Stochastic Simulation Library
Assignment for Selected Topics in Programming

Marius Mikučionis

April 9, 2024

1 Introduction

A rich class of stochastic systems can be simulated using Doob-Gillespie algorithm. Chemical reactions, biological ecosystems, traffic, and even disease epidemics can be modeled as a stochastic systems. Curiously, such systems exhibit emergent behavior: agents follow simple rules and give rise to a complex system behavior which is not part of any individual rules. Moreover, the system is probabilistic and independent simulations may exhibit different behaviors, therefore statistical methods are needed to understand the expected effects.

A stochastic system can be described by a number of agents of various species which interact, consume, and produce agents of other species. The interactions are described by a network of reaction rules. For example, the following reaction takes one agent of species A and one agent of C (catalyst) and produce one agent of B with a rate of λ:

$$\mathsf{A} + \mathsf{C} \xrightarrow{\lambda} \mathsf{B} + \mathsf{C}$$

The rate λ describes the intrinsic probability of individuals meeting and reacting over one time unit. If any input individuals are missing, then the reaction is not possible. If there are multiple individuals available, then the probability for such reaction is proportionally greater, therefore the overall reaction rate is *multiplied* by the number of input individuals available.

Figure 1 shows an evolution of the system described by the above equation, where the reaction rate is twice as high when there are two catalysing individuals C (the amount A is halved twice as fast), and the reaction has a similar half-life (period until the quantity is halved) even when starting with half the amounts. Observe that the quantities follow an exponential decay characteristic to the reaction rate being proportional to the input quantities.

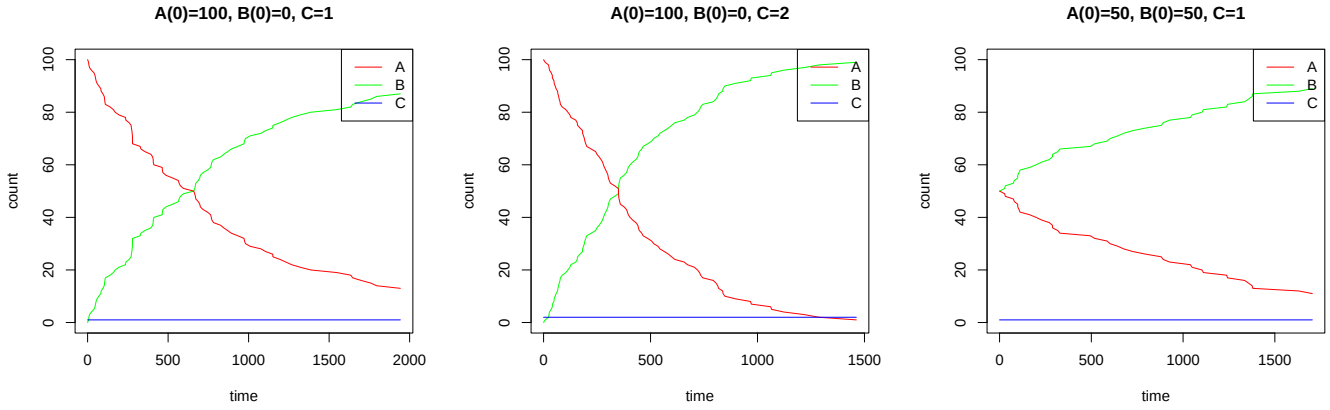

Figure 1: Sample trajectories of A, B and C quantities over time given $\lambda = 0.001$.

The reaction delay (time between reaction events) is stochastic: it follows an exponential distribution with the specific (exponential) reaction rate. If there are multiple reactions possible at the same time, then the reaction with the shortest delay is taken next in the simulation.

The goal of the assignment is to develop a library for stochastic simulations.

1.1 Stochastic Simulation of Reactions

A reaction can be simulated as a sequence of reaction steps. In each step small amounts of reactants react in necessary proportions to produce the products of the reaction. For example, when hard groundwater is exposed to air, carbon dioxide dissolves into carbonate ions which meet calcium ions to form milky limescale precipitate: $\mathsf{CO}_3^{2-} + \mathsf{Ca}^{2+} \xrightarrow{\lambda} \mathsf{CaCO}_3$. If $Q_{\mathsf{CO}_3^{2-}}$ is the number of carbonate ions, $Q_{\mathsf{Ca}^{2+}}$ is the number of calcium ions, and Q_{CaCO_3} is the number of calcium carbonate molecules, then the reaction step is simulated by subtracting 1 ion from $Q_{\mathsf{CO}_3^{2-}}$, 1 from $Q_{\mathsf{Ca}^{2+}}$ and adding 1 to Q_{CaCO_3}. When input amounts decrease, reactants are less likely to meet and hence takes longer to react. The time to reaction is stochastic and is generated as follows:

$$r.delay = Exp\Big(r.\lambda \cdot \textstyle\prod_{i \in r.inputs} Q_i\Big) \qquad (1)$$

Input : Set of reactions R, end time T, State $\langle Q_i \rangle$ with reactant amounts Q_i.

