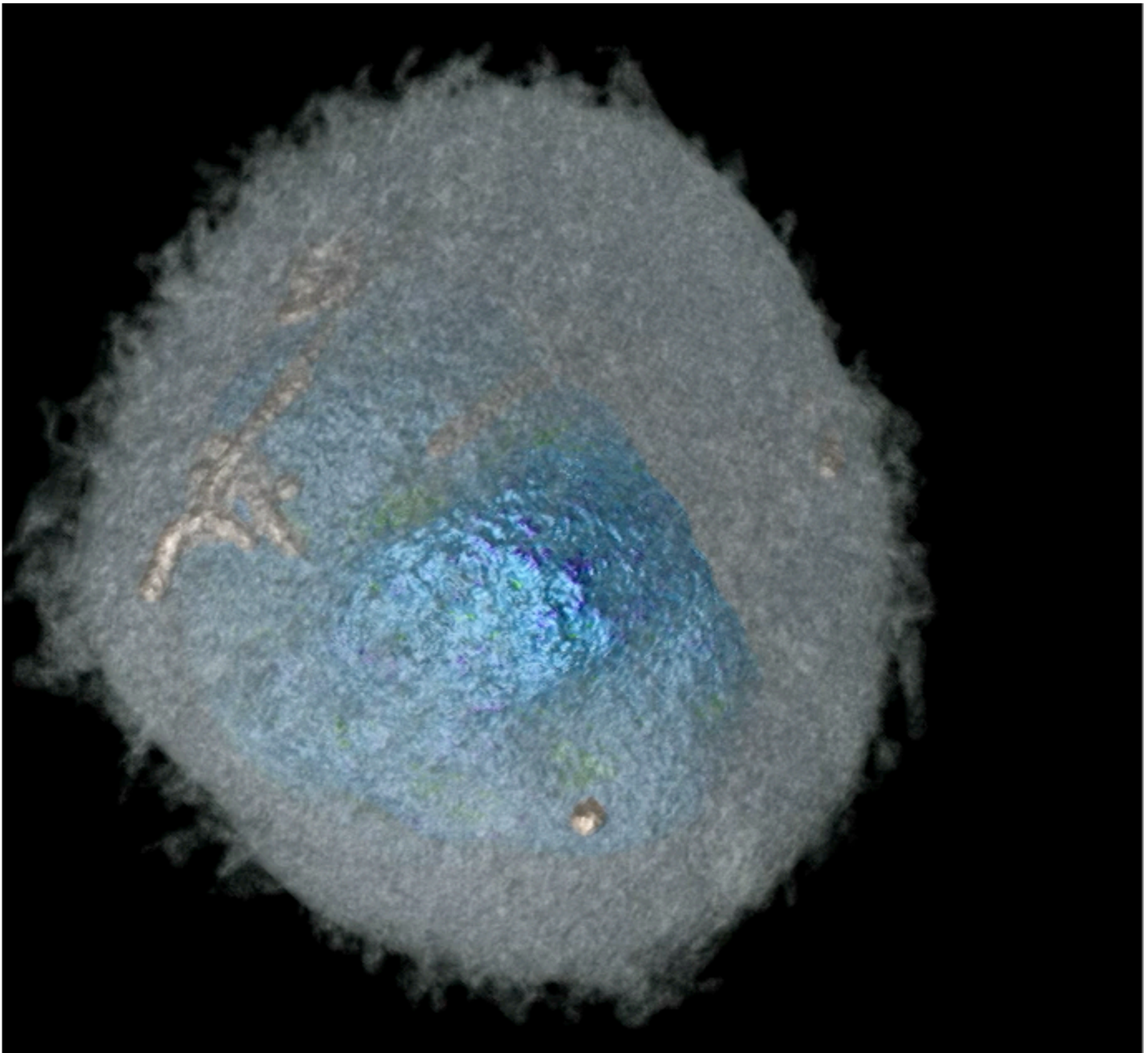
# Why is stochasticity in biochemical reactions important?

- Present in many cellular processes. Evolution.
- Arises from discreteness of chemical species populations.
- For example, gene expression.
  - Seen experimentally and theoretically.
  - See Arkin, Collins, Elowitz...
- Can serve useful biological purpose. See, for example, competence in *Bacillus subtilis*.



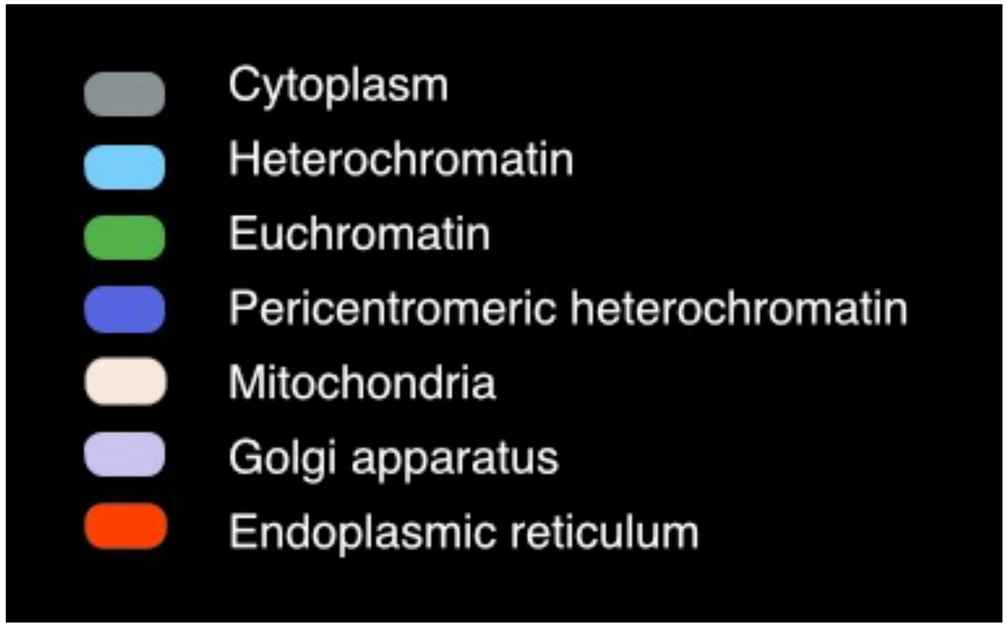



# Are cells well-mixed test-tube like environments?



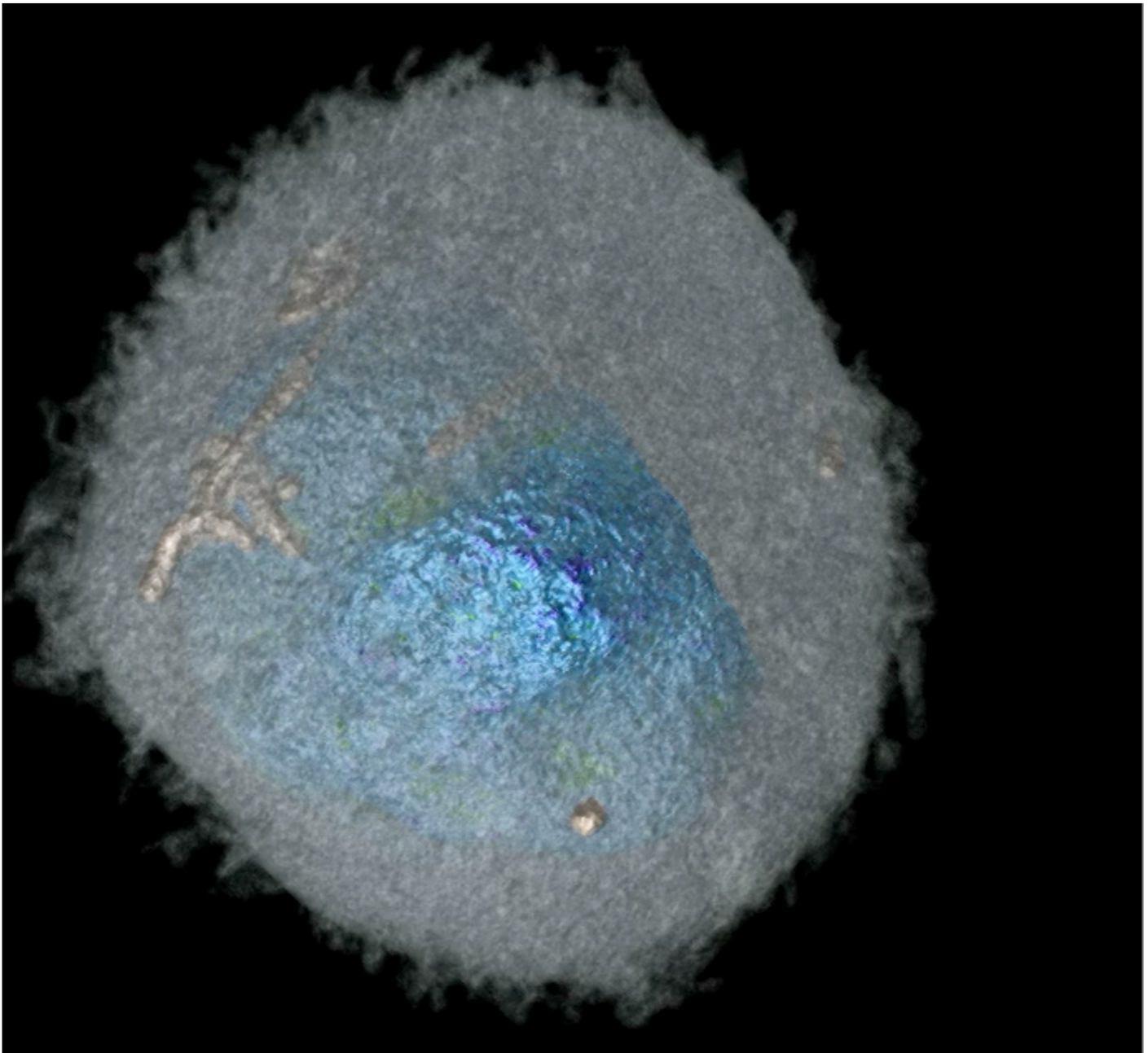
Segmentation of a human B cell reconstructed from soft x-ray tomographic imaging.

Organelles and substructure have been labelled in Amira.



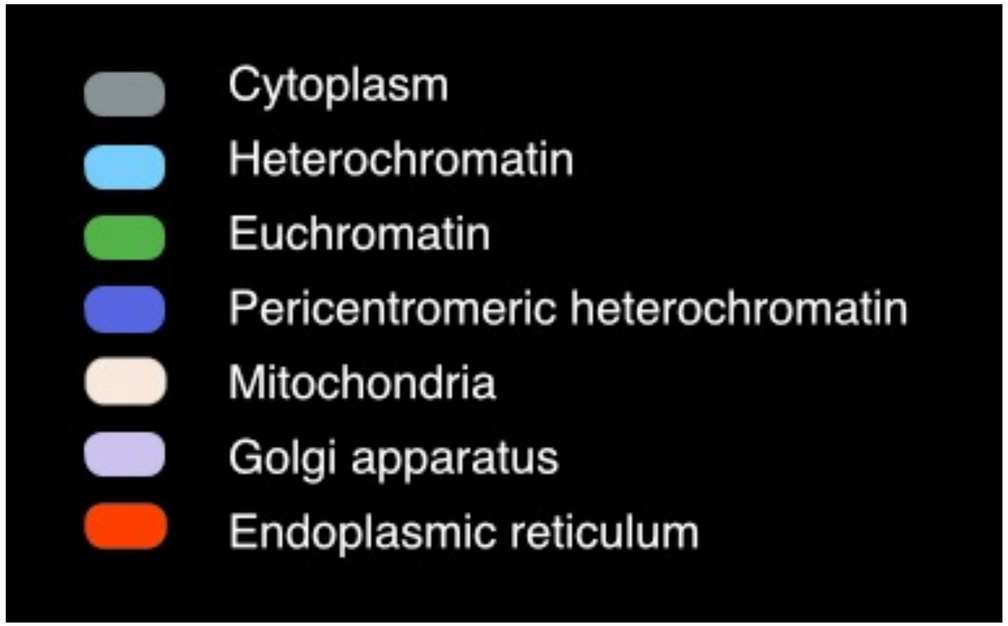

By M. Le Gros
National Center for X-ray Tomography

# Are cells well-mixed test-tube like environments?



Segmentation of a human B cell reconstructed from soft x-ray tomographic imaging.

