

Harnessing Probabilistic Logic Programming for the Transparent Local Modelling of Infectious Diseases

Felix Weitkämper^{1,*}, Ameen Almiftah¹ and Kailin Sun²

¹Institut für Informatik der LMU München, Oettingenstr. 67, 80538 München, Germany

²Department Biologie I der LMU München, Menzinger Str. 67, 80638 München, Germany

Abstract

Public health and infectious disease modelling is an area where interpretable artificial intelligent methods can have a profound impact. We believe that probabilistic logic programming in Problog is ideal for transparent and verifiable epidemiological modelling which is also adaptable in the face of an ongoing pandemic. As a proof of concept, our SimPLoID system showcases this potential within a network-based approach, which facilitates public discourse on behavioural responses at the local and individual level. This application area benefits especially from the transparency and precisely specified semantics of the Problog language, which are supported by well-maintained engines for Monte Carlo simulation and a variety of other learning and reasoning tasks.

1. Introduction

There have been some huge landmark events occurring in the early 2020s. The release of ChatGPT in November 2022 marked the start of a rapid upsurge in public interest in artificial intelligence. These technologies have heralded in a new era in the way we work and study, and have caused considerable concern about the power of artificial intelligence for good or evil. Writing in 2024, the far-reaching impacts of the COVID-19 pandemic still weigh heavy upon the public psyche. This global phenomenon showed the world how essential policy-making is in the field of public health and medicine. We believe that artificial intelligence can be harnessed for the benefit of public health understanding.

The early days of the COVID-19 pandemic illustrated the pitfalls of epidemiological modelling in the public eye. In the first months of the spread of the disease in Europe, experts came to wildly varying conclusions about the disease dynamics [1, 2, 3]. One reason why these early models were unreliable was because of the lack of quality data. As an emerging infectious disease, information was constantly being updated during the course of the pandemic. This shows that the ability to quickly integrate newly available data into an epidemiological model is essential.

The modelling results from the scientific community had a direct impact on national and international politics. In the face of uncertainty, governments throughout the world responded very differently, some implementing a total lockdown, others hoping that "herd immunity" would quickly be reached [4]. Difficult decisions also

had impacts on the local level, as policies like lockdowns affected everyday life. Individuals or local institutions must make behavioural decisions to minimise the spread of disease. On a small scale, network-based models are particularly helpful, as they model interactions between individuals in a closed environment. This will help people understand the interactions of their choices and the choices of those around them, their immediate effects locally and how this in turn impacts the bigger picture.

Here, we present a network-based, transparent, verifiable and adaptable solution for infectious disease modelling. The high stakes on human health, economic policies, and societal pressures during the pandemic highlighted the importance of modelling processes in political, local, and individual decision making, as well as public understanding of epidemiology. We hope that our contribution is a step towards this goal.

Infectious Disease Modelling

A classical approach to epidemiological modelling is via compartmental models such as Kermack and McKendrick's Susceptible-Infected-Recovered (SIR) model, which has been used successfully to model many human, animal and plant diseases [5]. This model divides the population into the three different stages of disease infection [6], summarised diagrammatically in Figure 1. The proportion of the population moving from being susceptible to infected to recovered in each time step is modelled by differential equations. The classic SIR model relies on some assumptions which mean it may not be the best model for all infectious diseases. For instance, a key assumption is a well-mixed population, the violation of which during the COVID-19 pandemic led to a significant deviation from the theoretical distribution of the illness [8]. Just as importantly, SIR models do not allow the behaviour of individuals to be encoded, nor do they

2nd Workshop on 'Public Interest AI' co-located with the 47th German Conference on AI (KI 2024), September 23, 2024, Julius-Maximilians Universität Würzburg, Germany

*Corresponding author.



Copyright © 2024 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

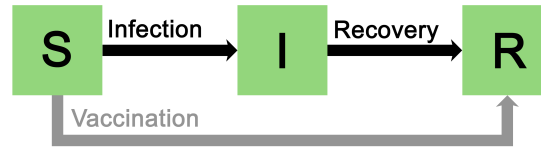


Figure 1: The classic SIR model, showing Susceptible, Infected and Recovered groups with their modes of transition [7], CC-BY 4.0.