1 $t := 0$;
2 **while** $t \le \mathsf{T}$ **do**
3 | **foreach** $r \in \mathsf{R}$ **do** Compute $r.delay$;
4 | $r := \text{argmin}_{r \in \mathsf{R}}\, r.delay$;
5 | $t := t + r.delay$;
6 | **if** $\forall i \in r.inputs : Q_i > 0$
7 | **then**
8 | | **foreach** $i \in r.input$ **do** $Q_i := Q_i - 1$;
9 | | **foreach** $i \in r.product$ **do** $Q_i := Q_i + 1$;
10 | **end**
11 | Print/store/observe the state $\langle Q_i \rangle$;
12 **end**

Algorithm 1: Simulation of multiple reactions

Here, $Exp(\lambda)$ is a random number distributed according to the exponential distribution with the rate parameter λ (see `std::exponential_distribution`), Q_i is the current amount of the reactant i. After *delay* time units, the reaction step is performed by consuming the input reactants ($Q_i := Q_i - 1$) and producing the reaction products ($Q_i := Q_i + 1$) (lines 8–9 in Alg. 1), as well as computing the new *delay* (line 3). If some input reactant is exhausted, then such reaction is not possible (its delay is infinite).

Algorithm 1 shows how several reactions are simulated. The algorithm uses reactions R and maintains the state of the system as the amounts of all the reactants $\langle Q_i \rangle$. It always picks a reaction with the smallest delay (line 4). An input of one reaction can be a product of another and vice versa, thus forming reaction chains. The delays of *all* reactions are recomputed (line 3) even though only one reaction is executed at a time. All delays can be recomputed at each iteration because the exponential distribution is memory-less. A more efficient implementation could recompute only the delays of the reactions whose rates where affected by the change in quantities. Another improvement could be to treat catalyst agents (present in both inputs and products) specially by omitting decrement and increment of their quantities.

2 Requirements

The goal of the assignment is to develop a library for creating and simulating reactions. The usage of the library should be demonstrated on three example stochastic systems, sample programs and unit tests. The solution should contain the following features:

1. The library should *overload operators* to support the reaction rule typesetting directly in C++ code.
2. Provide *pretty-printing* of the reaction network in a) human readable format and b) network graph (e.g. Fig. 4).
3. Implement a ***generic*** *symbol table* to *store* and *lookup* objects of user-defined key and value types. Support failure cases when a) the table does not contain a looked up symbol, b) the table already contains a symbol that is being added. Demonstrate the usage of the symbol table with the reactants (names and initial counts).
4. Implement the *stochastic simulation* (Alg. 1) of the system using the reaction rules.
5. Demonstrate the *application* of the library on the three examples (shown in Fig. 1, 2, 3).
6. *Display* simulation trajectories of how the amounts change. External tools/libraries can be used to visualize.
7. Implement a ***generic*** *support* for (any) user-supplied state observer function object or provide a lazy trajectory generation interface (coroutine). The observer itself should be supplied by the user/test and ***not*** be part of the library. To demonstrate the generic support, estimate the peak of hospitalized agents in Covid-19 example *without storing entire trajectory data*. Record the peak hospitalization values for population sizes of N_{NJ} and N_{DK}.
8. Implement support for *multiple computer cores by parallelizing* the computation of several simulations at the same time. Estimate the likely (average) value of the hospitalized peak over 100 simulations.
9. Implement *unit tests* (e.g. test symbol table methods, their failure cases, and pretty-printing of reaction rules).
10. *Benchmark* and compare the stochastic simulation performance (e.g. the time it takes to compute 100 simulations a single core, multiple cores, or improved implementation). Record the timings and make your conclusions.

Make sure to include testing and benchmarking code as well as the **sample results, measurements, plots and conclusions** into the report.

A Example Library-Usage Code

The following code listings are just suggested examples, one may choose a different operator overloading design, provided that the requirements are fulfilled.

A.1 Genetic Oscilator for Circadian Rhythm

Table 1 describes the reaction network responsible for the sense of circadian rhythm in humans[1], an example C++ typesetting is shown in Listing 1 and an expected sample trajectory is plotted in Figure 2.