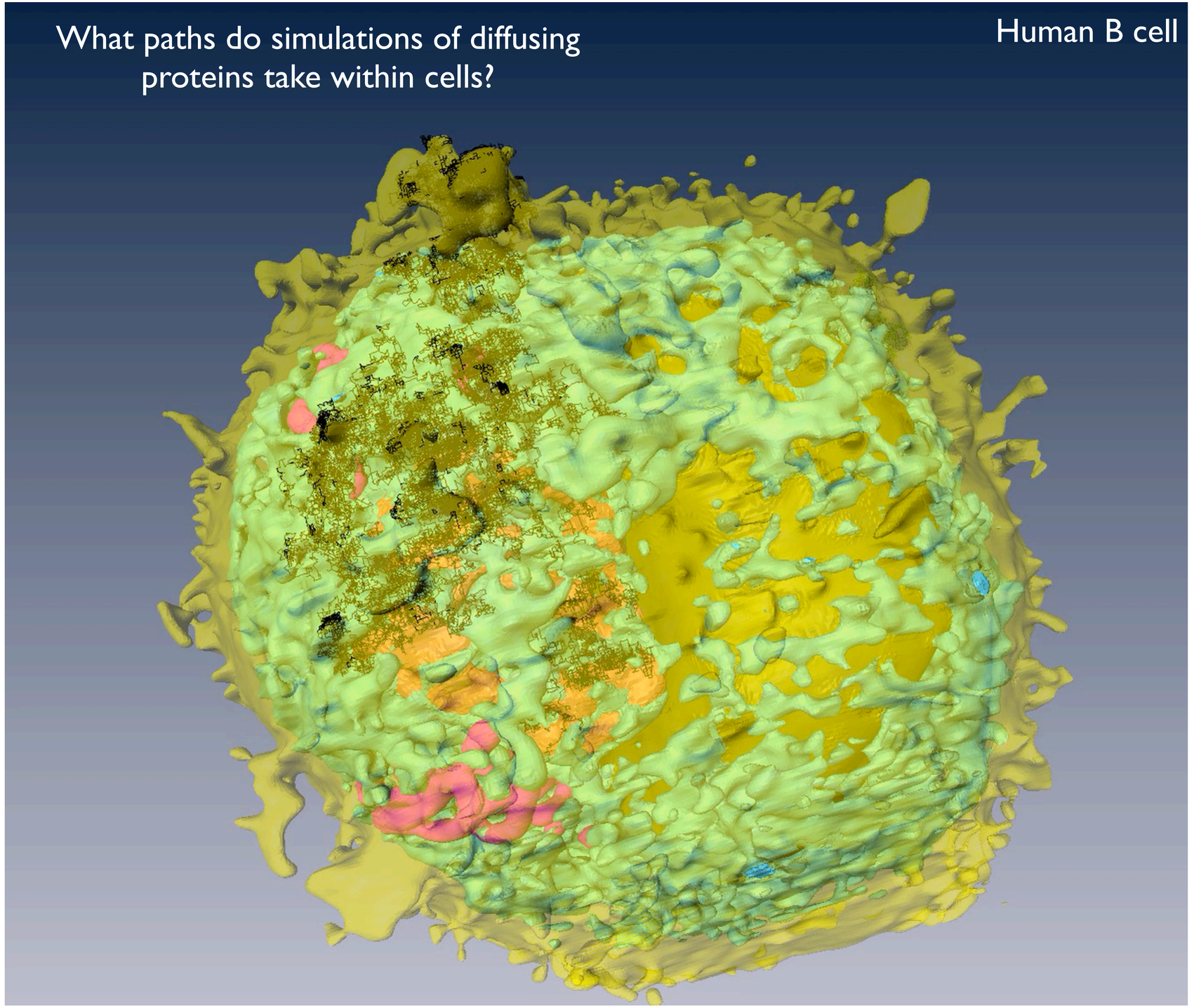
Organelles and substructure have been labelled in Amira.

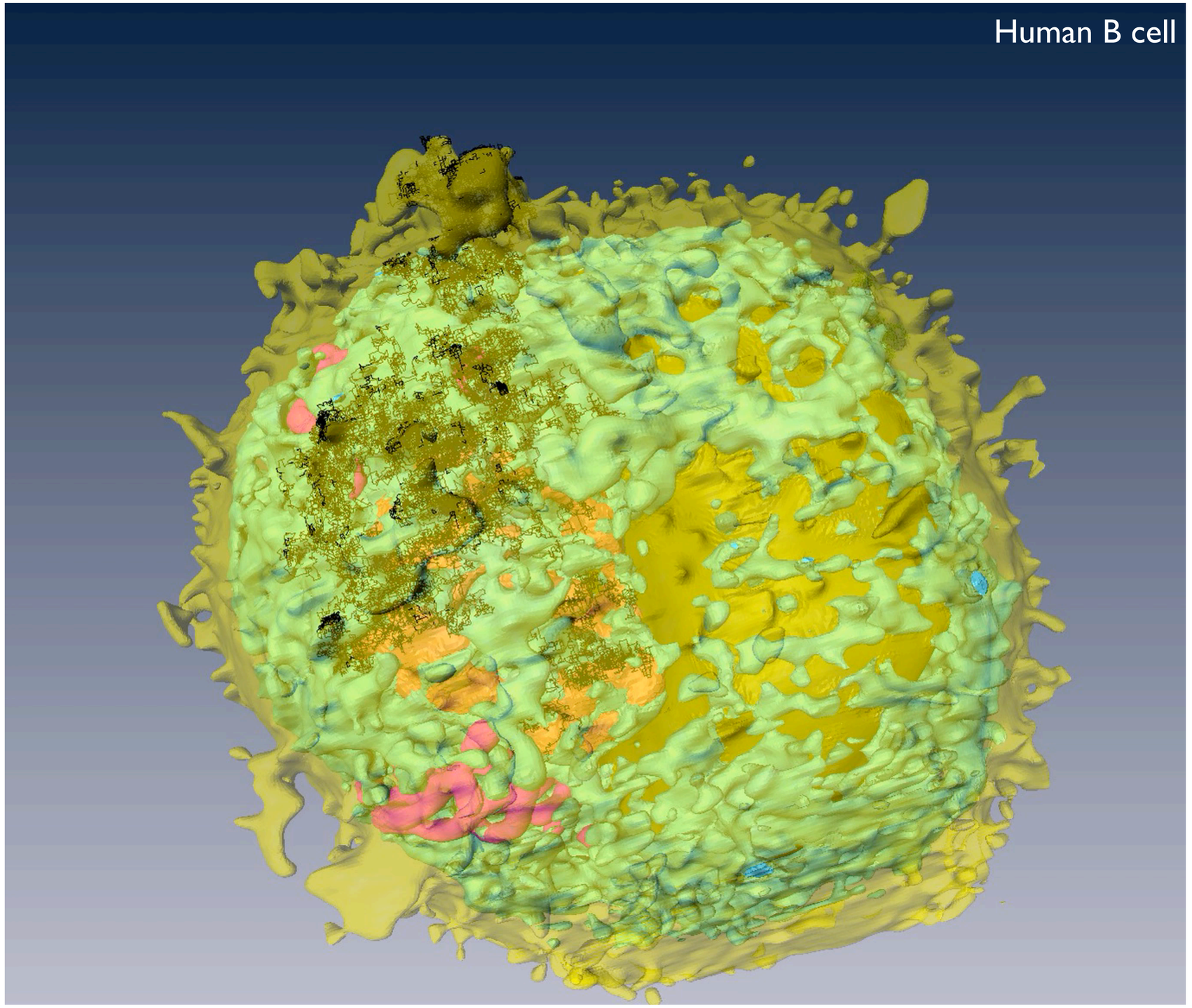



By M. Le Gros
National Center for X-ray Tomography

What paths do simulations of diffusing proteins take within cells?

Human B cell




Human B cell

**So clearly the interior of cells is a very complex, heterogeneous environment.**

**How does subcellular structure influence the dynamics of biochemical processes within cells?**

**So clearly the interior of cells is a very complex, heterogeneous environment.**

**How does subcellular structure influence the dynamics of biochemical processes within cells?**

**To understand this question we need to explicitly model the spatial movement of proteins and mRNAs.**

**So clearly the interior of cells is a very complex, heterogeneous environment.**

**How does subcellular structure influence the dynamics of biochemical processes within cells?**

**To understand this question we need to explicitly model the spatial movement of proteins and mRNAs.**

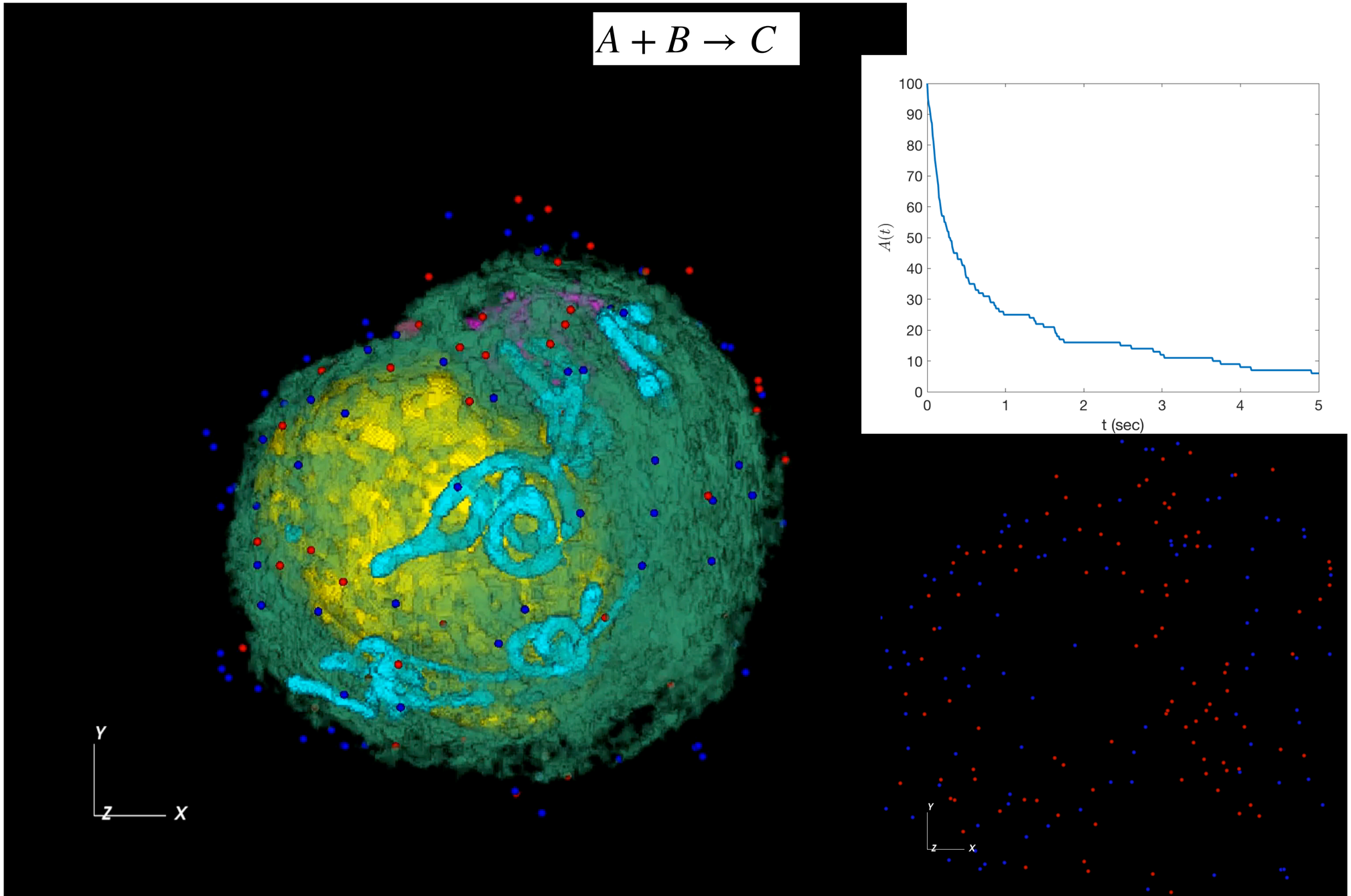
**To account for both spatial transport and stochasticity in the dynamics of proteins and mRNAs we use spatial, stochastic particle models.**