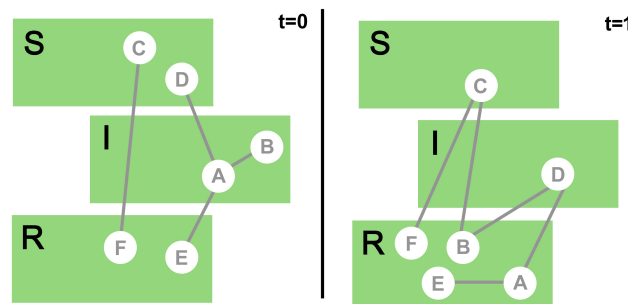


Figure 2: An example of a network-based model. Circles represent individuals, which may be either susceptible, infected or recovered. Lines represent connections between individuals. Across different timesteps, both the compartments of the individuals and the connections between them can shift.

allow predictions on the health status of given individuals. Given the huge importance of individual behaviour on public policy, there is a clear need for localised modelling that can explicitly account for individual behaviour and contact patterns.

These limitations are addressed by network-based models, which utilise the fundamental ideas behind SIR models, but applies them to a contact graph. Here, nodes represent individuals and edges show the connection between individuals, which individually are in one of the compartments of the classical model. As connections between individuals can change over time, and through hospitalisation, quarantine or other measures even in direct response to illness, it is important for the underlying connection graph to allow for changes over time.

In this contribution we explore the untapped potential of probabilistic logic programming for transparent and

scrutable epidemiological modelling at local scales. More specifically, the particularly simple yet expressive probabilistic logic programming language ProbLog is ideal for these purposes, because the parameters and underlying assumptions of the model are transparent and immediately accessible [9]. ProbLog is implemented by the well-maintained ProbLog 2 engine [10], which allows not only for the calculation of exact probabilities, but also for simulations to be generated from a specified model [11]; these useful features allow for the development of a versatile and extendable epidemiological tool.

Our exploration is set in the the context of our prototype implementation SimPloID, short for Simulations with Probabilistic Logic for Infectious Diseases. SimPloID aims to be a flexible and transparent framework for network-based epidemiological modelling that is easy to use for researchers, policy-makers and science communicators who are not trained in

programming. Additionally, the models generated in the SimPLoID framework are designed to represent individual events that can be combined modularly and individually critiques or justified. However, the current implementation is to be regarded as a proof of concept, showcasing the potential of probabilistic logic programming for disease epidemiology, rather than as production software. Its source code is available at <https://gitlab.com/jona5/epidemical-dsl-update/-/tree/main>, but all its features should still be regarded as subject to change without notice. An indication of what could be added in future work is given in Section 4. In addition to visualisation capabilities and a command-line user interface, SimPLoID encompasses a domain-specific language (DSL) to specify disease models more succinctly than in pure ProLog. However, since the design of the DSL is still in flux, this paper will focus on the ProLog representation itself.

2. Related Work

Our contribution is a modelling framework that is based on probabilistic logic programming. It serves as a proof of concept for an idea first presented by Weitkämper et al. [7]. Of course, given the importance of epidemiological modelling, a variety of tools and packages are available for this purpose. Most of them are written in imperative languages, and do not allow transparent access to modelling parameters beyond the user’s original settings. They also lack a clear underlying semantic framework.

However, there are exceptions to this pattern, of which we name the closest ones to our approach that we are aware of. There are a number of declarative frameworks for compartmental modelling, which also support far more nuanced approaches than the classical SIR model. Two particular exponents of this direction are Kendrick [12], whose domain-specific language is compiled to statements in an imperative host language, and rule-based models in dedicated rule specification languages such as κ [13]. While such aggregate approaches can capture subtleties beyond pure compartmental models, they still target larger scale simulations and are unable to make use of individual-level network data. Other approaches rely on functional reactive programming in Haskell as a base formalism for declarative modelling [14, 15]. While declarative, functional reactive programming is not inherently probabilistic, and therefore verifying the probabilistic assumptions underlying a simulation is still difficult. Their implementation depends on explicit signal generating functions, and the resulting programs are fully-fledged Haskell programs rather than accessible probabilistic models such as the probabilistic logic programs we invoke here.