Table 1: Reactions and values for circadian rhythm, where ∅ denotes decay into environment, m^{-1} – "per molecule", h^{-1} – "per hour".

$$\mathsf{A} + \mathsf{D}_A \xrightarrow{\gamma_A} \mathsf{D}'_A$$
$$\mathsf{D}'_A \xrightarrow{\theta_A} \mathsf{D}_A + \mathsf{A}$$
$$\mathsf{A} + \mathsf{D}_R \xrightarrow{\gamma_R} \mathsf{D}'_R$$
$$\mathsf{D}'_R \xrightarrow{\theta_R} \mathsf{D}_R + \mathsf{A}$$
$$\mathsf{D}'_A \xrightarrow{\alpha'_A} \mathsf{M}_A + \mathsf{D}'_A$$
$$\mathsf{D}_A \xrightarrow{\alpha_A} \mathsf{M}_A + \mathsf{D}_A$$
$$\mathsf{D}'_R \xrightarrow{\alpha'_R} \mathsf{M}_R + \mathsf{D}'_R$$
$$\mathsf{D}_R \xrightarrow{\alpha_R} \mathsf{M}_R + \mathsf{D}_R$$

$$\mathsf{M}_A \xrightarrow{\beta_A} \mathsf{M}_A + \mathsf{A}$$
$$\mathsf{M}_R \xrightarrow{\beta_R} \mathsf{M}_R + \mathsf{R}$$
$$\mathsf{A} + \mathsf{R} \xrightarrow{\gamma_C} \mathsf{C}$$
$$\mathsf{C} \xrightarrow{\delta_A} \mathsf{R}$$
$$\mathsf{A} \xrightarrow{\delta_A} \varnothing$$
$$\mathsf{R} \xrightarrow{\delta_R} \varnothing$$
$$\mathsf{M}_A \xrightarrow{\delta_{M_A}} \varnothing$$
$$\mathsf{M}_R \xrightarrow{\delta_{M_R}} \varnothing$$

α_A	$= 50\,\mathrm{h}^{-1}$	δ_{M_R}	$= 0.5\,\mathrm{h}^{-1}$
α'_A	$= 500\,\mathrm{h}^{-1}$	θ_A	$= 50\,\mathrm{h}^{-1}$
α_R	$= 0.01\,\mathrm{h}^{-1}$	θ_R	$= 100\,\mathrm{h}^{-1}$
α'_R	$= 50\,\mathrm{h}^{-1}$	D_A	$= 1\mathrm{m}$
β_A	$= 50\,\mathrm{h}^{-1}$	D'_A	$= 0$
β_R	$= 5\,\mathrm{h}^{-1}$	D_R	$= 1\mathrm{m}$
γ_A	$= 1\,\mathrm{m}^{-1}\mathrm{h}^{-1}$	D'_R	$= 0$
γ_R	$= 1\,\mathrm{m}^{-1}\mathrm{h}^{-1}$	M_A	$= 0$
γ_C	$= 2\,\mathrm{m}^{-1}\mathrm{h}^{-1}$	M_R	$= 0$
δ_A	$= 1\,\mathrm{h}^{-1}$	A	$= 0$
δ_R	$= 0.2\,\mathrm{h}^{-1}$	R	$= 0$
δ_{M_A}	$= 10\,\mathrm{h}^{-1}$	C	$= 0$

Listing 1: Circadian oscillator in C++

```
Vessel circadian_rhythm()
{
    const auto alphaA = 50;
    const auto alpha_A = 500;
    const auto alphaR = 0.01;
    const auto alpha_R = 50;
    const auto betaA = 50;
    const auto betaR = 5;
    const auto gammaA = 1;
    const auto gammaR = 1;
    const auto gammaC = 2;
    const auto deltaA = 1;
    const auto deltaR = 0.2;
    const auto deltaMA = 10;
    const auto deltaMR = 0.5;
    const auto thetaA = 50;
    const auto thetaR = 100;
    auto v = stochastic::Vessel{"Circadian Rhythm"};
    const auto env = v.environment();
    const auto DA = v.add("DA", 1);
    const auto D_A = v.add("D_A", 0);
    const auto DR = v.add("DR", 1);
    const auto D_R = v.add("D_R", 0);
    const auto MA = v.add("MA", 0);
    const auto MR = v.add("MR", 0);
    const auto A = v.add("A", 0);
    const auto R = v.add("R", 0);
    const auto C = v.add("C", 0);
    v.add((A + DA) >> gammaA >>= D_A);
    v.add(D_A >> thetaA >>= DA + A);
    v.add((A + DR) >> gammaR >>= D_R);
    v.add(D_R >> thetaR >>= DR + A);
    v.add(D_A >> alpha_A >>= MA + D_A);
    v.add(DA >> alphaA >>= MA + DA);
    v.add(D_R >> alpha_R >>= MR + D_R);
    v.add(DR >> alphaR >>= MR + DR);
    v.add(MA >> betaA >>= MA + A);
    v.add(MR >> betaR >>= MR + R);
    v.add((A + R) >> gammaC >>= C);
    v.add(C >> deltaA >>= R);
    v.add(A >> deltaA >>= env);
    v.add(R >> deltaR >>= env);
    v.add(MA >> deltaMA >>= env);
    v.add(MR >> deltaMR >>= env);
    return v;
}
```