# What are some consequences of noise in both spatial transport and reaction processes?

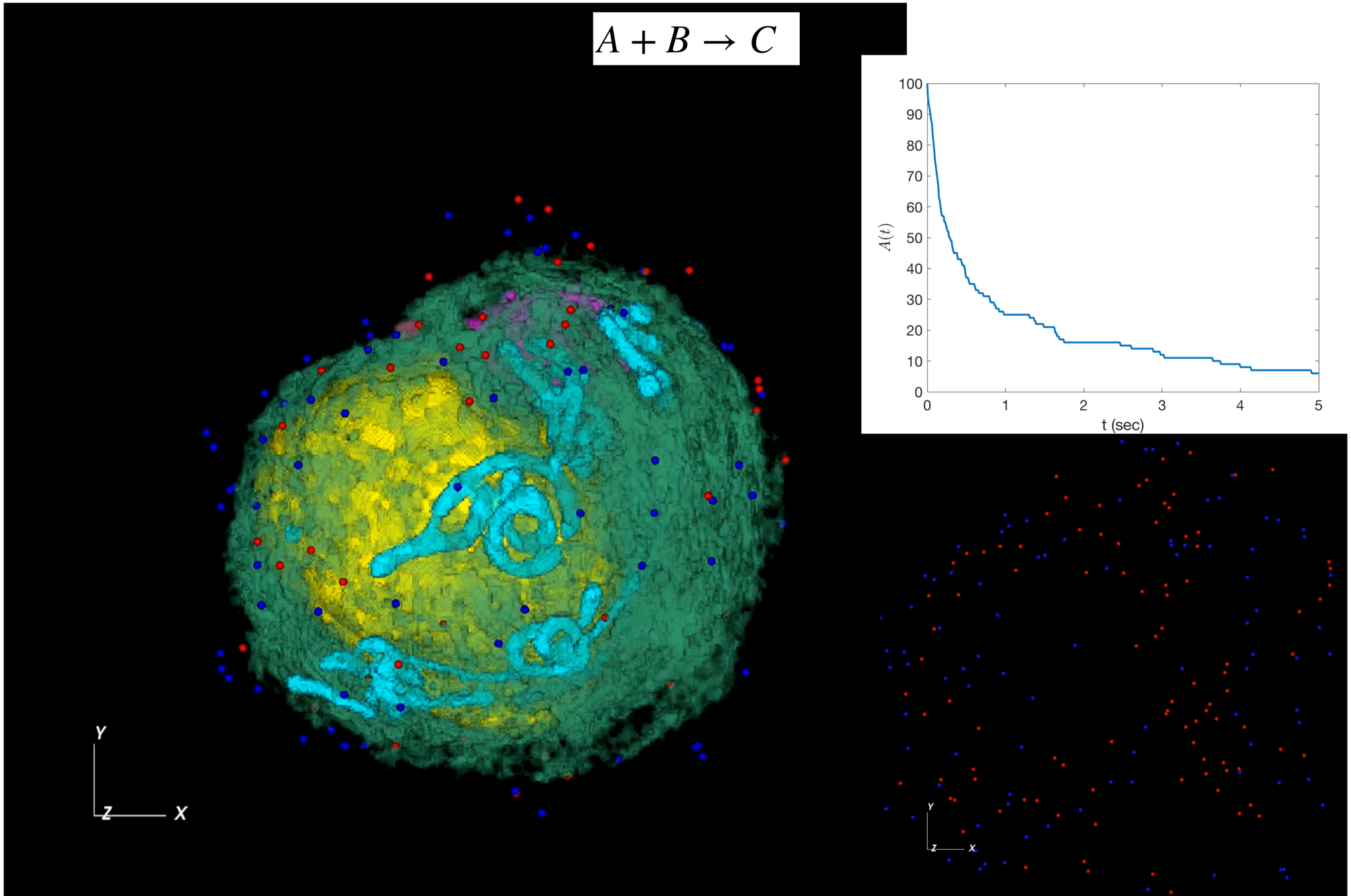
- The time it takes signals to propagate from the cell membrane to nucleus can experience significant variation just due to the cellular substructure of cells[1].
- Even in "empty" domains, spatial heterogeneity from noise in both spatial transport and reaction processes can impact the dynamics of signaling networks.
  - See, for example, MAPK signaling as discussed in Takahashi et al[2].
- Accounting for spatial structure and finite interaction distances can lead to substantively more accurate *predictions* for the ability of antibodies to neutralize antigens[3].


1. Ma, …, and Isaacson, *Strong Intracellular Signal Inactivation Produces Sharper and more Robust Signaling from Cell Membrane to Nucleus*, PLOS Comp. Bio (2020).
2. Takahashi et. al, *Spatio-temporal correlations can drastically change the response of a MAPK pathway,* PNAS (2010).
3. Huhn, …, Isaacson, Dushek, *Analysis of emergent bivalent antibody binding identifies the molecular reach as a critical determinant of SARS-CoV-2 neutralisation potency*, In Review (2024).

# What does a particle-based reaction-diffusion simulation look like? (1)



# What does a particle-based reaction-diffusion simulation look like? (1)



# What will we talk about today?

- Setup of Julia and running the Julia Jupyter notebooks.
- Part I: Simulation of well-mixed stochastic chemical kinetics models.
  - Focus on Monte Carlo methods for generating *statistically exact* sample paths.
  - Called Stochastic Simulation Algorithms (SSAs) or Gillespie Methods.
- Part II: Numerical methods for spatial stochastic chemical kinetics models.
  - Spatial lattice approximation methods (RDME and CRDME).
  - Time-permitting, time-step based Brownian Dynamics methods.

# What are some useful references for this material?

Part I Main Reference:

- Marchetti, Priami, and Thanh, *Simulation Algorithms for Computational Systems Biology*
  - Summarizes all the various well-mixed exact Stochastic Simulation Algorithms (SSAs) I'll discuss, and is only missing a few methods from their discussion.
  - Instead of referencing the papers for the individual methods just see this book and its references.

Part II References:

- Erban and Chapman, *Stochastic Modeling of Reaction-Diffusion Processes*
- Smith and Grima, *Spatial Stochastic Intracellular Kinetics: A Review of Modelling Approaches*, BMB (2019).
- My website for many relevant articles on spatial particle models: http://math.bu.edu/people/isaacson/.

# What do you need to follow along and run the code?

Download and install Julia 10.4 from:
https://julialang.org/downloads/#official_binaries_for_manual_download

**Notebook and script editor / viewer:**

Download and install Visual Studio Code from:
https://code.visualstudio.com/

Install the Julia Visual Studio Code Extension following the instructions at
https://www.julia-vscode.org/docs/dev/gettingstarted/#Installation-and-Configuration-1

Install the Microsoft Jupyter Notebook extensions from
https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter
https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter-renderers

# How do we get started with the notebooks?

1. Download the tutorial notebooks from Dropbox.
2. In the folder where you save them download and unzip:
   https://github.com/isaacsas/ibs_workshop_code.git
3. Start Visual Studio Code and open the well_mixed_models.ipynb notebook.

Let's open the first notebook and make sure everything is working.

# Part I: Simulation of well-mixed stochastic chemical kinetics models

## *(Stochastic Simulation Algorithms (SSAs))*

## What is the simplest model we might consider?

Suppose we have a population of bacterial cells, $N(t)$, with new cells created with probability per time (i.e. rate) $\lambda$.

- We can think of this as the reaction model

$$\emptyset \xrightarrow{\lambda} N$$

- Assuming $N(0) = 0$, in stochastic chemical kinetics $N(t)$ is just a *Poisson counting process*.
- Reaction occurs at random times with rate $\lambda$.
- Note the analogous ODE model is

$$\frac{dN}{dt} = \lambda$$

# What is the simplest model we might consider?

Suppose we have a population of bacterial cells, $N(t)$, with new cells created with probability per time (i.e. rate) $\lambda$.

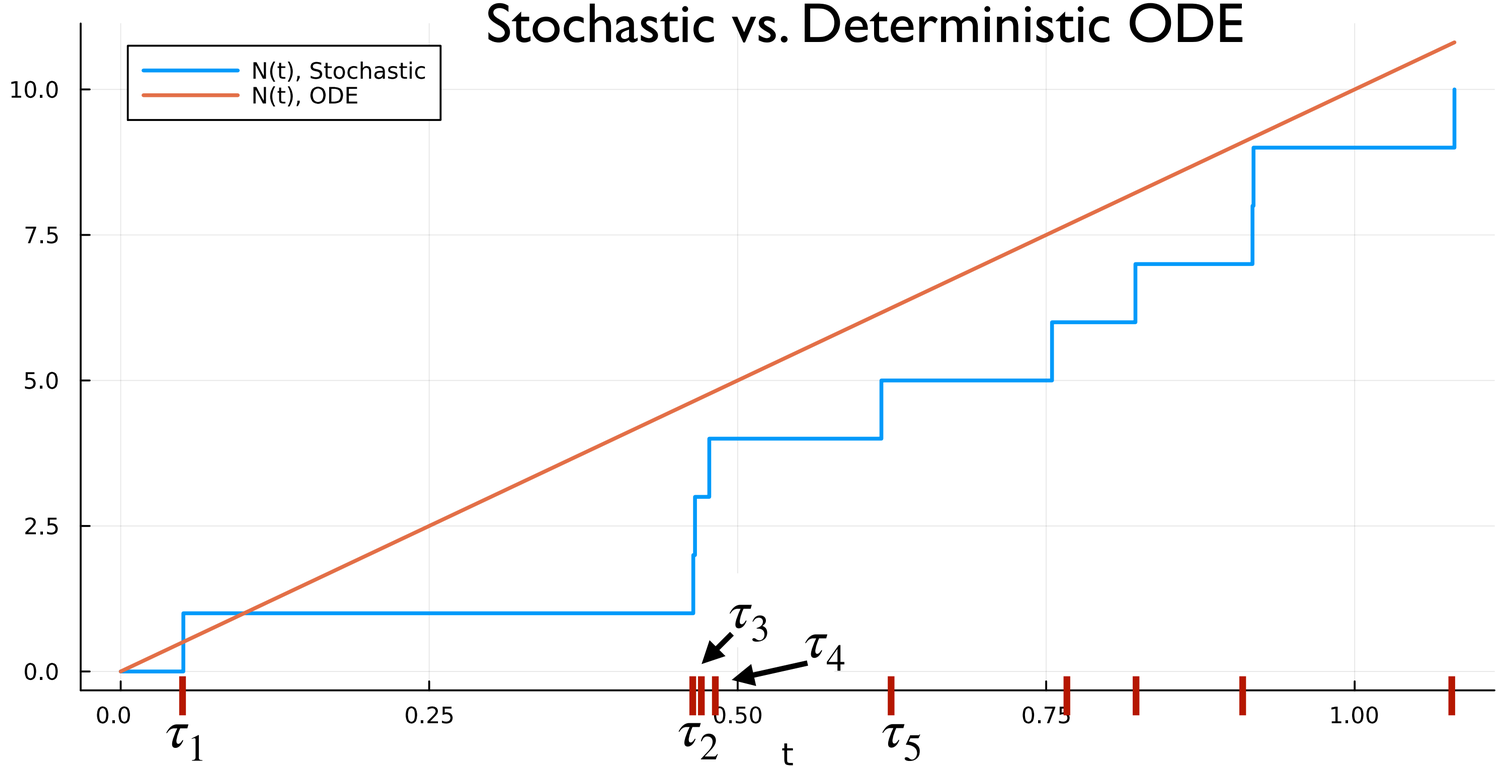
- We can think of this as the reaction model

$$\emptyset \xrightarrow{\lambda} N$$

- Assuming $N(0) = 0$, in stochastic chemical kinetics $N(t)$ is just a *Poisson counting process*.
- Reaction occurs at random times with rate $\lambda$.
- Note the analogous ODE model is

$$\frac{dN}{dt} = \lambda$$

Here $\tau_i$ = time of the $i$th reaction.



# How can we simulate the simplest stochastic chemical kinetics model?

To sample exact realizations of $N(t)$ we can use that the probability density for the next occurrence of the reaction can be calculated analytically[1] and is given by:

$$\rho(\tau; n) = \lambda e^{-\lambda\tau}$$

$=$ The probability density function (PDF) the next reaction occurs at $t + \tau$ given $N(t) = n$.

Units of per time.

1. Marchetti, Priami, and Thanh, *Simulation Algorithms for Computational Systems Biology*

# How can we simulate the simplest stochastic chemical kinetics model?

To sample exact realizations of $N(t)$ we can use that the probability density for the next occurrence of the reaction can be calculated analytically[1] and is given by:

$$\rho(\tau; n) = \lambda e^{-\lambda\tau}$$

$=$ The probability density function (PDF) the next reaction occurs at $t + \tau$ given $N(t) = n$.

Units of per time.

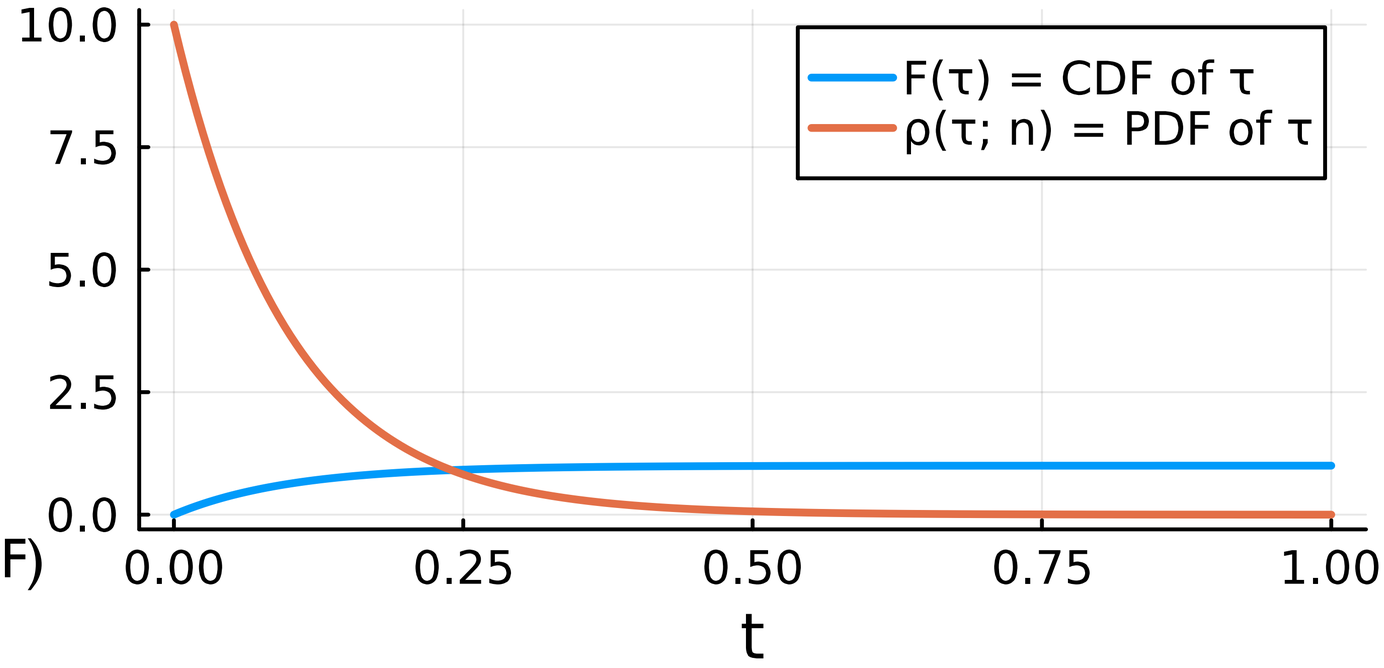
That is, $\tau$ is just an exponential random variable with parameter $\lambda$ . Let

$$F(\tau) = \text{Prob}[\text{next reaction time} < \tau]$$

$$= \int_0^\tau \rho(s; n)\, ds$$

$$= 1 - e^{-\lambda\tau}$$

$=$ Cumulative Distribution Function (CDF)




1. Marchetti, Priami, and Thanh, *Simulation Algorithms for Computatio*

# How can we simulate the simplest stochastic chemical kinetics model? (2)

$\tau$ is just an exponential random variable with parameter $\lambda$.