The approach perhaps closest to our own is that of EMULSION [16], which supports both compartmental and network-based approaches. Features of EMULSION include a graphical user interface for simulations as well as statistical tools for analysing simulations. EMULSION compiles the model file, written in its purpose-built domain-specific language, into a set of finite state machines, one for every modelled quantity. In a network-based model, this means a single finite state machine for every individual in the simulation. As this concept is closest to our approach, we discuss the differences between probabilistic logic programs and finite state machines as compilation targets in Section 4.

3. System Description

Models and queries are processed in two steps. First, a model file, written in a dedicated DSL, is parsed and compiled into ProLog clauses. Model files contain simple statements about the disease under investigation. A model file can be supplemented by default settings, which are contained in a default file, to avoid the strain of repetitive specifications. Individuals and their network of contacts can either be generated randomly, triggered by a corresponding statement in the model file, or they are specified as an additional file and loaded directly into ProLog 2.

The ProLog language allows for succinct sets of probabilistic rules which match natural human understanding quite closely. This is especially true when making use of the various language extensions supported by the ProLog 2 engine, in particular the support for inhibition effects [17]. A detailed discussion of ProLog and the capabilities and implementation of the ProLog 2 system can be found in the comprehensive paper by Fierens et al. [10].

Here, we illustrate the ProLog representation using an example. A simple network-based SIR model is given by Listing 1. We see here that we can load persons and airborne contacts from CSV files, rather than crowding our program by having to assert them explicitly. The relevant time points are provided using the Prolog builtin `between/3`, which is also available in ProLog 2, as is the basic Prolog arithmetic used in the remaining clauses. The clause at Line 7 asserts that being susceptible is the default value, which holds for all persons and time points unless there is a reason for it not to hold. Lines 9 and 21 then give the exceptions to this rule: Whenever an individual is either resistant or currently infected, they are not susceptible. Line 11 models the extrinsic rate of infection, where there is a 0.1 chance at any timestep for an individual to be infected from a cause outside of the system we are modelling. The clause at Line 14 models the intrinsic infection rate caused by airborne

Listing 1: ProbLog code for a flu model with SIR compartments

```

:- use_module(library(db)).
:- csv_load('individualsList.csv', 'person').
:- csv_load('contactList.csv', 'airborne_contact').

time(N) :-
    between(1,12,N).
flu__susceptible(X,N) :-
    time(N), person(X).
\+ flu__susceptible(X,N) :-
    time(N), flu__infected(X,N).
0.1::flu__infected(X,M) :-
    time(M), N is M-1, \+ flu__infected(X,N),
    flu__susceptible(X,N).
0.8::flu__infected(X,M) :-
    time(M), N is M-1, airborne_contact(X,Y,N),
    flu__susceptible(X,N), flu__infected(Y,N).
flu__infected(X,M) :-
    time(M), N is M-1, flu__infected(X,N).
\+ flu__infected(X,M) :-
    time(M), N is M-7, flu__infected(X,N).
\+ flu__susceptible(X,N) :-
    time(N), flu__resistant(X,N).
flu__recovered(X,M) :-
    time(M), N is M-1, flu__infected(X,N),
    \+ flu__infected(X,M).
0.9::flu__resistant(X,N) :-
    time(N), flu__recovered(X,N).
flu__resistant(X,M) :-
    time(M), N is M-1, flu__resistant(X,N).

query(flu__susceptible(X,N)).
query(flu__infected(X,N)).
query(flu__recovered(X,N)).
query(flu__resistant(X,N)).

