Individual.jl: Rewriting individual-based models for epidemiology using graph rewriting

Sean L. Wu
Institute for Health Metrics and Evaluation,
University of Washington,
Seattle, WA, USA
seanwu89@uw.edu

Sophie Libkind
Stanford University,
Palo Alto, CA, USA
slibkind@stanford.edu

Kristopher Brown
University of Florida,
Gainesville, FL, USA
kristopher.brown@ufl.edu

Evan Patterson
Topos Institute,
Berkeley, CA, USA
evan@topos.institute

James Fairbanks
University of Florida,
Gainesville, FL, USA
fairbanksj@ufl.edu

ABSTRACT

Modellers in epidemiology and ecology seeking to implement *individual-based models* (IBMs) can choose from a variety of software libraries, but nearly all suffer from common shortcomings. These include complex inheritance hierarchies, dissociation between source code and model structure, and inseparability of an event's state update rules and probability computation. We present Individual.jl, a Julia library that relies on Catlab.jl to specify and simulate IBMs so that model state and dynamics are transparent in their implementation and grounded in the mathematics of applied category theory.

1 MOTIVATION

Here we define IBM to mean a model which does not represent individuals as a density or other continuous quantity, and therefore does not have a system of differential equations as its natural mathematical description (although many may admit such a description for its Kolmogorov equations). In an IBM, the state of the world is represented by a set of *individuals* along with their attributes and relationships. Dynamics are specified by *events*, each of which has a function computing (1) its probability to *fire* (i.e. occur) given the current world state and (2) a state update rule.

Most IBM software is based on object-oriented programming (OOP) principles. Common generic software include [15, 10, 9, 13] and [1] is a successful library for epidemiology. In OOP based systems, users create a class for simulated individuals, which may fit into a complex inheritance hierarchy. Classes are often tightly coupled, and require expensive integration tests to ensure correctness after modifications. Inheritance hierarchies make it difficult to instantiate arbitrary or multiple types of agents (e.g. birds and humans) in a single model. Additionally, model structure is embedded in the source code rather than expressed as *data*, so tasks such as querying state require specialized design patterns.

Specification of events in standard software also suffers from tight coupling of update rules, computation of firing probability, and sampling. This means that the only way to interact with a model is to draw trajectories and statistically analyze results. In addition, there are generally no formal restrictions on what aspects of state may be updated by an event; a source of hard to diagnose bugs, especially for models which are updated to reflect changing real-world conditions. Conversely, some platforms exhibit the opposite problem where users are restricted to select from a predefined set of state transitions and probability computations [5, 7].

While not designed for epidemiological models, the Kappa software for biochemical simulation [2] based on graph rewriting is the closest in spirit to our approach, although [14] identified several domain-specific design choices which cause difficulty for epidemiological models.

2 DESIGN AND IMPLEMENTATION

Individual.jl is a library that relies on the graph rewriting capabilities of Catlab.jl [3] for specification and simulation of discrete-time IBMs. To create an IBM in Individual.jl, the user defines a schema, which completely describes the simulated world, including types of individuals, attributes of individuals, and relations between individuals. Computation is performed on efficient data structures called Attributed C-Sets (ACSets), which are programmatically generated from a schema [11]. For example, the event depicted in Figure 1 applies to the model whose schema is

$$\begin{tikzcd}
\mathcal{S} \arrow[dr] & \mathcal{I} \arrow[d] & \mathcal{R} \arrow[dl] \\
& \mathcal{A} &
\end{tikzcd}$$

and a world-state in this model is a functor G from the free category on this schema into FinSet. $G(\mathcal{S}), G(\mathcal{I}), G(\mathcal{R})$ are sets of susceptible, infected, and recovered persons. $G(\mathcal{A})$ is a set of persons. The morphisms specify that each disease state is assigned to a person. This schema could be expanded to include attributes such as each person's age.