- There are many ways to generate samples for it.
- "Textbook method", *Inverse Transform Sampling*:
  - Let $u \in \mathcal{U}[0,1)$ denote a standard uniform random variable (`rand()` in Julia).
  - We solve the following for $\tau$:

$$u = F(\tau) = 1 - e^{-\lambda\tau}$$

  - Giving

$$\tau = F^{-1}(u) = -\frac{1}{\lambda}\ln(1-u)$$

# How can we simulate the simplest stochastic chemical kinetics model? (3)

This gives the following simple algorithm to update the current number of cells by calculating the time the next new cell is created:

1. Assuming $N(0) = 0$, set $n = 0$ and $t = 0$.
2. while not done
3. $u = \mathcal{U}([0,1))$ (i.e. call `rand()`)
4. $\tau = -\frac{1}{\lambda}\ln(1-u)$ (the time till the next cell is created)
5. $t = t + \tau$ (the time the next cell is created)
6. $n = n + 1$ (the number of cells at this time)
7. end

Let's look at the "well_mixed_models.ipynb" notebook to see a first performance consideration in writing such a (simple) method in Julia!

# What about models with many chemical reactions?

Let's first introduce some notation to describe a general system of chemical reactions.

Notation for species amounts:

- $N =$ the number of chemical species.
- $X_n(t) =$ the stochastic process for the number of molecules of species $n$ at $t$ ($n \in 1,\ldots,N$).
- $\boldsymbol{X}(t) = (X_1(t), \ldots, X_N(t)) =$ the full state vector for the number of molecules of all each species.
- $\boldsymbol{x} = (x_1, \ldots, x_N) =$ a possible value of $\boldsymbol{X}(t)$, i.e. $\boldsymbol{X}(t) = \boldsymbol{x}$.

## What about models with many chemical reactions?

Let's first introduce some notation to describe a general system of chemical reactions.

Notation for species amounts:

- $N =$ the number of chemical species.
- $X_n(t) =$ the stochastic process for the number of molecules of species $n$ at $t$ ($n \in 1,\ldots,N$).
- $\boldsymbol{X}(t) = (X_1(t), \ldots, X_N(t)) =$ the full state vector for the number of molecules of all each species.
- $\boldsymbol{x} = (x_1, \ldots, x_N) =$ a possible value of $\boldsymbol{X}(t)$, i.e. $\boldsymbol{X}(t) = \boldsymbol{x}$.

Example, consider $A + B \rightarrow C$

- $\boldsymbol{X}(t) = (A(t), B(t), C(t))$.
- $\boldsymbol{x} = (a, b, c)$.
- If at time $t$ there are 10 *A*, 5 *B*, and 7 *C*, we can set $\boldsymbol{x} = (10,5,7)$ and have $\boldsymbol{X}(t) = \boldsymbol{x}$.

## What about models with many chemical reactions? (2)

Notation for reactions:

- $M =$ the number of different reactions that can occur.
- $R_k$ labels the $k$th reaction.
- $a_k(\boldsymbol{x}) =$ the propensity, i.e. probability per time for $R_k$ to occur when $\boldsymbol{X}(t) = \boldsymbol{x}$ (ignoring other reactions).

## What about models with many chemical reactions? (2)

Notation for reactions:

- $M =$ the number of different reactions that can occur.
- $R_k$ labels the $k$th reaction.
- $a_k(\boldsymbol{x}) =$ the propensity, i.e. probability per time for $R_k$ to occur when $\boldsymbol{X}(t) = \boldsymbol{x}$ (ignoring other reactions).
  - For the jump process we are constructing this corresponds to a (state-dependent) transition rate for the $k$th possible jump type.
  - For a mass action reaction this is just the rate for a *minimal* set of substrates multiplied by the number of combinations of substrates, e.g.
  - Let $k$ denote probability per time *one* randomly selected set of substrates will react then:

## What about models with many chemical reactions? (2)

Notation for reactions:

- $M =$ the number of different reactions that can occur.
- $R_k$ labels the $k$th reaction.
- $a_k(\boldsymbol{x}) =$ the propensity, i.e. probability per time for $R_k$ to occur when $\boldsymbol{X}(t) = \boldsymbol{x}$ (ignoring other reactions).
  - For the jump process we are constructing this corresponds to a (state-dependent) transition rate for the $k$th possible jump type.
  - For a mass action reaction this is just the rate for a *minimal* set of substrates multiplied by the number of combinations of substrates, e.g.
  - Let $k$ denote probability per time *one* randomly selected set of substrates will react then:

$A + B \xrightarrow{k} C$ has $a_1(\boldsymbol{x}) = kab$,

$A + A \xrightarrow{k} B$ has $a_1(\boldsymbol{x}) = \frac{k}{2}a(a-1)$,

$A \xrightarrow{k} B$ has $a_1(\boldsymbol{x}) = ka$,

$\emptyset \xrightarrow{k} A$ has $a_1(\boldsymbol{x}) = k$.

# What about models with many chemical reactions? (3)

Notation for reactions continued:

- $\nu_k$ = The change in the state when the $k$th reaction occurs
  - If $\boldsymbol{X}(t) = \boldsymbol{x}$ and reaction $k$ occurs, then $\boldsymbol{x} \to \boldsymbol{x} + \nu_k$.
- $\boldsymbol{\nu} = [\nu_1, \ldots, \nu_M]$ = the stoichiometry matrix with column $k$ given by $\nu_k$.
- *$\boldsymbol{\nu}$ determines how the state changes when a reaction occurs.*

# What about models with many chemical reactions? (4)

Let

$$\rho_k(\tau; \boldsymbol{x}) = \text{ the probability density (per time) the next reaction occurs a time } \tau \text{ in the future and is of type } k \text{ given } \boldsymbol{X}(t) = \boldsymbol{x}.$$

Let $a(\boldsymbol{x}) = \sum_{k=1}^{M} a_k(\boldsymbol{x})$ denote the *total propensity for the system when* $\boldsymbol{X}(t) = \boldsymbol{x}$.

$a(\boldsymbol{x})$ represents the probability per time that some reaction occurs

It can be shown that $\rho_k(\tau; \boldsymbol{x}) = a_k(\boldsymbol{x}) e^{-a(\boldsymbol{x})\tau}$.

# What about models with many chemical reactions? (5)

From this one can derive that

- $\rho(\tau; \boldsymbol{x}) = a(\boldsymbol{x})e^{-a(\boldsymbol{x})\tau} =$ the probability density that the *any* reaction occurs at time $\tau$ given $\boldsymbol{X}(t) = \boldsymbol{x}$
- $P_k = \dfrac{a_k(\boldsymbol{x})}{a(\boldsymbol{x})} =$ the probability the next reaction is of type $k$.

# What about models with many chemical reactions? (5)

Recall for $\emptyset \xrightarrow{\lambda} N$ we had the exponential random variable with density:

$$\rho(\tau; \boldsymbol{x}) = \lambda e^{-\lambda\tau}$$

From this one can derive that

- $\rho(\tau; \boldsymbol{x}) = a(\boldsymbol{x})e^{-a(\boldsymbol{x})\tau} =$ the probability density that the *any* reaction occurs at time $\tau$ given $\boldsymbol{X}(t) = \boldsymbol{x}$
- $P_k = \dfrac{a_k(\boldsymbol{x})}{a(\boldsymbol{x})} =$ the probability the next reaction is of type $k$.

## What about models with many chemical reactions? (5)

Recall for $\emptyset \xrightarrow{\lambda} N$ we had the exponential random variable with density:

$$\rho(\tau; \boldsymbol{x}) = \lambda e^{-\lambda\tau}$$

From this one can derive that

- $\rho(\tau; \boldsymbol{x}) = a(\boldsymbol{x})e^{-a(\boldsymbol{x})\tau} =$ the probability density that the *any* reaction occurs at time $\tau$ given $\boldsymbol{X}(t) = \boldsymbol{x}$
- $P_k = \dfrac{a_k(\boldsymbol{x})}{a(\boldsymbol{x})} =$ the probability the next reaction is of type $k$.

Hence given $\boldsymbol{X}(t) = \boldsymbol{x}$ we can

1. Sample the time until the next reaction, $\tau$, as an exponential random variable with parameter $a(\boldsymbol{x})$, using any of the previously discussed methods.
2. Update $t \to t + \tau$
3. Sample the reaction, $k$, that occurs from the discrete distribution $\{P_1, \ldots, P_M\}$.
4. Update $\boldsymbol{x} \to \boldsymbol{x} + \nu_k$.

## How do we sample $k$, i.e. which reaction occurs next?

To sampled the discrete probability distribution defined by $\{P_k\}_{k=1}^{M}$

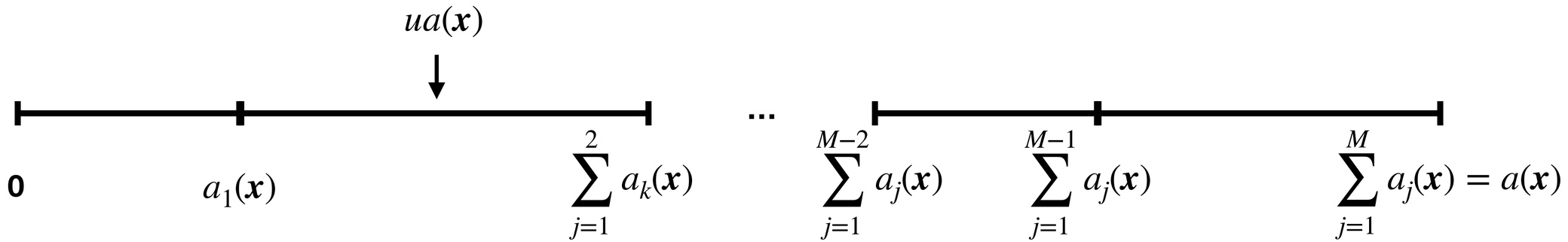
- Sample a uniform random number $u \in \mathcal{U}([0,1))$.
- Select the smallest $k$ such that $\sum_{j=1}^{k} a_j(\boldsymbol{x}) \geq u a(\boldsymbol{x})$.

# How do we sample $k$, i.e. which reaction occurs next?

To sampled the discrete probability distribution defined by $\{P_k\}_{k=1}^{M}$

- Sample a uniform random number $u \in \mathcal{U}([0,1))$.
- Select the smallest $k$ such that $\sum_{j=1}^{k} a_j(\boldsymbol{x}) \geq ua(\boldsymbol{x})$.

Example, below we'd sample $k = 2$:

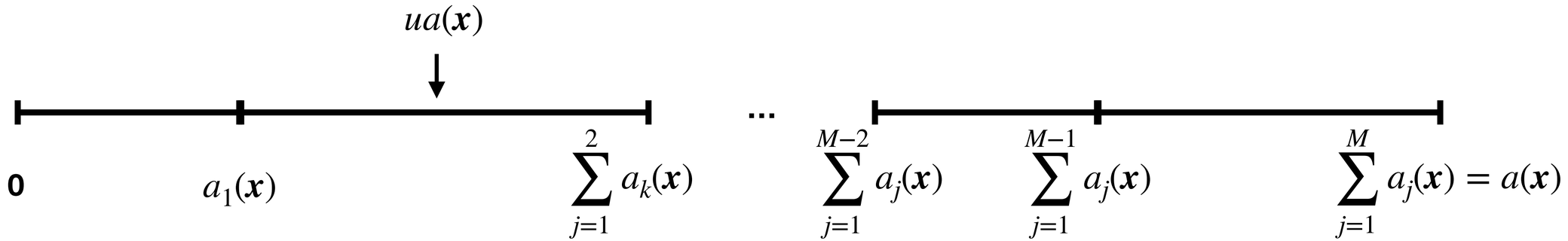


## How do we sample $k$, i.e. which reaction occurs next?

To sampled the discrete probability distribution defined by $\{P_k\}_{k=1}^{M}$

- Sample a uniform random number $u \in \mathcal{U}([0,1))$.
- Select the smallest $k$ such that $\sum_{j=1}^{k} a_j(\boldsymbol{x}) \geq ua(\boldsymbol{x})$.

Example, below we'd sample $k = 2$:




A linear search would take $O(M)$ in the worst case, while a binary search would be $O(\log(M))$.
What search is fastest will depend on how big $M$ is.
(Note, calculating the cumulative sums before searching is $O(M)$ work!)

# What is the Direct Method SSA?

It is the simplest method for sampling the next reaction time and next reaction.