```

contact between a susceptible and an infected individual. Line 17 models that in principle, if someone is ill in one timestep they will still be ill in the next timestep. Line 19 then models recovery after a fixed time of infection, the inhibitor with the negated head again "overruling" the previous clause. Note that a stochastic duration of illness could also be modelled by replacing these two clauses with a single probabilistic clause. Line 23 explains that an individual is recovered if they had been ill in the previous timestep but are no longer ill in the current timestep. Line 26 asserts that there is a 0.9 chance of obtaining immunity upon recovery, and Line 28 ensures that immunity is permanent.

Observe in particular the modularity of the ProbLog code: Every possible cause for infection can be added and removed separately, without any alteration to the

remainder of the program. Those causes are then treated as independent possible triggers for the event specified by the head. For instance, if an individual has contact with two infected individuals in a given timestep, the program above will calculate three separate and independent grounds for infection, the two contacts and the baseline rate, resulting in an overall infection probability of

$$1 - (1 - 0.1)(1 - 0.8)(1 - 0.8) = 0.964$$

according to the laws of probability.

The query goals in the final four lines instruct ProbLog 2 to simulate susceptible, infected, recovered and resistant individuals. Note that had fewer goals been queried, the simulation engine would still have had to simulate almost all of the other ground atoms since they are all mutually connected through the program clauses.

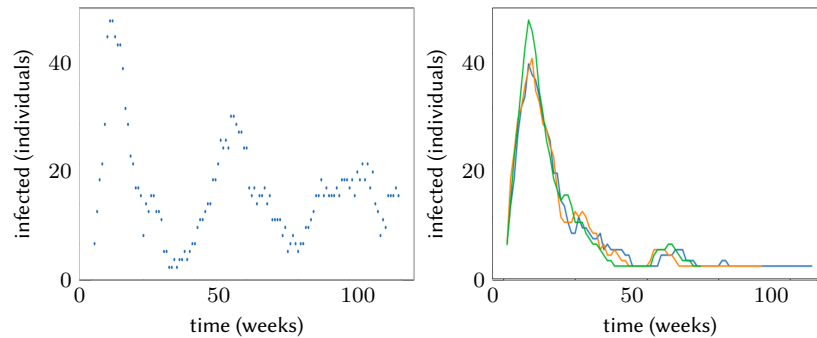


Figure 3: SimPLoID output graphs showing the progress of the number of infected individuals over time from different settings (scatter plot vs line graph, 1 vs 5 simulations, short vs long period of resistance)

Thus there is little efficiency gain expected from querying only for, say, the infected individuals. These clauses are then passed to the ProbLog 2 system, whose Monte Carlo engine [11] then generates one or more simulation runs. Their results in turn can be processed by various Python utilities for graphing and tabulating the output. In our application, we use the matplotlib package [18] for graphing.

To further enhance usability, the different functions of our application can be accessed from an interactive shell, which also provides help and documentation on the different available options. Several well-known disease models have been implemented using the system, including different transmission forms, vaccination regimes and maternally-derived immunity.

It is also possible to include further information such as the number of simulation runs or whether individuals and their contacts should be generated automatically or loaded from a file, and which compartments should be queried. However, these metaparameters can also be specified in the interactive shell, which has been found preferable in most cases.

After running the simulation, the results of the individual runs can be graphed or tabulated, either collectively or individually. Figure 3 shows different ways of displaying the simulation outcomes, with different periods of immunity.

The ProbLog code generated by the current implementation is still in parts more verbose than the code example given in Listing 1, but it follows closely the structure illustrated there. The main deviation is that for performance reasons, our implementation explicitly grounds out the time variable in a preprocessing stage since the general grounding algorithm in ProbLog proved to be an efficiency bottleneck for larger simulations.