Here we list some advantages of encoding model structure as an ACSet. (1) The schema provides constraints on allowed world states such that an invalid state cannot be sampled during a simulation run. (2) Adapting a model, by adding a type or attribute of individuals, is an unambiguous, local process. Furthermore, using functorial data migration, events defined on one model can be automatically reinterpreted as events on the adjusted model. (3) In OOP software, querying model state is often expensive, involving applying a predicate to every individual. In contrast, querying ACSets takes advantage of their efficient database structure. For more complex queries, Catlab.jl supports disjoint unions of conjunctive queries (duc queries) [12] on models.

Model dynamics are defined by a set of events specified as graph rewrite rules, each of which describes its preconditions and how state will change if it fires. Currently we use the double-pushout (DPO) approach which encodes this information in a span of ACSets $L \hookleftarrow I \rightarrow R$ (Figure 1). A match morphism m from preconditions L into world-state G indicates a possible application of the rule. Users also define a function which computes the probability of firing. This probability is used to sample which possible applications are scheduled during that time-step, and scheduled rewrites are applied at the end of the time-step. The clock then advances and the process continues. Time steps may be of arbitrary size, and are used for computing firing probabilities (e.g. via an exponential distribution function).

Figure 1 shows an example of an infection rewrite rule for an SIR (Susceptible-Infectious-Recovered) model. A sampled trajectory is shown in Figure 2.

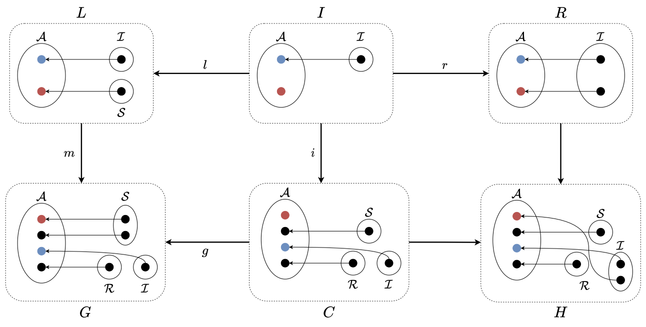

Figure 1: SIR infection update with DPO rewriting. G is the current world state, which is updated to become H by applying the rule $L \hookleftarrow I \rightarrow R$. $\mathcal{A}$ is the set of individuals, and $\mathcal{S}, \mathcal{I}, \mathcal{R}$ refer to those states of the SIR model. Coloring indicates the data of the ACSet homomorphisms.

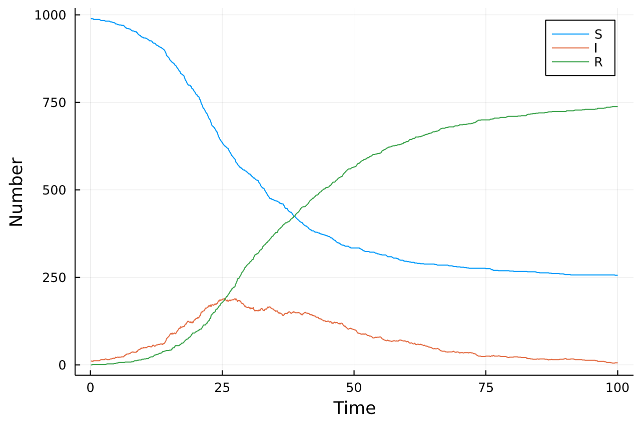

Figure 2: Sampled epidemic trajectory

Events which fire after a non-geometric delay are also accommodated. If a match m into world-state G_t is identified at time t, it can be scheduled for $t + n$ by modifying m to point to G_{t+n}. The *partial* morphisms between each intervening model update can be post-composed with m to yield a match into G_{t+n}. If the scheduled match was interrupted prior to $t + n$ then no composition exists and m is no longer valid. For example, if an individual dies before being discharged from hospital, the discharge event will be properly thrown out without the user needing to manually cancel it within the death event, as common in existing platforms.

3 DISCUSSION AND FURTHER WORK

Using schemas to define the simulated world, ACSets to store data, and graph rewriting rules to specify dynamics has benefits for practical modeling. Ill formed rewrite rules such as those with dangling edges or non-injunctive matches will simply not be possible to apply. These guard rails alleviate one source of bugs and reduce the need for expensive integration testing. Resulting code is smaller, more maintainable, and amenable to unit testing, an increasing concern for complex infectious disease models [8].