**Algorithm 2** Direct Method (DM)

**Input:** a biochemical reaction network of $M$ reactions in which each reaction $R_j$, $j = 1, \ldots, M$, is accompanied with the state change vector $\mathbf{v}_j$ and the propensity $a_j$, the initial state $\mathbf{x}_0$ at time 0 and the simulation ending time $T_{max}$

**Output:** a trajectory $X(t)$, $0 \leq t \leq T_{max}$, of the biochemical reaction network

1: initialize time $t = 0$ and state $X = \mathbf{x}_0$
2: **while** $(t < T_{max})$ **do**
3: set $a_0 = 0$
4: **for all** (reaction $R_j$) **do**
5: compute $a_j$
6: update $a_0 = a_0 + a_j$
7: **end for**
8: generate two random numbers $r_1, r_2 \sim \mathrm{U}(0,1)$ (see Appendix B.1)
9: select $R_\mu$ with the smallest index $\mu$ such that $\sum_{j=1}^{\mu} a_j \geq r_1 a_0$
10: compute $\tau = 1/a_0 \ln(1/r_2)$
11: update state $X = X + \mathbf{v}_\mu$
12: set $t = t + \tau$
13: **end while**

It is also called:

- The Gillespie method
- Kinetic Monte Carlo
- Doob's Method
- etc

Marchetti, Priami, and Thanh, *Simulation Algorithms for Computational Systems Biology*

# What is the Direct Method SSA?

It is the simplest method for sampling the next reaction time and next reaction.

**Algorithm 2** Direct Method (DM)

**Input:** a biochemical reaction network of $M$ reactions in which each reaction $R_j$, $j = 1, \ldots, M$, is accompanied with the state change vector $\mathbf{v}_j$ and the propensity $a_j$, the initial state $\mathbf{x}_0$ at time 0 and the simulation ending time $T_{max}$

**Output:** a trajectory $X(t)$, $0 \leq t \leq T_{max}$, of the biochemical reaction network

1: initialize time $t = 0$ and state $X = \mathbf{x}_0$
2: **while** $(t < T_{max})$ **do**
3: set $a_0 = 0$
4: **for all** (reaction $R_j$) **do**
5: compute $a_j$
6: update $a_0 = a_0 + a_j$
7: **end for**
8: generate two random numbers $r_1, r_2 \sim \mathrm{U}(0,1)$ (see Appendix B.1)
9: select $R_\mu$ with the smallest index $\mu$ such that $\sum_{j=1}^{\mu} a_j \geq r_1 a_0$
10: compute $\tau = 1/a_0 \ln(1/r_2)$
11: update state $X = X + \mathbf{v}_\mu$
12: set $t = t + \tau$
13: **end while**

***Updating*** propensity values, $O(M)$ work per step.

***Searching*** for the next reaction, **worst case** $O(M)$**.**

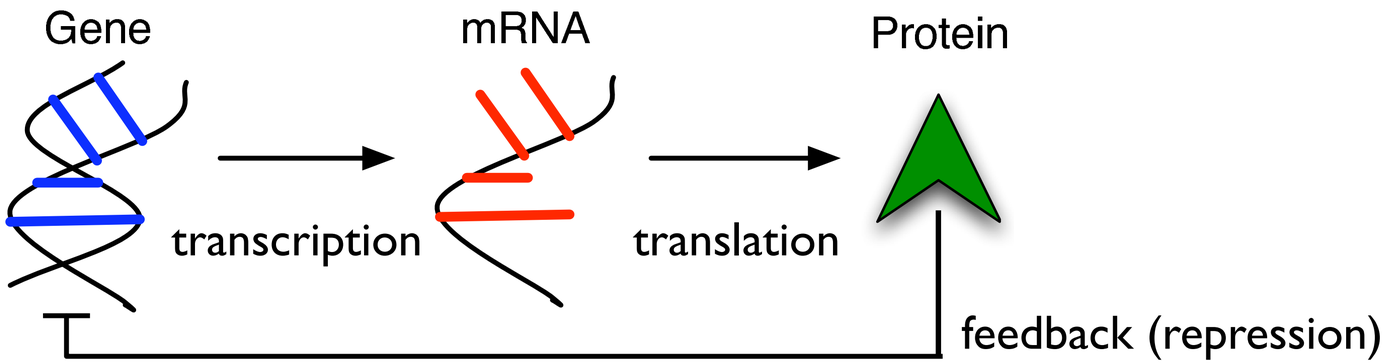
It is also called:

- The Gillespie method
- Kinetic Monte Carlo
- Doob's Method

Marchetti, Priami, and Thanh, *Simulation Algorithms for Computational Systems Biology*

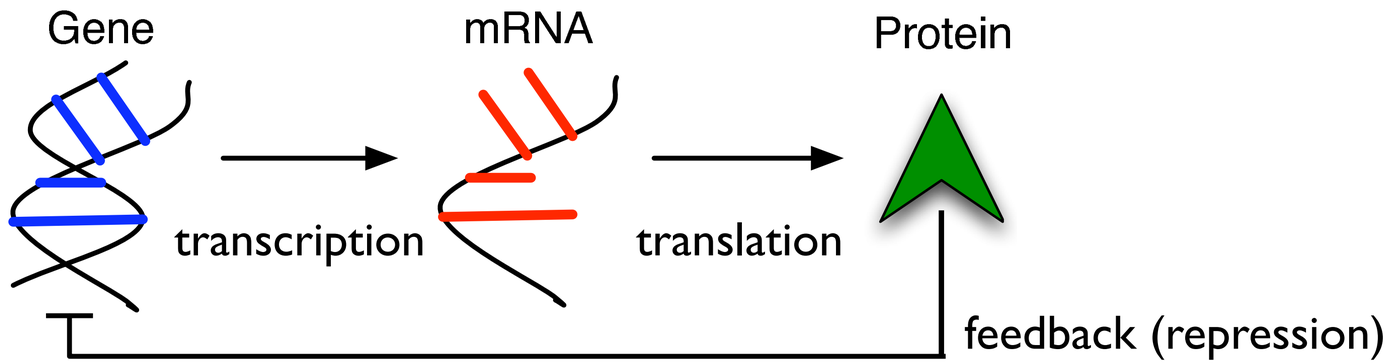
# What example will we illustrate a basic Direct Method on?

Consider the following simple negative feedback autoregulatory model:



# What example will we illustrate a basic Direct Method on?

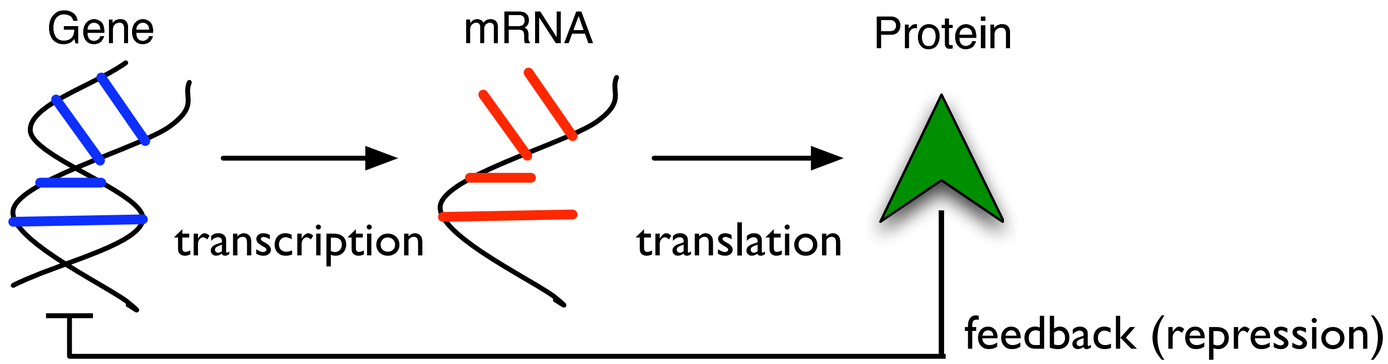
Consider the following simple negative feedback autoregulatory model:




| | | |
|---|---|---|
| $R_1$: | $DNA \rightarrow DNA + mRNA$ | transcription |
| $R_2$: | $mRNA \rightarrow mRNA + P$ | translation |
| $R_3$: | $mRNA \rightarrow \emptyset$ | mRNA degradation |
| $R_4$: | $P \rightarrow \emptyset$ | protein degradation |
| $R_5$: | $DNA + P \rightarrow DNAR$ | proteins can bind and repress the gene |
| $R_6$: | $DNAR \rightarrow DNA + P$ | bound proteins can unbind and unrepress the gene |

# What example will we illustrate a basic Direct Method on?

Consider the following simple negative feedback autoregulatory model:




| | | |
|---|---|---|
| $R_1$: | $DNA \rightarrow DNA + mRNA$ | transcription |
| $R_2$: | $mRNA \rightarrow mRNA + P$ | translation |
| $R_3$: | $mRNA \rightarrow \emptyset$ | mRNA degradation |
| $R_4$: | $P \rightarrow \emptyset$ | protein degradation |
| $R_5$: | $DNA + P \rightarrow DNAR$ | proteins can bind and repress the gene |
| $R_6$: | $DNAR \rightarrow DNA + P$ | bound proteins can unbind and unrepress the gene |

*Let's look at a basic Direct Method implementation for this model in our notebook.*

## What are some algorithmic optimizations we can make to generate more efficient, but still statistically exact SSAs?

As we saw, the Direct method is $O(M)$ work per step.

- One clear optimization would be to only recalculate propensities that have changed after the state update.
  - Most optimized algorithms use *dependency graphs* to only update propensities that depend on species whose number changed in the current step.
  - This exploits that such graphs are usually *sparse* for chemical reaction systems (i.e. the states changed by a given reaction are only depended on by a small number of other reactions).
  - If one has a system with a dense graph there probably won't be a big speedup.
  - If, on average, only $D$ reactions need their propensities recalculated, the work per step in updating propensities becomes $O(D)$ instead of $O(M)$ .

## What are some algorithmic optimizations we can make to generate more efficient, but still statistically exact SSAs?

As we saw, the Direct method is $O(M)$ work per step.

- One clear optimization would be to only recalculate propensities that have changed after the state update.
  - Most optimized algorithms use *dependency graphs* to only update propensities that depend on species whose number changed in the current step.
  - This exploits that such graphs are usually *sparse* for chemical reaction systems (i.e. the states changed by a given reaction are only depended on by a small number of other reactions).
  - If one has a system with a dense graph there probably won't be a big speedup.
  - If, on average, only $D$ reactions need their propensities recalculated, the work per step in updating propensities becomes $O(D)$ instead of $O(M)$ .

Marchetti et al. call the resulting method the *Enhanced Direct Method* (EDM).

# How do we typically describe a reaction-reaction dependency graph?

Consider our gene regulation model:

$$
\begin{aligned}
R_1: &\quad DNA \rightarrow DNA + mRNA \\
R_2: &\quad mRNA \rightarrow mRNA + P \\
R_3: &\quad mRNA \rightarrow \emptyset \\
R_4: &\quad P \rightarrow \emptyset \\
R_5: &\quad DNA + P \rightarrow DNAR \\
R_6: &\quad DNAR \rightarrow DNA + P
\end{aligned}
$$

## How do we typically describe a reaction-reaction dependency graph?

Consider our gene regulation model:

$$
\begin{aligned}
R_1: &\quad DNA \rightarrow DNA + mRNA \\
R_2: &\quad mRNA \rightarrow mRNA + P \\
R_3: &\quad mRNA \rightarrow \emptyset \\
R_4: &\quad P \rightarrow \emptyset \\
R_5: &\quad DNA + P \rightarrow DNAR \\
R_6: &\quad DNAR \rightarrow DNA + P
\end{aligned}
$$

- $R_1$ changes $mRNA$, which determines the propensities for $R_2$ and $R_3$.
- Hence $\text{Dependents}(R_1) = \{R_1, R_2, R_3\}$.
  - To ensure its propensity is also updated, a reaction is always a dependent of itself.
- $R_2$ changes $P$, hence $\text{Dependents}(R_2) = \{R_2, R_4, R_5\}$.
- Similarly $\text{Dependents}(R_3) = \{R_2, R_3\}$, $\text{Dependents}(R_4) = \{R_4, R_5\}$, $\text{Dependents}(R_5) = \{R_1, R_4, R_5, R_6\}$, $\text{Dependents}(R_6) = \{R_1, R_2, R_5, R_6\}$.

## How do we typically describe a reaction-reaction dependency graph?

Consider our gene regulation model:

$$
\begin{aligned}
R_1: &\quad DNA \rightarrow DNA + mRNA \\
R_2: &\quad mRNA \rightarrow mRNA + P \\
R_3: &\quad mRNA \rightarrow \emptyset \\
R_4: &\quad P \rightarrow \emptyset \\
R_5: &\quad DNA + P \rightarrow DNAR \\
R_6: &\quad DNAR \rightarrow DNA + P
\end{aligned}
$$

- $R_1$ changes $mRNA$, which determines the propensities for $R_2$ and $R_3$.
- Hence $\text{Dependents}(R_1) = \{R_1, R_2, R_3\}$.
  - To ensure its propensity is also updated, a reaction is always a dependent of itself.
- $R_2$ changes $P$, hence $\text{Dependents}(R_2) = \{R_2, R_4, R_5\}$.
- Similarly $\text{Dependents}(R_3) = \{R_2, R_3\}$, $\text{Dependents}(R_4) = \{R_4, R_5\}$, $\text{Dependents}(R_5) = \{R_1, R_4, R_5, R_6\}$, $\text{Dependents}(R_6) = \{R_1, R_2, R_5, R_6\}$.
- $D$ is then the average number of dependents of a reaction.
  - In this example, $D = 3$.