[1] José M. G. Vilar, Hao Yuan Kueh, Naama Barkai, and Stanislas Leibler. Mechanisms of noise resistance in genetic oscillators. Proceedings of the National Academy of Sciences, 99(9):5988–5992, 2002. doi:10.1073/pnas.092133899

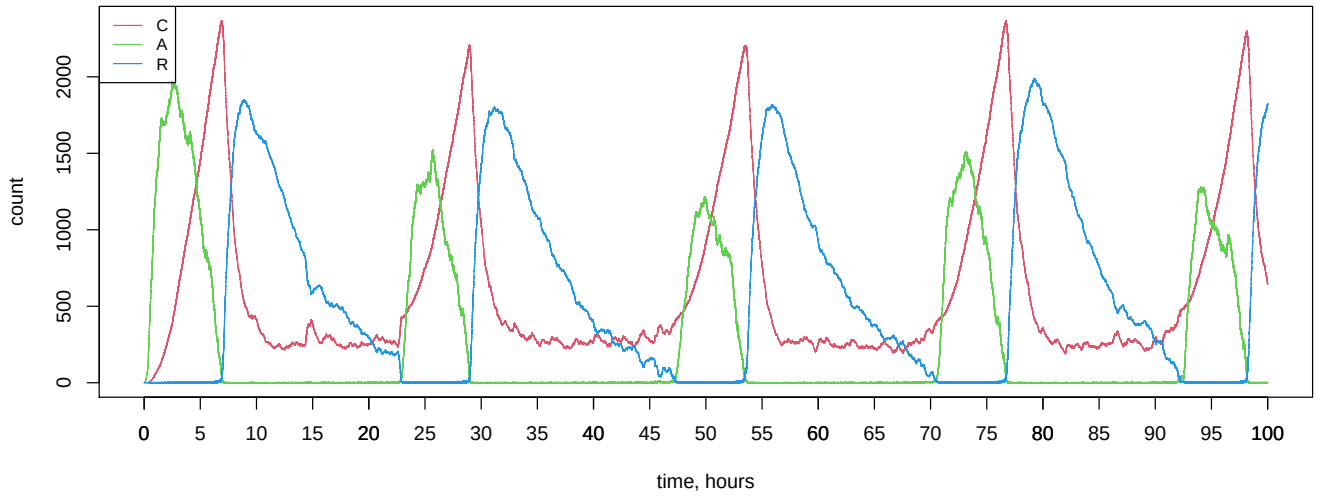

Figure 2: Sample trajectory of circadian rhythm: agent counts repeat with ≈ 24 hour periods.

A.2 SEIHR Model for Covid-19

Table 2 describes Covid-19 epidemic model without any quarantine measures. The model is typeset in C++ in Listing 2 and an expected sample trajectory is shown in Figure 3. Tasks: estimate the expected (mean) and maximum hospitalization peak for the populations of North Jutland (N_{NJ}) and Denmark (N_{DK}).

Table 2: Reactions and values for Covid-19 model, where d^{-1} means "per day".

