Stochastic Simulation Library
Assignment for Selected Topics in Programming

Marius Mikučionis

May 26, 2023

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 system. Curiously, such systems exhibit emergent behavior: the 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 different simulations may exhibit different behaviors, therefore statistical methods are needed to understand the expected behavioral effects.

A stochastic system can be described by a number of agents from 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, 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 (the rate is zero). 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 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.

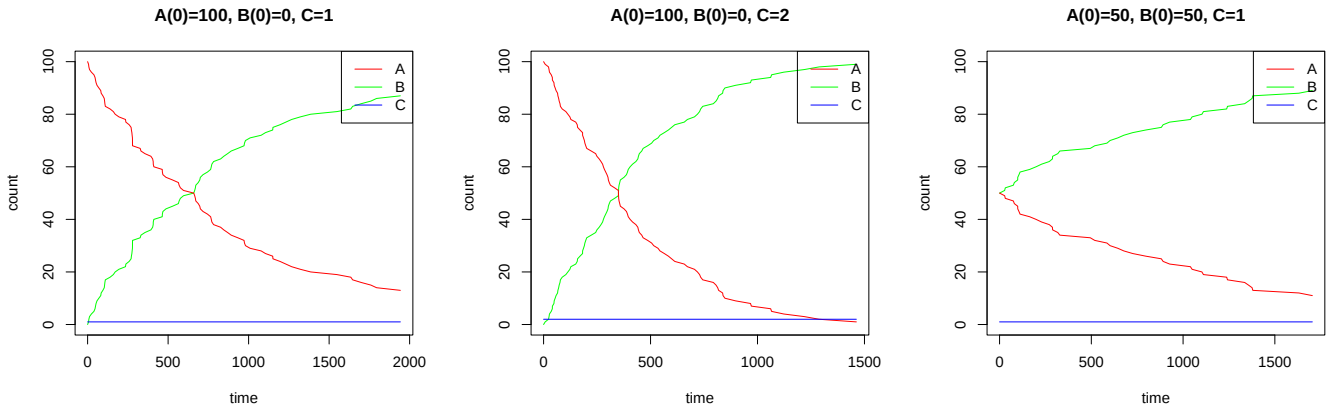

Figure 1: Sample trajectories of A, B and C quantities over time given $\lambda = 0.001$.

The reaction delay (time between reaction events) is stochastic: it follows an exponential distribution with the specific (exponential) reaction rate as computed above. If there are multiple reactions available, 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, a hydrogen molecule reacts with a chlorine molecule to produce two molecules of hydrogen chloride ($\mathsf{H_2 + Cl_2} \xrightarrow{\lambda} \mathsf{2HCl}$). If R_H is the amount of hydrogen, R_{Cl} is the amount of chlorine, and P_{HCl} is the amount of hydrogen chloride, then the reaction step is simulated by subtracting $\Delta_{R_H} = 1$ from R_H and $\Delta_{R_{Cl}} = 1$ from R_{Cl} as well as adding $\Delta_{P_{HCl}} = 2$ to P_{HCl}. As explained above, the speed of the reaction depends on the amount of reactants. If these amounts decrease, the reaction slows down. This is simulated by increasing the time delay to the next reaction step. The delay between the reaction steps k and $k+1$ is stochastic and is generated as follows:

```
   Input  : Set of reactions S, end time T
1  State ⟨R_i⟩ with reactant amounts R_i;
2  while t ≤ T do
3  |  foreach r ∈ S do Compute r.delay;
4  |  r := argmin_{r∈S} r.delay;
5  |  t := t + r.delay;
6  |  if ∀Δ_{R_i} ∈ r : Δ_{R_i} ≥ R_i ∧ ∀Δ_{C_i} ∈ r : Δ_{C_i} ≤ C_i
7  |   then
8  |  |  foreach Δ_{R_i} ∈ r do R_i = R_i − Δ_{R_i};
9  |  |  foreach Δ_{P_i} ∈ r do P_i = P_i + Δ_{P_i};
10 |  end
11 |  Print/save/monitor the state ⟨R_i⟩;
12 end
```

Algorithm 1: Simulation of multiple reactions

$$delay_k = Exp(\lambda_k), \text{ where } \lambda_k = \lambda \cdot \prod_i R_{i,k} \tag{1}$$

Here, $Exp(\lambda_k)$ is a random number distributed according to the exponential distribution with the rate parameter λ_k (see `std::exponential_distribution`), λ is the intrinsic rate of the reaction as described above, $R_{i,k}$ is the amount of the i-th reactant at step k. After $delay_k$ time units, the reaction step $k+1$ is performed by reducing the amounts of reactants ($R_{i,k+1} = R_{i,k} - \Delta_{R_i}$) and increasing the amounts of reaction products ($P_{i,k+1} = P_{i,k} + \Delta_{P_i}$) (lines 8–9 in the pseudocode), as well as computing the new $delay_{k+1}$ (line 3). This can be repeated as long as there are necessary amounts of reactants ($\forall i : \Delta_{R_i} \geq R_{i,k}$) (line 6).