# What is the Enhanced Direct Method?

**Algorithm 3** Enhanced Direct Method (EDM)

**Input:** a biochemical reaction network of $M$ reactions in which each reaction $R_j$, $j = 1, \ldots, M$, is accompanied with the state change vector $\mathbf{v}_j$ and the propensity $a_j$, the initial state $\mathbf{x}_0$ at time 0 and the simulation ending time $T_{max}$

**Output:** a trajectory $X(t)$, $0 \leq t \leq T_{max}$, of the biochemical reaction network

1: build the reaction dependency graph $G$
2: set $a_0 = 0$
3: **for all** (reaction $R_j$) **do**
4: compute $a_j$
5: update $a_0 = a_0 + a_j$
6: **end for**
7: initialize time $t = 0$ and state $X = \mathbf{x}_0$
8: **while** $(t < T_{max})$ **do**
9: generate two random numbers $r_1, r_2 \sim \mathtt{U}(0,1)$ (see Appendix B.1)
10: select $R_\mu$ with the smallest index $\mu$ such that $\sum_{j=1}^{\mu} a_j \geq r_1 \, a_0$
11: compute $\tau = 1/a_0 \ln(1/r_2)$
12: update state $X = X + \mathbf{v}_\mu$
13: set $t = t + \tau$
14: **for all** (reaction $R_j \in$ Dependents$(R_\mu)$) **do**
15: compute $a_j^{new}$
16: update $a_0 = a_0 + (a_j^{new} - a_j)$
17: set $a_j = a_j^{new}$
18: **end for**
19: **end while**

# What is the Enhanced Direct Method?

**Algorithm 3** Enhanced Direct Method (EDM)

**Input:** a biochemical reaction network of $M$ reactions in which each reaction $R_j$, $j = 1, \ldots, M$, is accompanied with the state change vector $\mathbf{v}_j$ and the propensity $a_j$, the initial state $\mathbf{x}_0$ at time 0 and the simulation ending time $T_{max}$

**Output:** a trajectory $X(t)$, $0 \leq t \leq T_{max}$, of the biochemical reaction network

1: build the reaction dependency graph $G$
2: set $a_0 = 0$
3: **for all** (reaction $R_j$) **do**
4: compute $a_j$
5: update $a_0 = a_0 + a_j$
6: **end for**
7: initialize time $t = 0$ and state $X = \mathbf{x}_0$
8: **while** $(t < T_{max})$ **do**
9: generate two random numbers $r_1, r_2 \sim \mathrm{U}(0,1)$ (see Appendix B.
10: select $R_\mu$ with the smallest index $\mu$ such that $\sum_{j=1}^{\mu} a_j \geq r_1 \, a_0$
11: compute $\tau = 1/a_0 \ln(1/r_2)$
12: update state $X = X + \mathbf{v}_\mu$
13: set $t = t + \tau$
14: **for all** (reaction $R_j \in$ Dependents$(R_\mu)$) **do**
15: compute $a_j^{new}$
16: update $a_0 = a_0 + (a_j^{new} - a_j)$
17: set $a_j = a_j^{new}$
18: **end for**
19: **end while**

***Searching*** for the next reaction, **worst case** $O(M)$**.**

***Updating*** dependent propensities and total propensity after a reaction is now $O(D)$ work. Notice, we don't re-sum all the propensities, but instead now update a running sum of the total propensity ($a_0$ here)!

## What are some additional algorithmic optimizations we can make to generate more efficient, but still statistically exact sampling algorithms?

- Suppose over timescales much longer than the typical time between reactions, $\tau$, a few reactions dominate, i.e. have the largest propensities.
  - We can reorder the propensity vector so that these are the first ones checked when sampling which reactions occur next.
  - This gives the *Sorting Direct Method*, which monitors which reactions are occurring most frequently and reorders them to come first in the propensity vector.
  - $O(M) + O(D)$ (worst-case) work like the Enhanced Direct Method, but can be much faster in practice.

## What are some additional algorithmic optimizations we can make to generate more efficient, but still statistically exact sampling algorithms?

- Using *rejection sampling* can both speed up the search for the next reaction, and reduce the frequency of needing to recalculate propensities.
  - Rejection SSA (*RSSA*) minimizes need to recalculate propensities.
  - If on average skip a fraction $\beta$ of propensity updates get $O((1-\beta)D)$ total work.

## What are some additional algorithmic optimizations we can make to generate more efficient, but still statistically exact sampling algorithms?

- Using *rejection sampling* can both speed up the search for the next reaction, and reduce the frequency of needing to recalculate propensities.
  - Rejection SSA (*RSSA*) minimizes need to recalculate propensities.
  - If on average skip a fraction $\beta$ of propensity updates get $O((1-\beta)D)$ total work.

- Storing propensities in various *data structures* can also speed up the search and updating process.
  - Direct Method with Composition-Rejection (*DirectCR*), uses a table data structure combined with rejection sampling to make searching *constant order*, and updating $O(D)$.
  - RSSA with Composition-Rejection (*RSSACR*), combines *RSSA* and *DirectCR* to make constant order searching with $O((1-\beta)D)$ updating.

# What is the Rejection SSA (*RSSA*)?

Consider the reaction $R_1 := B + C \xrightarrow{k} \emptyset$

- Let $b$ and $c$ denote the current values of $B(t)$ and $C(t)$.
- Let $a_1(b, c) = kbc$ denote the propensity for the reaction.
- If $\underline{b} \leq b \leq \bar{b}$ and $\underline{c} \leq c \leq \bar{c}$ then
  - $a_1(\underline{b}, \underline{c}) \leq a_1(b, c) \leq a_1(\bar{b}, \bar{c})$.
  - *i.e. the propensities are monotone in the species amounts!*

# What is the Rejection SSA (*RSSA*)?

Consider the reaction $R_1 := B + C \xrightarrow{k} \emptyset$

- Let $b$ and $c$ denote the current values of $B(t)$ and $C(t)$.
- Let $a_1(b, c) = kbc$ denote the propensity for the reaction.
- If $\underline{b} \leq b \leq \bar{b}$ and $\underline{c} \leq c \leq \bar{c}$ then
  - $a_1(\underline{b}, \underline{c}) \leq a_1(b, c) \leq a_1(\bar{b}, \bar{c})$.
  - *i.e. the propensities are monotone in the species amounts!*

Most common propensity functions are monotone, e.g. mass action rates or Hill functions.

# What is the Rejection SSA (*RSSA*)?

Consider the reaction $R_1 := B + C \xrightarrow{k} \emptyset$

- Let $b$ and $c$ denote the current values of $B(t)$ and $C(t)$.
- Let $a_1(b, c) = kbc$ denote the propensity for the reaction.
- If $\underline{b} \leq b \leq \bar{b}$ and $\underline{c} \leq c \leq \bar{c}$ then
  - $a_1(\underline{b}, \underline{c}) \leq a_1(b, c) \leq a_1(\bar{b}, \bar{c})$.
  - *i.e. the propensities are monotone in the species amounts!*

Most common propensity functions are monotone, e.g. mass action rates or Hill functions.

- Idea: in the Direct Method replace propensities by their upper bound for some *appropriately chose* $\bar{b}$ and $\bar{c}$, i.e. $a_1(b, c)$ by $a_1(\bar{b}, \bar{c})$, when calculating the next reaction time and type.
- Use rejection sampling to decide whether to accept the selected reaction.
- After updating the state, we only recalculate the propensity bounds when the species bound are first violated.
  - For the reaction above this would occur when $b \notin [\underline{b}, \bar{b}]$ or $c \notin [\underline{b}, \bar{b}]$.
- We are able to avoid some fraction of updates, reducing the update cost from $O(D)$!

## What is a pseudocode for the Rejection SSA (*RSSA*)?

Assume all species $\underline{x}_i \leq x_i \leq \bar{x}_i$ and all propensities $a_k(\underline{\boldsymbol{x}}) \leq a_k(\boldsymbol{x}) \leq a_k(\bar{\boldsymbol{x}})$ are bounded.

Assume we have stored $\boldsymbol{x}$, calculated $\{a_k(\underline{\boldsymbol{x}})\}_{k=1}^{M}$, $\{a_k(\bar{\boldsymbol{x}})\}_{k=1}^{M}$.

1. Calculate $a(\bar{\boldsymbol{x}}) = \Sigma_{k=1}^{M} a_k(\bar{\boldsymbol{x}})$.
2. Sample $\tau$ from an exponential distribution with parameter $a(\bar{\boldsymbol{x}})$.
3. $t \to t + \tau$
4. For $u_1 \sim \mathcal{U}([0,1))$, find the smallest $k$ such that $\Sigma_{j=1}^{k} a_j(\bar{\boldsymbol{x}}) \geq u_1 a(\bar{\boldsymbol{x}})$.
5. For $u_2 \sim \mathcal{U}([0,1))$,
   a. **Accept** the reaction if $u_2 a_k(\bar{\boldsymbol{x}}) \leq a_k(\underline{\boldsymbol{x}})$.
   b. **Accept** the reaction if $u_2 a_k(\bar{\boldsymbol{x}}) \leq a_k(\boldsymbol{x})$.
   c. Else **reject** the reaction.
6. If accepted the reaction, update $\boldsymbol{x}$.
   a. For any $x_i \notin [\underline{x}_i, \bar{x}_i]$ calculate a new bounding interval.
   b. For all reactions with propensities that depend on $x_i$ recalculate $a_k(\underline{\boldsymbol{x}})$ and $a_k(\bar{\boldsymbol{x}})$.
   c. Update $a(\bar{x})$ for these changed propensities.

# What is a pseudocode for the Rejection SSA (*RSSA*)?

Assume all species $\underline{x}_i \leq x_i \leq \bar{x}_i$ and all propensities $a_k(\underline{\boldsymbol{x}}) \leq a_k(\boldsymbol{x}) \leq a_k(\bar{\boldsymbol{x}})$ are bounded.

Assume we have stored $\boldsymbol{x}$, calculated $\{a_k(\underline{\boldsymbol{x}})\}_{k=1}^{M}$, $\{a_k(\bar{\boldsymbol{x}})\}_{k=1}^{M}$.

1. Calculate $a(\bar{\boldsymbol{x}}) = \Sigma_{k=1}^{M} a_k(\bar{\boldsymbol{x}})$.
2. Sample $\tau$ from an exponential distribution with parameter $a(\bar{\boldsymbol{x}})$.
3. $t \to t + \tau$
4. For $u_1 \sim \mathcal{U}([0,1))$, find the smallest $k$ such that $\Sigma_{j=1}^{k} a_j(\bar{\boldsymbol{x}}) \geq u_1 a(\bar{\boldsymbol{x}})$.
5. For $u_2 \sim \mathcal{U}([0,1))$,
   a. **Accept** the reaction if $u_2 a_k(\bar{\boldsymbol{x}}) \leq a_k(\underline{\boldsymbol{x}})$.
   b. **Accept** the reaction if $u_2 a_k(\bar{\boldsymbol{x}}) \leq a_k(\boldsymbol{x})$.
   c. Else **reject** the reaction.
6. If accepted the reaction, update $\boldsymbol{x}$.
   a. For any $x_i \notin [\underline{x}_i, \bar{x}_i]$ calculate a new bounding interval.
   b. For all reactions with propensities that depend on $x_i$ recalculate $a_k(\underline{\boldsymbol{x}})$ and $a_k(\bar{\boldsymbol{x}})$.
   c. Update $a(\bar{x})$ for these changed propensities.

- As $M$ increases, updating is often the dominant cost in Direct Method variants.
- Now only $O((1-\beta)D)$ where $\beta$ is the average fraction of skipped updates.
- How the bounds, e.g. $\bar{b}$ and $\underline{b}$, are chosen can impact performance.

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA?

- Propensities are partitioned into $L$ *groups*.
- $G_l = \{a_k(\boldsymbol{x}) \,|\, 2^{\sigma_l - 1} \leq a_k(\boldsymbol{x}) < 2^{\sigma_l}\}$ = the $l$th group.
  - Here for an appropriate integer $I$ the $\sigma_l$ are chosen as $I + l - 1$.
- For each group we store the sum of its current propensities $a^l = \Sigma_{\{k|a_k(\boldsymbol{x}) \in G_l\}} a_k(\boldsymbol{x})$.

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA?

- Propensities are partitioned into $L$ *groups*.
- $G_l = \{a_k(\boldsymbol{x}) \,|\, 2^{\sigma_l - 1} \leq a_k(\boldsymbol{x}) < 2^{\sigma_l}\}$ = the $l$th group.
  - Here for an appropriate integer $I$ the $\sigma_l$ are chosen as $I + l - 1$.
- For each group we store the sum of its current propensities $a^l = \Sigma_{\{k|a_k(\boldsymbol{x})\in G_l\}} a_k(\boldsymbol{x})$.

- Sampling which reaction occurs then involves two steps:

1. Find which group $ua(\boldsymbol{x})$ lies within, i.e. the smallest $l$ such that

$$\sum_{l'=1}^{l-1} a^{l'}(\boldsymbol{x}) \leq ua(\boldsymbol{x}) < \sum_{l'=1}^{l} a^{l'}(\boldsymbol{x}) \qquad (O(L) \text{ work in the worst case})$$

2. Use rejection sampling to select which reaction from $G_l$ occurs:

   (a) Sample a reaction index, $k$, uniformly from those stored in $G_l$.