$$
\begin{aligned}
\mathsf{S} + \mathsf{I} &\xrightarrow{\beta/N} \mathsf{E} + \mathsf{I} \\
\mathsf{E} &\xrightarrow{\alpha} \mathsf{I} \\
\mathsf{I} &\xrightarrow{\gamma} \mathsf{R} \\
\mathsf{I} &\xrightarrow{\kappa} \mathsf{H} \\
\mathsf{H} &\xrightarrow{\tau} \mathsf{R}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{N} &= 10^4 & R_0 &= 2.4 \\
\varepsilon &= 9 \cdot 10^{-4} & \alpha &= 1/5.1\,\mathrm{d}^{-1} \\
\mathsf{E} &= N \cdot \varepsilon \cdot 15 & \gamma &= 1/3.1\,\mathrm{d}^{-1} \\
\mathsf{I} &= N \cdot \varepsilon & \beta &= R_0 \cdot \gamma \\
\mathsf{H} &= 0 & P_{\mathsf{H}} &= 9 \cdot 10^{-4} \\
\mathsf{R} &= 0 & \kappa &= \gamma \cdot P_{\mathsf{H}} \cdot (1 - P_{\mathsf{H}}) \\
\mathsf{N}_{\mathrm{DK}} &= 5'822'763 & \tau &= 1/10.12\,\mathrm{d}^{-1} \\
\mathsf{N}_{\mathrm{NJ}} &= 589'755
\end{aligned}
$$

Listing 2: Covid-19 model in C++

```
/// SEIHR model for COVID19 epidemic with population size N
Vessel seihr(uint32_t N)
{
    auto v = Vessel{"COVID19 SEIHR: " + std::to_string(N)};
    const auto eps = 0.0009;                            // initial fraction of infectious
    const auto I0 = size_t(std::round(eps * N));        // initial infectious
    const auto E0 = size_t(std::round(eps * N * 15));   // initial exposed
    const auto S0 = N - I0 - E0;                        // initial susceptible
    const auto R0 = 2.4;                              // initial basic reproductive number
    const auto alpha = 1.0 / 5.1;                     // incubation rate (E -> I) ~5.1 days
    const auto gamma = 1.0 / 3.1;                     // recovery rate (I -> R) ~3.1 days
    const auto beta = R0 * gamma;                     // infection/generation rate (S+I -> E+I)
    const auto P_H = 0.9e-3;                          // probability of hospitalization
    const auto kappa = gamma * P_H * (1.0 - P_H); // hospitalization rate (I -> H)
    const auto tau = 1.0 / 10.12;                     // removal rate in hospital (H -> R) ~10.12 days
    const auto S = v.add("S", S0);                    // susceptible
    const auto E = v.add("E", E0);                    // exposed
    const auto I = v.add("I", I0);                    // infectious
    const auto H = v.add("H", 0);                     // hospitalized
    const auto R = v.add("R", 0);                     // removed/immune (recovered + dead)
    v.add((S + I) >> beta / N >>= E + I);             // susceptible becomes exposed by infectious
    v.add(E >> alpha >>= I);                          // exposed becomes infectious
```

```
    v.add(I >> gamma >>= R);                    // infectious becomes removed
    v.add(I >> kappa >>= H);                    // infectious becomes hospitalized
    v.add(H >> tau >>= R);                      // hospitalized becomes removed
    return v;
}
```

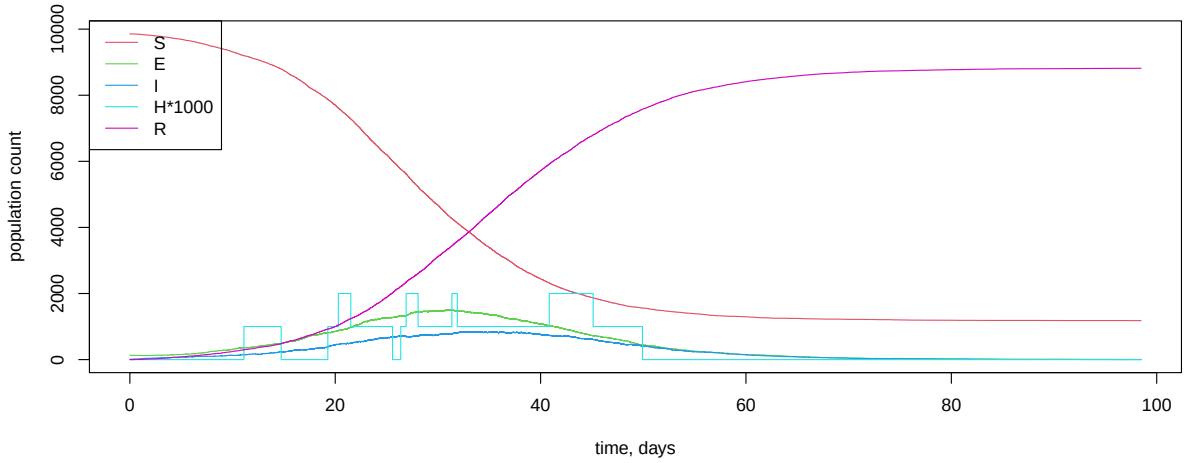

Figure 3: Sample trajectory of Covid-19 population $\mathsf{N} = 10000$: hospitalizations peak at 2.