4. Discussion

The declarative, intrinsically probabilistic nature of ProbLog specifications can be a huge asset. A ProbLog program is made up modularly of individual clauses, which have an interpretation as independent, individual events. This makes them understandable and open to scrutiny individually. Thus, criticism or justification of a model can be reduced to criticism of individual claims. This modularity distinguishes ProbLog from, say, the finite state machines underpinning EMULSION, which manually aggregate effects within one state transition. Furthermore, ProbLog’s support for inhibition effects [17], which provide a concise and easily readable syntax for conditions which prevent an event by negating the head, extends modularity and clear semantics to preventative effects. Despite the detailed individual-level models, the ProbLog programs are kept compact and readable by encoding classes of events in first-order relational rules that are parameterised by variables.

Among probabilistic logic programming languages, ProbLog is particularly simple in its syntax, and simulation is supported by two well-maintained implementations, cplint (in Prolog) and ProbLog 2 (in Python) [11, 19]. Beyond simulation, they provide a variety of sophisticated learning and inference modes which could be added in future versions without altering the process generating the underlying ProbLog model. For instance, there are current plans to incorporate a learning mode which allows the estimation of disease parameters from observational data, which would ordinarily require a complete rewrite of the software.

The main price to be paid for our approach is in the mediocre performance of the MCMC sampling process. This has two main causes. Firstly, the Python implementation of MCMC sampling in ProbLog 2 is optimised for ease of use rather than performance, as has been demonstrated empirically in the past [19]. Secondly, the code automatically generated by our

translator is itself not ideal, since it does not cleanly separate deterministic and stochastic predicates. This is particularly problematic since in our experience, deterministic predicates tend to perform rather poorly under ProbLog 2 compared to state-of-the-art Prolog systems. However, the key focus of our work is not to provide a computationally competitive modelling framework. Rather, we want to offer a system for fine-grained local analysis that can illustrate the effect of behavioural choices in an individual's immediate vicinity.

Public Interest Discussion

We believe that our contribution is a step towards artificial intelligence for the social good. Although the notion of "public interest" can be difficult to define, authors have put forward frameworks to understand artificial intelligence in this context [20, 21].

For example, Züger and Asghari [21] outline some requirements that should be met for artificial intelligence applications to be considered "serving the public interest". Firstly, our approach uses artificial intelligence to understand infectious disease dynamics on a local scale, thereby having a public rather than a profit oriented justification. As probabilistic logic programming models of infectious disease are explicit and human-interpretable, bias and discrimination in the models is open to public scrutiny. By giving the general public the opportunity to evaluate the consequences of their actions against verifiable, openly accessible local models, we hope to contribute to a closing of the knowledge gap between experts, policymakers and those affected by their decisions.

Transparent specifications can ultimately move the field towards a truly "deliberative and participatory design process" [21, Subsection 3.3] of epidemiological models, as the model assumptions themselves can be critiqued and refined by members of the public.

This is not only important for the medical and policy reaction to epidemics, but also for public communication. The effectiveness of measures taken in response to epidemics depends crucially on the behaviour of individual actors. On the other hand, public health interventions have profound implications for the daily life and the wider prospects of individuals, posing challenges to both political culture and social cohesion. Individual-level, network-based approaches to epidemiological modelling are of particular significance for communication, since they can be used to model the impact of an epidemic on an individual himself and their own close circle.

This puts a huge weight on communicating the impact of measures to the public in a transparent and responsible way. This brings us to Züger and Asghari's [21] final two requirements for public interest: that public interest AI

systems need to implement technical safeguards, and that these systems need to be open to validation. Validation of simulation systems is a multi-faceted issue, which covers the validation of the model assumptions, validation of the model under those assumptions, and the validation of the simulation framework. In our case, validating the model itself is out of scope, since our solution is presented as a general framework for user-specified models. However, the transparent design and the clear semantics of the ProbLog language make the underlying assumptions of the models explicit and endow the modelling code with intuitively meaningful formal content that can then be externally validated as desired.

As a proof of concept that is not yet designed for public usage, no further technical safeguards have been implemented in the current version of SimPloID. In particular, the responsibility for ensuring data privacy lies with the user of the system. However, SimPloID is entirely stand-alone, with no data transmission taking place at any stage in the process. Therefore, sensitive data entered into the system by the user remains entirely within the user's domain.