   (b) Accept $k$ if for $u' \in \mathcal{U}([0,1))$ we have $u' \leq \frac{a_k(\boldsymbol{x})}{2^{\sigma_l}}$.

   (c) Else repeat until a reaction is accepted.

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA?

- Propensities are partitioned into $L$ *groups*.
- $G_l = \{a_k(\boldsymbol{x}) \,|\, 2^{\sigma_l - 1} \leq a_k(\boldsymbol{x}) < 2^{\sigma_l}\}$ = the $l$th group.
  - Here for an appropriate integer $I$ the $\sigma_l$ are chosen as $I + l - 1$.
- For each group we store the sum of its current propensities $a^l = \Sigma_{\{k|a_k(\boldsymbol{x})\in G_l\}} a_k(\boldsymbol{x})$.

- Sampling which reaction occurs then involves two steps:

1. Find which group $ua(\boldsymbol{x})$ lies within, i.e. the smallest $l$ such that

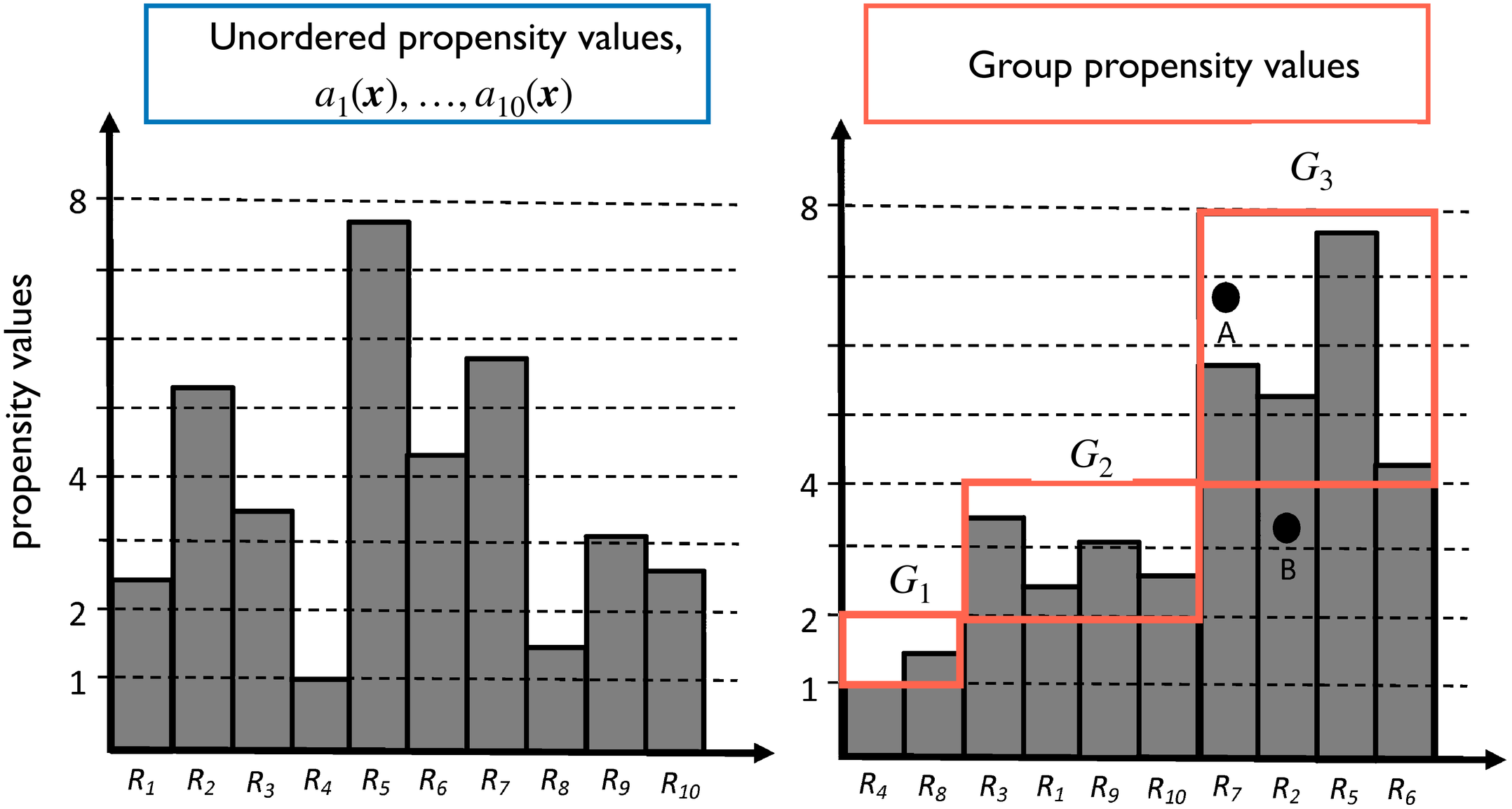
$$\sum_{l'=1}^{l-1} a^{l'}(\boldsymbol{x}) \leq ua(\boldsymbol{x}) < \sum_{l'=1}^{l} a^{l'}(\boldsymbol{x}) \qquad (O(L) \text{ work in the worst case})$$

2. Use rejection sampling to select which reaction from $G_l$ occurs:
   (a) Sample a reaction index, $k$, uniformly from those stored in $G_l$.
   (b) Accept $k$ if for $u' \in \mathcal{U}([0,1))$ we have $u' \leq \frac{a_k(\boldsymbol{x})}{2^{\sigma_l}}$.
   (c) Else repeat until a reaction is accepted.

*By the definition of the groups, the acceptance probability is* $\geq \frac{1}{2}$.

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA? (2)

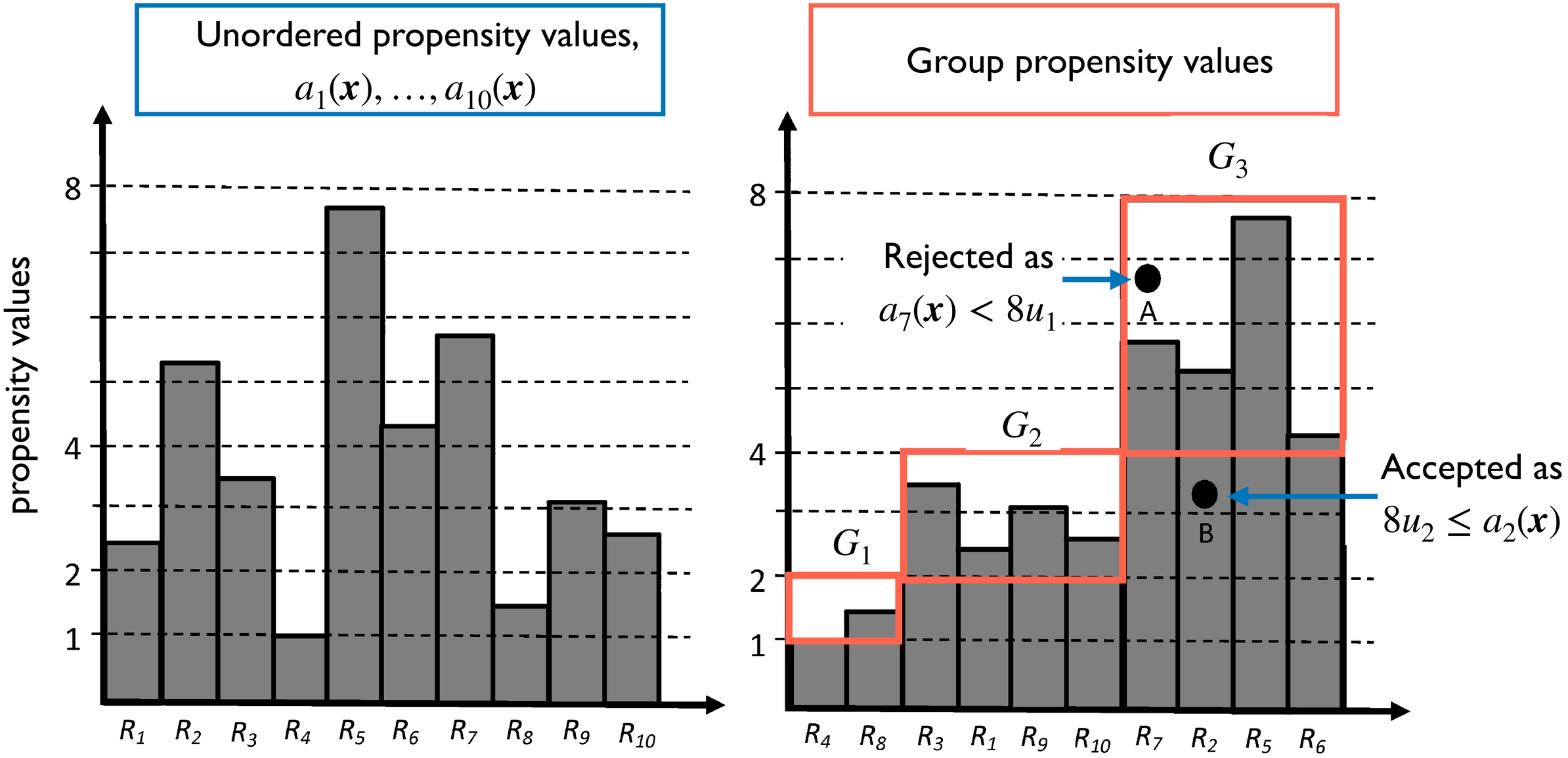
Example[1], suppose we have a system with 10 reactions:




Taken from Marchetti, Priami, and Thanh, *Simulation Algorithms for Computational Systems Biology*

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA? (2)

Example[1], suppose we have a system with 10 reactions:




Taken from Marchetti, Priami, and Thanh, *Simulation Algorithms for Computational Systems Biology*

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA? (3)

- Updating a single reaction propensity within the table is constant work.
  - The reaction gets its propensity value updated in the table.
  - By storing a map from reaction index $k$ to group and location within the group this requires no searching.
  - If the new value is outside its current group, it is swapped with the last reaction in its group, and appended to the end of the new group it should be stored in.
- Updating after the reaction is executed uses a dependency graph and is therefore again $O(D)$ work.

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA? (3)

- Updating a single reaction propensity within the table is constant work.
  - The reaction gets its propensity value updated in the table.
  - By storing a map from reaction index $k$ to group and location within the group this requires no searching.
  - If the new value is outside its current group, it is swapped with the last reaction in its group, and appended to the end of the new group it should be stored in.
- Updating after the reaction is executed uses a dependency graph and is therefore again $O(D)$ work.
- Finding the next reaction group by a linear search is $O(L)$ in the worst case.
- Sampling the next reaction within the group is constant order work (over half the time we don't reject, so expect only a few rejected sampling attempts will occur usually).
- $O(L) + O(D)$ total work per step.

# What is the Direct Method with Composition-Rejection (*DirectCR*) SSA? (3)

- Updating a single reaction propensity within the table is constant work.
  - The reaction gets its propensity value updated in the table.
  - By storing a map from reaction index $k$ to group and location within the group this requires no searching.
  - If the new value is outside its current group, it is swapped with the last reaction in its group, and appended to the end of the new group it should be stored in.
- Updating after the reaction is executed uses a dependency graph and is therefore again $O(D)$ work.
- Finding the next reaction group by a linear search is $O(L)$ in the worst case.
- Sampling the next reaction within the group is constant order work (over half the time we don't reject, so expect only a few rejected sampling attempts will occur usually).
- $O(L) + O(D)$ total work per step.

This looks like our best method yet, but in practice the constants in the asymptotic work scalings are sufficiently large that it only starts showing good performance for *large* networks.

# What is the RSSA with Composition-Rejection (*RSSACR*)?

- Combines RSSA with the Direct Method with Composition-Rejection.
  - Our groups now store the upper bound propensities instead $a_k(\boldsymbol{x})$.
  - C*andidate* reaction sampling is the same as DirectCR.
  - Once a candidate reaction is selected the RSSA rejection algorithm is run to determine whether to accept it or not.
  - Propensities stored in the table then only need to be recalculated when the new value of $\boldsymbol{x}$ is no longer between $\underline{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$.

# What is the RSSA with Composition-Rejection (*RSSACR*)?

- Combines RSSA with the Direct Method with Composition-Rejection.
  - Our groups now store the upper bound propensities instead $a_k(\boldsymbol{x})$.
  - C*andidate* reaction sampling is the same as DirectCR.
  - Once a candidate reaction is selected the RSSA rejection algorithm is run to determine whether to accept it or not.
  - Propensities stored in the table then only need to be recalculated when the new value of $\boldsymbol{x}$ is no longer between $\underline{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$.

- The expected work is now $O(L) + O((1-\beta)D)$ for searching and updating respectively.
  - This is our asymptotically fastest method, but again needs a large enough system to show performance advantages.
  - Note, both *RSSA* and *RSSACR* as formulated require more memory than other methods since they store upper and lower bounds on the propensities!

# What are First Reaction Type Methods?

- The Direct method samples the time of the next reaction, and which reaction occurs at that time.
- It is *statistically equivalent* to
  - Sample for each reaction the time, $\tau_k$, until which it would *independently* occur if it were the *only* reaction within the system
  - Chose the time until the next reaction, and which reaction occurs as corresponding to the minimum of these times.
  - Called the First Reaction Method (*FRM*).