B Example Graph Layouts

This particular visualization is just an example and not part of the requirements: one may choose a different graph visualization framework and render the image using other *tools*.

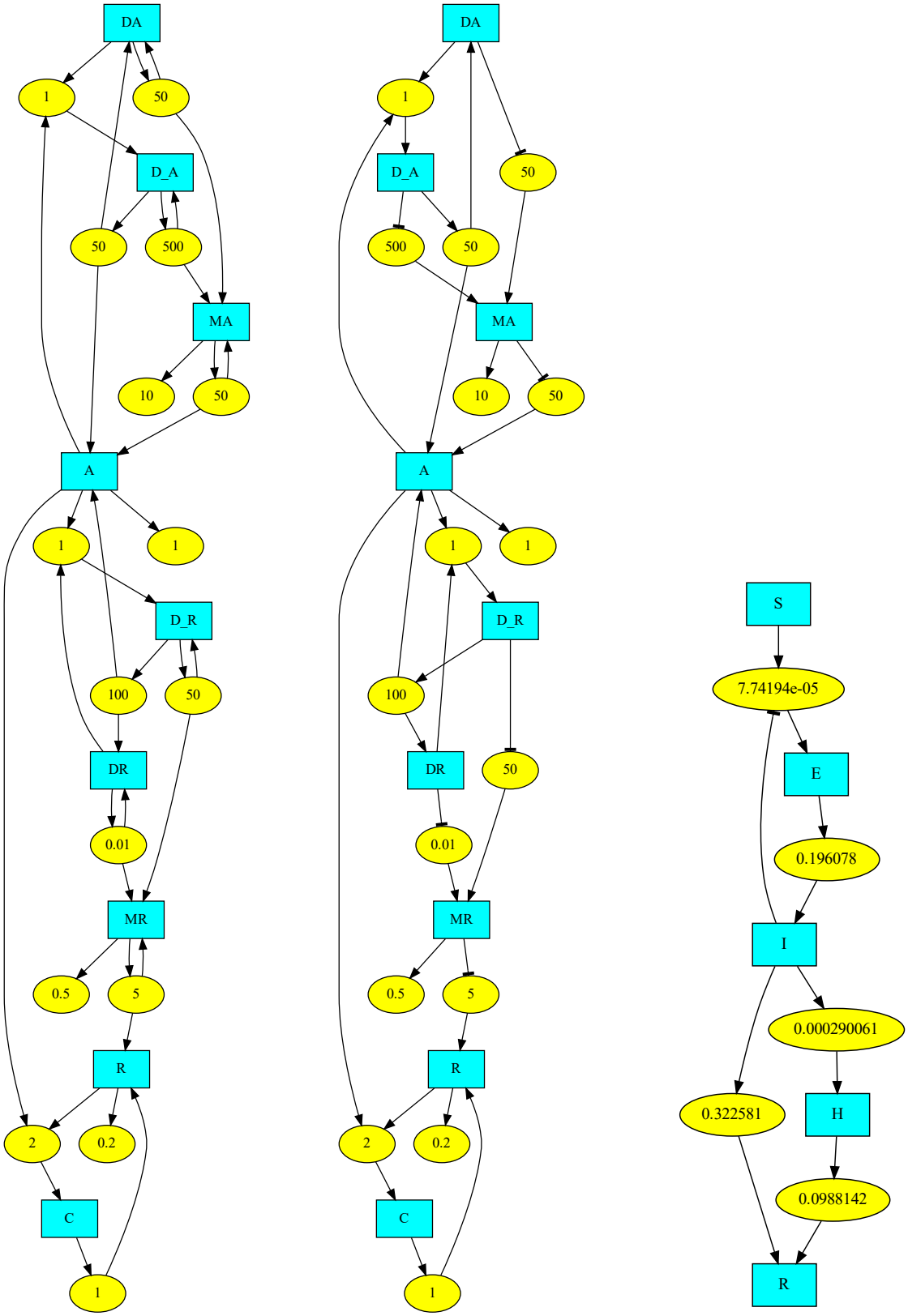

Figure 4: Layout of reaction networks: circadian model, circadian with catalysts, Covid-19 from Listings 3,4,5. The images are rendered using command-line `dot` utility from Graphiz.

C Example Graph Description

The listings below describe the reaction networks in Graphviz `dot` format. The description format is just an example and not part of the requirements: one may choose a different graph visualization framework.