5. Outlook

Although competitive performance is not the ultimate aim of SimPloID, epidemiological modelling can serve as a motivating application for the further development of probabilistic logic programming tools, systems and algorithms. Currently, work is ongoing to enhance the performance by transitioning from ProbLog 2 to a variant of the cplint system, more specifically to a sampling algorithm adapted from MCINTYRE [19] under the XSB Prolog system [22]. This is made possible by the recent release of the Janus bridge [23], which allows for a seamless integration of XSB from Python and thereby mitigates the downside of using a system from outside the host programming language of SimPloID. The availability of a competitive tabled Prolog engine such as XSB would also obviate the need for manually grounding out the time variable in a preprocessing step, keeping the programs passed to the engine compact. On the other hand, porting our application also requires adding support for inhibition effects by reimplementing the syntactic transformation described by Meert and Vennekens [17].

6. Conclusion

By making available a proof-of-concept prototype of a network-based epidemiological simulation suite based on ProbLog, we demonstrate the potential of probabilistic logic programming for supporting the discourse around public health and the effects of individual choices. The

suitability of probabilistic logic programming for this purpose is underpinned by its human-readable relational syntax and its well-defined declarative semantics, which sees separate clauses as independent causes of a prescribed effect. As a programming language can only ever be as usable as the systems that implement it, our framework demonstrates that the ProbLog ecosystem is sufficiently mature to support such an application, and opens the doors to various extensions enabled by the variety of query types and algorithms it provides.