The pseudocode shows how several simultaneous reactions are simulated. The algorithm maintains a set of reactions and the state of the system as the amounts of all the species of reactants. It always picks a reaction with the smallest delay (line 4). The reactants of one reaction can be the products of another and vice versa. That is why the delays of all reactions are recomputed in line 3 first. Note that the exponential distribution is "memory-less" and can be recomputed at any time, even for reactions that were not executed. A more efficient implementation could recompute only the delays of the reactions which were affected in the previous iteration.

2 Requirements

The goal of the assignment is to develop a library for simulating reactions. The usage of the library should be demonstrated on three example stochastic systems. The library implementation should support the following features:

1. Use *operator overloading* to support the reaction rule typesetting directly in C++ code.
2. *Pretty print* the reaction network in a) human readable format and b) network graph (e.g. Figure 4).
3. Implement a ***generic*** *symbol table* to *store* and *lookup* objects of user defined types (e.g. information about agents and reactions). Support failure cases when the table does not contain a looked up value.
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 the state monitor* in the stochastic simulation algorithm. Use it to estimate the peak of hospitalized agents in Covid-19 example without storing trajectory data for 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 (mean) value of the hospitalized peak over 20 simulations.
9. Implement *unit tests* (e.g. test symbol table methods and pretty printing of reaction rules).
10. *Benchmark* and compare the stochastic simulation performance (e.g. the time it takes to compute 20 simulations a single core, multiple cores, or improved implementation). Make your conclusions.

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

A Example Client 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. Tasks: compute and plot the average of C, A and R signals over time by sampling 100 trajectories.

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

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

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

$\alpha_A = 50\,\mathrm{h}^{-1}$	$\delta_{M_R} = 0.5\,\mathrm{h}^{-1}$
$\alpha'_A = 500\,\mathrm{h}^{-1}$	$\theta_A = 50\,\mathrm{h}^{-1}$
$\alpha_R = 0.01\,\mathrm{h}^{-1}$	$\theta_R = 100\,\mathrm{h}^{-1}$
$\alpha'_R = 50\,\mathrm{h}^{-1}$	$\mathsf{D}_A = 1\mathrm{m}$
$\beta_A = 50\,\mathrm{h}^{-1}$	$\mathsf{D}'_A = 0$
$\beta_R = 5\,\mathrm{h}^{-1}$	$\mathsf{D}_R = 1\mathrm{m}$
$\gamma_A = 1\,\mathrm{m}^{-1}\mathrm{h}^{-1}$	$\mathsf{D}'_R = 0$
$\gamma_R = 1\,\mathrm{m}^{-1}\mathrm{h}^{-1}$	$\mathsf{M}_A = 0$
$\gamma_C = 2\,\mathrm{m}^{-1}\mathrm{h}^{-1}$	$\mathsf{M}_R = 0$
$\delta_A = 1\,\mathrm{h}^{-1}$	$\mathsf{A} = 0$
$\delta_R = 0.2\,\mathrm{h}^{-1}$	$\mathsf{R} = 0$
$\delta_{M_A} = 10\,\mathrm{h}^{-1}$	$\mathsf{C} = 0$