# What are First Reaction Type Methods?

- The Direct method samples the time of the next reaction, and which reaction occurs at that time.
- It is *statistically equivalent* to
  - Sample for each reaction the time, $\tau_k$, until which it would *independently* occur if it were the *only* reaction within the system
  - Chose the time until the next reaction, and which reaction occurs as corresponding to the minimum of these times.
  - Called the First Reaction Method (*FRM*).

Hence given $\boldsymbol{X}(t) = \boldsymbol{x}$ we can

1. Sample $\tau_k$ from the exponential distribution $a_k(\boldsymbol{x})e^{-a_k(\boldsymbol{x})\tau_k}$ for all $k = 1, \ldots, M$.
2. Set $k = \underset{k' \in \{0,\ldots,M\}}{\operatorname{argmin}} \ \tau_{k'}$
3. Set $t \to t + \tau_k$ and update the state as $\boldsymbol{x} \to \boldsymbol{x} + \nu_k$.
4. Repeat until done simulating.

# What are First Reaction Type Methods?

- The Direct method samples the time of the next reaction, and which reaction occurs at that time.
- It is *statistically equivalent* to
  - Sample for each reaction the time, $\tau_k$, until which it would *independently* occur if it were the *only* reaction within the system
  - Chose the time until the next reaction, and which reaction occurs as corresponding to the minimum of these times.
  - Called the First Reaction Method (*FRM*).

Hence given $\boldsymbol{X}(t) = \boldsymbol{x}$ we can

1. Sample $\tau_k$ from the exponential distribution $a_k(\boldsymbol{x})e^{-a_k(\boldsymbol{x})\tau_k}$ for all $k = 1, \ldots, M$. $O(M)$ work
2. Set $k = \underset{k' \in \{0,\ldots,M\}}{\operatorname{argmin}} \tau_{k'}$ $O(M)$ work
3. Set $t \to t + \tau_k$ and update the state as $\boldsymbol{x} \to \boldsymbol{x} + \nu_k$.
4. Repeat until done simulating.

$O(M)$ total work per step

# What is the (Gibson-Bruck) Next Reaction Method (*NRM*)?

- Instead of considering the time until each reaction occurs, we consider the physical absolute times at which each would occur, i.e. $t_k = t + \tau_k$.
- We store these times in an indexed priority queue (IPQ), a tree-type data structure that can be stored within a single vector.
  - Each node of the tree stores one pair $(t_k, k)$.
  - The tree is ordered such that a parent node has a smaller $t_k$ than its children.
  - The root node is therefore the time of the next reaction and can be accessed in constant time.
  - Updating the tree when times change is $O(\log_2(M))$.

# What is the (Gibson-Bruck) Next Reaction Method (*NRM*)?

- Instead of considering the time until each reaction occurs, we consider the physical absolute times at which each would occur, i.e. $t_k = t + \tau_k$.
- We store these times in an indexed priority queue (IPQ), a tree-type data structure that can be stored within a single vector.
  - Each node of the tree stores one pair $(t_k, k)$.
  - The tree is ordered such that a parent node has a smaller $t_k$ than its children.
  - The root node is therefore the time of the next reaction and can be accessed in constant time.
  - Updating the tree when times change is $O(\log_2(M))$.
- It is statistically exact to only recalculate $t_{k'}$ for each reaction $R_{k'} \in \text{Dependents}(R_k)$.
- So the total work in the method is $O(D \log_2(M))$.

## What is the (Gibson-Bruck) Next Reaction Method (*NRM*)?

- Instead of considering the time until each reaction occurs, we consider the physical absolute times at which each would occur, i.e. $t_k = t + \tau_k$.
- We store these times in an indexed priority queue (IPQ), a tree-type data structure that can be stored within a single vector.
  - Each node of the tree stores one pair $(t_k, k)$.
  - The tree is ordered such that a parent node has a smaller $t_k$ than its children.
  - The root node is therefore the time of the next reaction and can be accessed in constant time.
  - Updating the tree when times change is $O(\log_2(M))$.
- It is statistically exact to only recalculate $t_{k'}$ for each reaction $R_{k'} \in \text{Dependents}(R_k)$.
- So the total work in the method is $O(D \log_2(M))$.

*This is often thought of as one of the fastest methods, but in practice is often much slower than one would expect due to constants in the asymptotic work (for example, nonuniform memory access in updating the IPQ).*

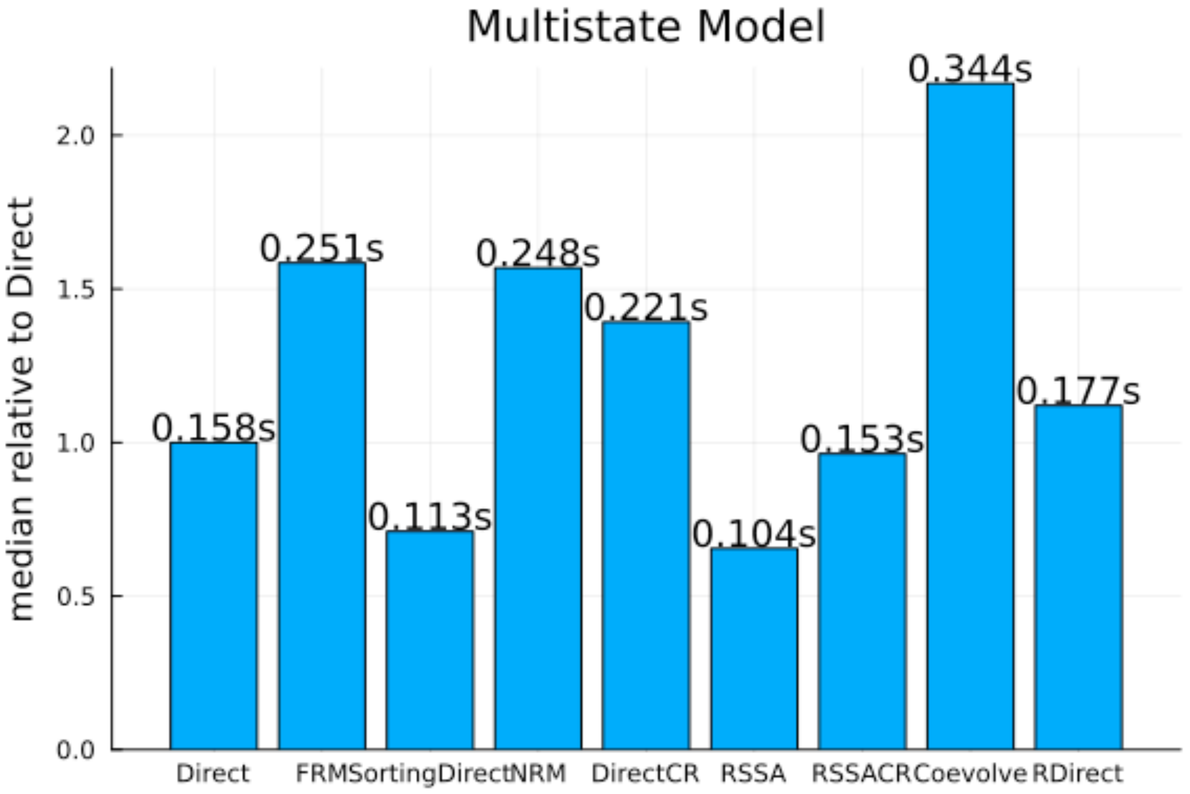
## How do all these methods perform for different biological models?

- Let's start with our gene expression model (see Julia notebook).

# How do all these methods perform for different biological models?

- Let's start with our gene expression model (see Julia notebook).
- Let's see a simple benchmark from a suite we use for testing performance in Catalyst and JumpProcesses, https://docs.sciml.ai/SciMLBenchmarksOutput/stable/.
- This is a Multistate signaling model[1] with 9 species and 18 reactions:

```
kon, S1 + S2 --> S4
kAon, S1 + S3 --> S5
kon, S2 + S5 --> S6
koff, S4 --> S1 + S2
kAon, S3 + S4 --> S6
kAoff, S5 --> S1 + S3
koff, S6 --> S2 + S5
kAoff, S6 --> S3 + S4
kAp, S6 --> S7
koff, S7 --> S2 + S8
kAoff, S7 --> S4 + S9
kAdp, S7 --> S6
kon, S2 + S8 --> S7
kAon, S1 + S9 --> S8
kAon, S4 + S9 --> S7
kAoff, S8 --> S1 + S9
kAdp, S8 --> S5
kAdp, S9 --> S3
```

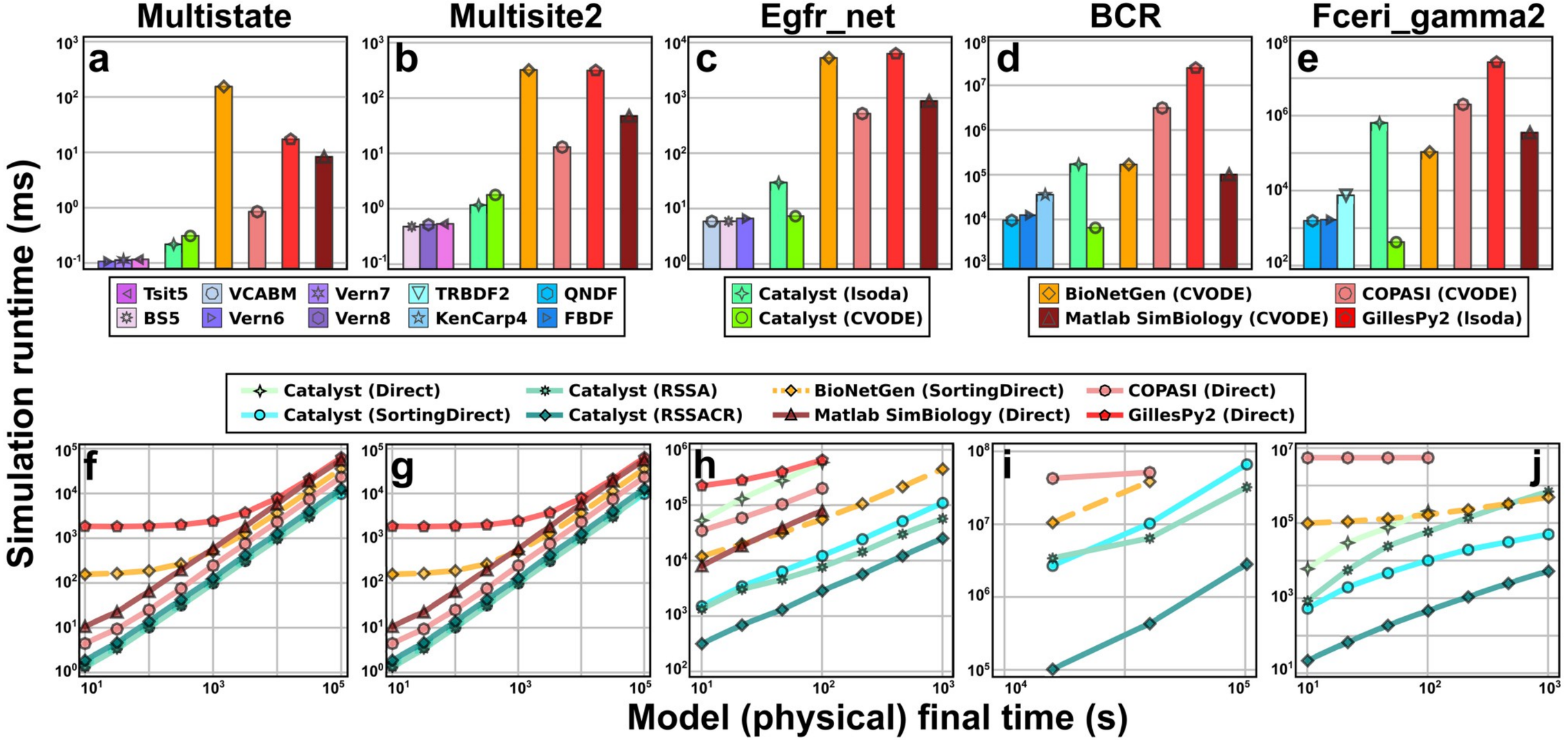



1. Gupta and Mendes, *An Overview of Network-Based and -Free Approaches for Stochastic Simulation of Biochemical Systems*, Computation, 6 (9), 2018.

# How do all these methods perform for different biological models? (2)

***Number of:***

| | | | | | |
|---|---|---|---|---|---|
| Species | 9 | 66 | 356 | 1122 | 3744 |
| Reactions | 18 | 288 | 3749 | 24388 | 58276 |




T. Loman, …, and S.A. Isaacson, *Catalyst: Fast and flexible modeling of reaction networks*, PLOS Computational Biology 19(10): e1011530 (2023).

# What methods have I left out?

- Constant Complexity NRM by Sanft and Othmer.
  - Replaces the IPQ with a table that allows constant asymptotic work in updating after a reaction occurs.
  - Similar to the DirectCR method.
  - Should be the fastest of the FRM-family methods.
- Partial Propensity Family Methods (including with composition-rejection).
  - When including composition-rejection total work is $O(L) + O(N)$ where again $L$ is the number of groups and $N$ is the number of *species*.
- Methods that allow time-varying rates (for example, cell growth), delays, hybrid methods, inexact methods, …