Listing 3: Covid-19 reaction network in dot-format.

```
digraph {
    s0[label="S",shape="box",style="filled",fillcolor="cyan"];
    s1[label="E",shape="box",style="filled",fillcolor="cyan"];
    s2[label="I",shape="box",style="filled",fillcolor="cyan"];
    s3[label="R",shape="box",style="filled",fillcolor="cyan"];
    s4[label="H",shape="box",style="filled",fillcolor="cyan"];
    r0[label="7.74194e-05",shape="oval",style="filled",fillcolor="yellow"];
    s2 -> r0 [arrowhead="tee"];
    s0 -> r0;
    r0 -> s1;
    r1[label="0.196078",shape="oval",style="filled",fillcolor="yellow"];
    s1 -> r1;
    r1 -> s2;
    r2[label="0.322581",shape="oval",style="filled",fillcolor="yellow"];
    s2 -> r2;
    r2 -> s3;
    r3[label="0.000290061",shape="oval",style="filled",fillcolor="yellow"];
    s2 -> r3;
    r3 -> s4;
    r4[label="0.0988142",shape="oval",style="filled",fillcolor="yellow"];
    s4 -> r4;
    r4 -> s3;
}
```

Listing 4: Circadian rhythm reaction network in dot-format.

```
digraph {
    s0[label="DA",shape="box",style="filled",fillcolor="cyan"];
    s1[label="D_A",shape="box",style="filled",fillcolor="cyan"];
    s2[label="DR",shape="box",style="filled",fillcolor="cyan"];
    s3[label="D_R",shape="box",style="filled",fillcolor="cyan"];
    s4[label="MA",shape="box",style="filled",fillcolor="cyan"];
    s5[label="MR",shape="box",style="filled",fillcolor="cyan"];
    s6[label="A",shape="box",style="filled",fillcolor="cyan"];
    s7[label="R",shape="box",style="filled",fillcolor="cyan"];
    s8[label="C",shape="box",style="filled",fillcolor="cyan"];
    r0[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r0;
    s0 -> r0;
    r0 -> s1;
    r1[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s1 -> r1;
    r1 -> s0;
    r1 -> s6;
    r2[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r2;
    s2 -> r2;
    r2 -> s3;
    r3[label="100",shape="oval",style="filled",fillcolor="yellow"];
    s3 -> r3;
    r3 -> s2;
    r3 -> s6;
    r4[label="500",shape="oval",style="filled",fillcolor="yellow"];
    s1 -> r4;
    r4 -> s4;
    r4 -> s1;
    r5[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s0 -> r5;
    r5 -> s4;
    r5 -> s0;
    r6[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s3 -> r6;
    r6 -> s5;
    r6 -> s3;
    r7[label="0.01",shape="oval",style="filled",fillcolor="yellow"];
    s2 -> r7;
    r7 -> s5;
    r7 -> s2;
    r8[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s4 -> r8;
    r8 -> s4;
    r8 -> s6;
    r9[label="5",shape="oval",style="filled",fillcolor="yellow"];
    s5 -> r9;
    r9 -> s5;
    r9 -> s7;
    r10[label="2",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r10;
    s7 -> r10;
    r10 -> s8;
    r11[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s8 -> r11;
    r11 -> s7;
    r12[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r12;
    r13[label="0.2",shape="oval",style="filled",fillcolor="yellow"];
    s7 -> r13;
    r14[label="10",shape="oval",style="filled",fillcolor="yellow"];
    s4 -> r14;
    r15[label="0.5",shape="oval",style="filled",fillcolor="yellow"];
    s5 -> r15;
}
```