Listing 1: Circadian oscillator in C++

```
vessel_t circadian_oscillator()
{
    auto alphaA = 50.0;
    auto alpha_A = 500.0;
    auto alphaR = 0.01;
    auto alpha_R = 50.0;
    auto betaA = 50.0;
    auto betaR = 5.0;
    auto gammaA = 1.0;
    auto gammaR = 1.0;
    auto gammaC = 2.0;
    auto deltaA = 1.0;
    auto deltaR = 0.2;
    auto deltaMA = 10.0;
    auto deltaMR = 0.5;
    auto thetaA = 50.0;
    auto thetaR = 100.0;
    auto v = vessel_t{};
    auto env = v.environment();
    auto DA = v("DA", 1);
    auto D_A = v("D_A", 0);
    auto DR = v("DR", 1);
    auto D_R = v("D_R", 0);
    auto MA = v("MA", 0);
    auto MR = v("MR", 0);
    auto A = v("A", 0);
    auto R = v("R", 0);
    auto C = v("C", 0);
    v(A + DA >>= D_A, gammaA);
    v(D_A >>= DA + A, thetaA);
    v(A + DR >>= D_R, gammaR);
    v(D_R >>= DR + A, thetaR);
    v(D_A >>= MA + D_A, alpha_A);
    v(DA >>= MA + DA, alphaA);
    v(D_R >>= MR + D_R, alpha_R);
    v(DR >>= MR + DR, alphaR);
    v(MA >>= MA + A, betaA);
    v(MR >>= MR + R, betaR);
    v(A + R >>= C, gammaC);
    v(C >>= R, deltaA);
    v(A >>= env, deltaA);
    v(R >>= env, deltaR);
    v(MA >>= env, deltaMA);
    v(MR >>= env, deltaMR);
    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

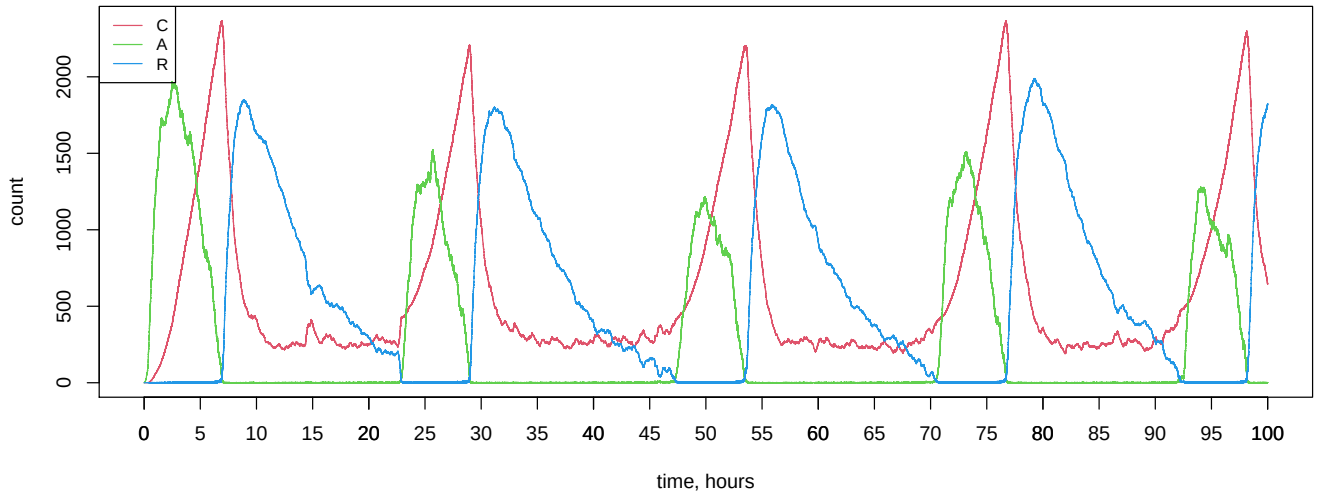

Figure 2: Sample trajectory of circadian rhythm: agent counts repeat with ≈ 24 hour periods.

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".

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

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

Listing 2: Covid-19 model in C++