Separation of state update rules and probability computation means that users can interact with models beyond sampling trajectories. As in Kappa, causal dependence of particular states upon certain chains of events could be identified [4]. For policy evaluation this allows comparison of an intervened upon system to its exact counterfactual, rather than statistical comparison of many independent trajectories. This was considered for SIR type epidemics in [6], but required laborious manual construction of model specific graphs. Our framework can enable counterfactual analysis for any expressible model.

Further development can target specific challenges for epidemiology as identified by [14]. Network models can be easily handled, indeed Catlab.jl already includes schemas for directed multigraphs. Secondly, spatial structure can be elegantly implemented. Basic development into efficient ways to cache identified matches and restricting morphism searching into updated parts of state are important areas for performance optimization.

4 AUTHOR CONTRIBUTIONS

S.L.W designed the simulation framework, wrote the manuscript, and developed examples. K.B developed graph rewriting tools for ACSets. K.B and S.L revised the manuscript, and developed the mathematics. E.P and J.F supervised the project.

REFERENCES

[1] Anna Bershteyn et al. “Implementation and applications of EMOD, an individual-based multi-disease modeling platform”. In: *Pathogens and disease* 76.5 (2018), fty059.

[2] Pierre Boutillier et al. “The Kappa platform for rule-based modeling”. In: *Bioinformatics* 34.13 (2018), pp. i583–i592.

[3] Kristopher Brown, Evan Patterson, and James Fairbanks. “Double pushout rewriting of C-sets”. In: *arXiv preprint arXiv:2111.03784* (2021).

[4] Vincent Danos et al. “Graphs, rewriting and pathway reconstruction for rule-based models”. In: *FSTTCS 2012-IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science.* Vol. 18. 2012, pp. 276–288.

[5] Thomas Hladish et al. “EpiFire: An open source C++ library and application for contact network epidemiology”. In: *BMC bioinformatics* 13.1 (2012), pp. 1–12. DOI: 10.1186/1471-2105-13-76.

[6] Joshua Kaminsky et al. “Perfect counterfactuals for epidemic simulations”. In: *Philosophical Transactions of the Royal Society B* 374.1776 (2019), p. 20180279.

[7] Jori Liesenborgs et al. “SimpactCyan 1.0: An Open-source Simulator for Individual-Based Models in HIV Epidemiology with R and Python Interfaces”. In: *Scientific reports* 9.1 (2019), pp. 1–13. DOI: 10.1101/440834.

[8] Tim CD Lucas et al. “Responsible modelling: unit testing for infectious disease epidemiology”. In: *Epidemics* 33 (2020), p. 100425.

[9] David Masad and Jacqueline Kazil. "MESA: an agent-based modeling framework". In: *14th PYTHON in Science Conference*. Citeseer. 2015, pp. 53–60. doi: 10.25080/majora-7b98e3ed-009.

[10] Michael J North et al. "Complex adaptive systems modeling with Repast Simphony". In: *Complex adaptive systems modeling* 1.1 (2013), pp. 1–26. doi: 10.1186/2194-3206-1-3.

[11] Evan Patterson, Owen Lynch, and James Fairbanks. "Categorical Data Structures for Technical Computing". In: *arXiv preprint arXiv:2106.04703* (2021).

[12] David I Spivak. "Functorial aggregation". In: *arXiv preprint arXiv:2111.10968* (2021).

[13] Ali R Vahdati. "Agents. jl: agent-based modeling framework in Julia". In: *Journal of Open Source Software* 4.42 (2019), p. 1611.

[14] William Waites et al. "Rule-based epidemic models". In: *Journal of theoretical biology* 530 (2021), p. 110851.

[15] Uri Wilensky. *NetLogo*. http://ccl.northwestern.edu/netlogo/. Northwestern University, Evanston, IL: Center for Connected Learning and Computer-Based Modeling, 1999. url: http://ccl.northwestern.edu/netlogo/.