References

- [1] J. Lourenço, F. Pinotti, C. Thompson, and S. Gupta, “The impact of host resistance on cumulative mortality and the threshold of herd immunity for SARS-CoV-2,” *MedRxiv*, vol. 2020–07, 2020.
- [2] R. Verity, L. C. Okell, I. Dorigatti, P. Winskill, C. Whittaker, N. Imai, G. Cuomo-Dannenburg, H. Thompson, P. G. Walker, H. Fu *et al.*, “Estimates of the severity of coronavirus disease 2019: a model-based analysis,” *The Lancet infectious diseases*, vol. 20, no. 6, pp. 669–677, 2020.
- [3] J. P. Ioannidis, “Reconciling estimates of global spread and infection fatality rates of COVID-19: an overview of systematic evaluations,” *European journal of clinical investigation*, vol. 51, no. 5, p. e13554, 2021.
- [4] S. Jenkins, “Was I wrong about coronavirus? Even the world’s best scientists can’t tell me,” *The Guardian*, 2020-04-02. [Online]. Available: <https://www.theguardian.com/commentisfree/2020/apr/02/wrong-coronavirus-world-scientists-optimism-experts>
- [5] R. M. Anderson and R. M. May, *Infectious diseases of humans: dynamics and control*. Oxford UP, 1991.
- [6] W. O. Kermack and A. G. McKendrick, “Contribution to the mathematical modelling of diseases I,” *Proc. R. Soc. Lond. A*, vol. 115, pp. 700–721, 1927.
- [7] F. Weitkämper, B. Sarbu, and K. Sun, “Modelling infectious disease dynamics with probabilistic logic programming,” in *ICLP 2021 Workshops: PLP 2021*, ser. CEUR Workshop Proceedings, J. Arias *et al.*, Eds., vol. 2970. CEUR-WS.org, 2021. [Online]. Available: <https://ceur-ws.org/Vol-2970/plppaper2.pdf>
- [8] F. Wong and J. J. Collins, “Evidence that coronavirus superspreading is fat-tailed,” *Proc. Natl. Acad. Sci. USA*, vol. 117, no. 47, p. 29416, Nov 2020.
- [9] L. De Raedt, A. Kimmig, and H. Toivonen, “ProbLog: A probabilistic Prolog and its application in link discovery,” in *IJCAI 2007*, M. M. Veloso, Ed., 2007, pp. 2462–2467. [Online]. Available: <http://ijcai.org/Proceedings/07/Papers/396.pdf>
- [10] D. Fierens, G. V. den Broeck, J. Renkens, D. S. Shterionov, B. Gutmann, I. Thon, G. Janssens, and L. D. Raedt, “Inference and learning in probabilistic logic programs using weighted Boolean formulas,” *Theory Pract. Log. Program.*, vol. 15, no. 3, pp. 358–401, 2015.
- [11] A. Dries, “Declarative data generation with ProbLog,” in *Proceedings of the Sixth International Symposium on Information and Communication Technology*, ser. SoICT 2015. New York, NY, USA: Association for Computing Machinery, 2015, pp. 17–24.
- [12] T.-M.-A. Bui, S. Stinckwich, M. Ziane, B. Roche, and T. V. Ho, “KENDRICK: a domain specific language and platform for mathematical epidemiological modelling,” in *11th IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF 2015)*. Can Tho, Vietnam: IEEE, Jan. 2015, pp. 132–137.
- [13] W. Waites, M. Cavaliere, D. Manheim, J. Panovska-Griffiths, and V. Danos, “Rule-based epidemic models,” *Journal of Theoretical Biology*, vol. 530, p. 110851, 2021.
- [14] I. Vendrov, C. Dutchyn, and N. D. Osgood, “Frabjous: A declarative domain-specific language for agent-based modeling,” in *Social Computing, Behavioral-Cultural Modeling and Prediction - 7th International Conference, SBP 2014, Washington, DC, USA, April 1-4, 2014. Proceedings*, ser. Lecture Notes in Computer Science, W. G. Kennedy, N. Agarwal, and S. J. Yang, Eds., vol. 8393. Springer, 2014, pp. 385–392.
- [15] J. Thaler, T. Altenkirch, and P. Siebers, “Pure functional epidemics: An agent-based approach,” in *Proceedings of the 30th Symposium on Implementation and Application of Functional Languages, IFL 2018, Lowell, MA, USA, September 5-7, 2018*, M. Cimini and J. McCarthy, Eds. ACM, 2018, pp. 1–12.
- [16] S. Picault, Y.-L. Huang, V. Sicard, S. Arnoux, G. Beaunée, and P. Ezanno, “EMULSION: transparent and flexible multiscale stochastic models in human, animal and plant epidemiology,” *PLoS comput. biol.*, vol. 15, no. 9, p. e1007342, 2019.
- [17] W. Meert and J. Vennekens, “Inhibited effects in CP-Logic,” in *PGM 2014*, L. C. van der Gaag and A. J. Feelders, Eds. Springer, 2014, pp. 350–365.
- [18] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Comp. in Sci. & Eng.*, vol. 9, no. 3, pp. 90–95, 2007.
- [19] F. Riguzzi, “MCINTYRE: a Monte Carlo system for probabilistic logic programming,” *Fund. Inform.*, vol. 124, no. 4, pp. 521–541, 2013.
- [20] L. Floridi, J. Cowls, T. C. King, and M. Taddeo,

- “How to design AI for social good: Seven essential factors,” *Science and Engineering Ethics*, vol. 26, no. 3, pp. 1771–1796, Jun 2020. [Online]. Available: <https://doi.org/10.1007/s11948-020-00213-5>
- [21] T. Züger and H. Asghari, “AI for the public. how public interest theory shifts the discourse on AI,” *AI & SOCIETY*, vol. 38, no. 2, pp. 815–828, Apr 2023. [Online]. Available: <https://doi.org/10.1007/s00146-022-01480-5>
- [22] T. Swift and D. S. Warren, “XSB: extending Prolog with tabled logic programming,” *Theory Pract. Log. Program.*, vol. 12, no. 1-2, pp. 157–187, 2012.
- [23] C. Andersen and T. Swift, “The Janus system: A bridge to new Prolog applications,” in *Prolog: The Next 50 Years*, D. S. Warren *et al.*, Eds. Springer, 2023, pp. 93–104.