```
vessel_t seihr(uint32_t N)
{
    auto v = vessel_t{};
    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; // basic reproductive number (initial, without lockdown etc)
    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; // recovery/death rate in hospital (H -> R) ~10.12 days
    auto S = v("S", S0); // susceptible
    auto E = v("E", E0); // exposed
    auto I = v("I", I0); // infectious
    auto H = v("H", 0);  // hospitalized
    auto R = v("R", 0);  // removed/immune (recovered + dead)
    v(S+I >>= E+I, beta/N); // susceptible becomes exposed through infectious
    v(E >>= I, alpha); // exposed becomes infectious
    v(I >>= R, gamma); // infectious becomes removed
```

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

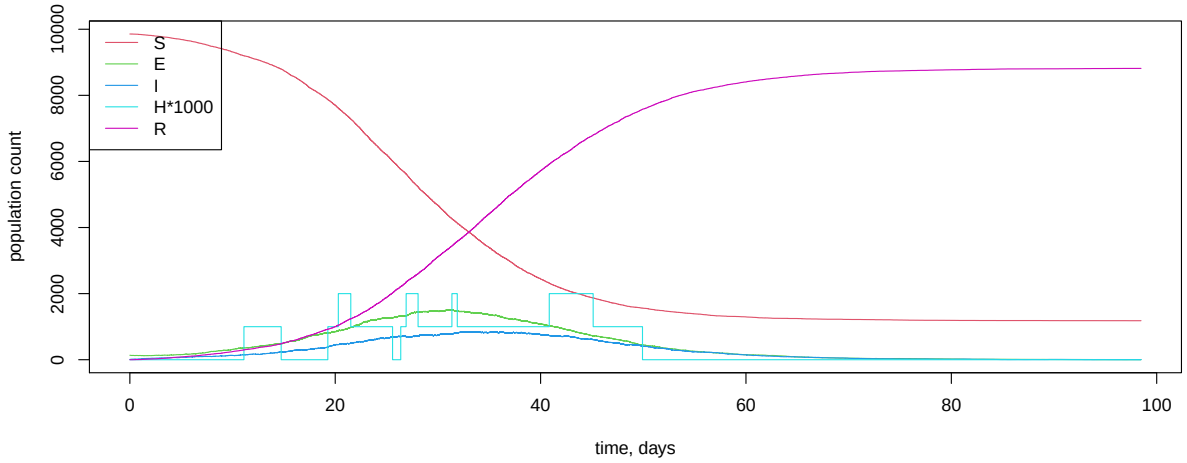

Figure 3: Sample trajectory of Covid-19 population $\mathsf{N} = 10000$: hospitalizations peak at 2.

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*.

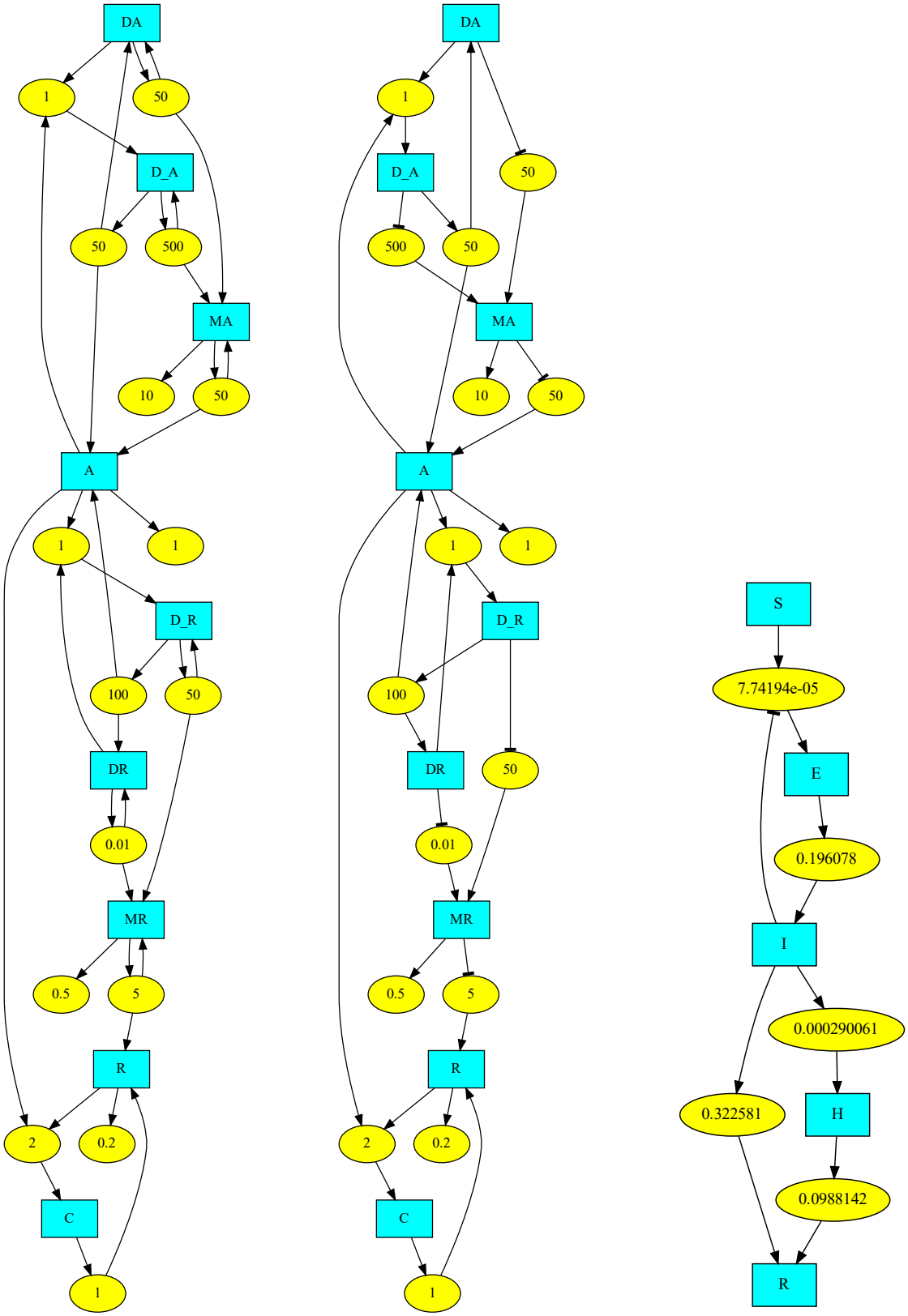

Figure 4: Layout of reaction networks: circadian model, circadian with catalysts, Covid-19 from Listings 3,4,5. The images are rendered using command-line `dot` utility from Graphiz.

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
}
```