Listing 5: Circadian rhythm with catalysts reaction network in dot-format.

```
digraph {
    s0[label="DA",shape="box",style="filled",fillcolor="cyan"];
    s1[label="D_A",shape="box",style="filled",fillcolor="cyan"];
    s2[label="DR",shape="box",style="filled",fillcolor="cyan"];
    s3[label="D_R",shape="box",style="filled",fillcolor="cyan"];
    s4[label="MA",shape="box",style="filled",fillcolor="cyan"];
    s5[label="MR",shape="box",style="filled",fillcolor="cyan"];
    s6[label="A",shape="box",style="filled",fillcolor="cyan"];
    s7[label="R",shape="box",style="filled",fillcolor="cyan"];
    s8[label="C",shape="box",style="filled",fillcolor="cyan"];
    r0[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r0;
    s0 -> r0;
    r0 -> s1;
    r1[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s1 -> r1;
    r1 -> s0;
    r1 -> s6;
    r2[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s2 -> r2;
    s6 -> r2;
    r2 -> s3;
    r3[label="100",shape="oval",style="filled",fillcolor="yellow"];
    s3 -> r3;
    r3 -> s2;
    r3 -> s6;
    r4[label="500",shape="oval",style="filled",fillcolor="yellow"];
    s1 -> r4 [arrowhead="tee"];
    r4 -> s4;
    r5[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s0 -> r5 [arrowhead="tee"];
    r5 -> s4;
    r6[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s3 -> r6 [arrowhead="tee"];
    r6 -> s5;
    r7[label="0.01",shape="oval",style="filled",fillcolor="yellow"];
    s2 -> r7 [arrowhead="tee"];
    r7 -> s5;
    r8[label="50",shape="oval",style="filled",fillcolor="yellow"];
    s4 -> r8 [arrowhead="tee"];
    r8 -> s6;
    r9[label="5",shape="oval",style="filled",fillcolor="yellow"];
    s5 -> r9 [arrowhead="tee"];
    r9 -> s7;
    r10[label="2",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r10;
    s7 -> r10;
    r10 -> s8;
    r11[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s8 -> r11;
    r11 -> s7;
    r12[label="1",shape="oval",style="filled",fillcolor="yellow"];
    s6 -> r12;
    r13[label="0.2",shape="oval",style="filled",fillcolor="yellow"];
    s7 -> r13;
    r14[label="10",shape="oval",style="filled",fillcolor="yellow"];
    s4 -> r14;
    r15[label="0.5",shape="oval",style="filled",fillcolor="yellow"];
    s5 -> r15;
}
```

Listing 6: Listing style code for typesetting `dot` format in LaTeX.

```
\lstdefinelanguage{dot}{
  keywords={digraph,label,shape,style,fillcolor},
  sensitive=true, string=*[d]{"}
}

\lstdefinestyle{colorDot}{
  language=dot, basicstyle=\ttfamily\scriptsize,
  keywordstyle=\textcolor{blue},
  stringstyle=\slshape\textcolor{red!70!black},
  commentstyle=\slshape\textcolor{green!50!black},
  morecomment=[s][\bfseries\slshape\textcolor{green!50!black}]{/**}{*/},
  tabsize=4,
  showstringspaces=false,
  breaklines=true, breakatwhitespace=true,
  prebreak={\hbox{\quad\textcolor{red}{$\rhookswarrow$}}},
  postbreak={\hbox{\textcolor{red}{$\lhookrightarrow$}}},
  breakindent={-8pt}, breakautoindent=false, % numbers=left, numberstyle=\tiny,
  frameshape={RYR}{N}{N}{YYY} % frame=tb,frameround=tttt
}
```

D Semantics of Rules

This section explains formally the meaning of the reaction rules.

For k, ℓ, m, n positive integer counts, R_i positive integer count of species i and t positive real-valued time:

$$\frac{k\mathsf{A} + \ell\mathsf{B} \xrightarrow{\lambda} m\mathsf{C} + n\mathsf{D},\ \langle Q_i \mid Q_\mathsf{A} \geq k \ \wedge\ Q_\mathsf{B} \geq \ell \rangle}{\mathrm{Prob}(\langle Q_\mathsf{A}, Q_\mathsf{B}, Q_\mathsf{C}, Q_D \rangle \longrightarrow \langle Q_\mathsf{A} - k, Q_\mathsf{B} - \ell, Q_\mathsf{C} + m, Q_\mathsf{D} + n \rangle, t) = \mathrm{PDF}_{\exp(\lambda \cdot Q_\mathsf{A} \cdot Q_\mathsf{B})}(t)}$$

Given

1. the reaction *rule* where k count of A meet ℓ count of B with a rate of λ and produce m count of C and n count of D, and
2. the *system state* $\langle Q_\mathsf{A}, Q_\mathsf{B}, Q_\mathsf{C}, Q_\mathsf{D} \rangle$ with enough input agent counts: $Q_\mathsf{A} \geq k$ and $Q_\mathsf{B} \geq \ell$,

Then

- the probability of a transition from state $\langle Q_\mathsf{A}, Q_\mathsf{B}, Q_\mathsf{C}, Q_D \rangle$ to $\langle Q_\mathsf{A} - k, Q_\mathsf{B} - \ell, Q_\mathsf{C} + m, Q_\mathsf{D} + n \rangle$ after time delay t is equal to the probability density function of the exponential distribution of rate $(\lambda \cdot Q_\mathsf{A} \cdot Q_\mathsf{B})$ at t.