Computational Discovery with Formal Languages and Synthesis

Sebastijan Dumancic[1], **Reuben Gardos Reid**[1]
[1]TU Delft
s.dumancic@tudelft.nl, r.j.gardosreid@tudelft.nl

Extended abstract

With computational modelling and simulation taking equal reign, next to theory and experimentation, as a paradigm of scientific discovery, many *formal languages* for describing natural phenomena have been invented. Kappa (Boutillier et al. 2018) and BioNetGen (Harris et al. 2016) are languages for modelling biochemical systems, including signal transduction, metabolic, and genetic regulatory networks. MØD (Andersen et al. 2014) is a graph-based language for describing chemical reaction systems. MedPath (Trajano et al. 2021) is a language for describing medicare care pathways. These formal languages essentially rely on various *formal models* from theoretical computer science to precisely describe the computation a natural system performs. For instance, graph rewriting systems are well suited for simulating chemical systems (Andersen et al. 2014), while qualitative networks, process algebra, Petri nets, and state machines are well suited for simulating gene regulatory and protein-protein interaction networks in biology (Fisher and Henzinger 2007). Relying on formal models has clear benefits: scientific findings become precise and unambiguous, can be simulated and compared with data, and can be analysed and verified (Konur 2023).

Unfortunately, formal languages are at odds with the modern-day machine-learning approaches to discovery. Most of these approaches operate on vectorised representations, and formal language does not fit this paradigm. While there is no doubt that these vector-based approaches have been effective, they are also not very well suited for scientific discovery. Namely, they are black boxes, and this has two significant consequences. First, whatever they discover about the underlying physical process remains opaque and cannot be verified. Second, the purely data-driven approaches are prone to learning *shortcuts* that perform well on training data but fail to generalise beyond.

In this talk, I will outline our alternative vision for data-driven discovery compatible with formal languages. Our vision builds upon program synthesis, a field of machine learning concerned with learning programs from data and/or demonstrations. Approaching computational discovery from a program synthesis perspective is attractive first because it makes it possible to make discoveries in the formal languages outlined above. These formal languages can be seen as simple programming languages, and program synthesis techniques induce models in arbitrary programming languages provided by the users. Second, program synthesis techniques make it easy to incorporate expert knowledge, both quantitative and qualitative. Third, such a flexible language for expressing knowledge makes integrating data from different sources and modalities easier. Fourth, program synthesis techniques are data-efficient which is important as big data is a luxury. Finally, formal languages make keeping the experts in the loop significantly easier as every intermediate model or conjunction is understandable.

The potential of program synthesis for computational discovery has already been demonstrated in executable biology (Koksal et al. 2013; Köksal et al. 2017; Fisher et al. 2015) and chemistry (Dalchau et al. 2015; Cardelli et al. 2017). However, each success was achieved with an ad hoc synthesis method and required a fresh start. Therefore, when moving from one problem to another, the synthesis techniques cannot be readily reused.

Our vision is to build an *universal* computational discovery engine where the formal languages for science form a foundation. The engine should be universal because it supports various formal languages with dedicated techniques. In my talk, I will argue why this combination makes sense, outline our vision, and discuss recent progress.

Relevant references

As this project has recently started, we do not yet have references to show for our work. We are, however, running two projects on scientific discovery at TU Delft, and our collaborators have substantial experience in computational discovery (Koksal et al. 2013; Köksal et al. 2017; Fisher et al. 2015; Fisher, Piterman, and Bodik 2014) We have substantial experience in program synthesis (Hocquette, Dumancic, and Cropper 2023; Cropper and Dumancic 2022; Dumancic, Guns, and Cropper 2021). We hope you will recognise that we can be productive participants at the Symposium.

References

Andersen, J. L.; Flamm, C.; Merkle, D.; and Stadler, P. F. 2014. 50 Shades of Rule Composition. In Fages, F.; and

Piazza, C., eds., *Formal Methods in Macro-Biology*, 117–135. Cham: Springer International Publishing. ISBN 978-3-319-10398-3.

Boutillier, P.; Maasha, M.; Li, X.; Medina-Abarca, H. F.; Krivine, J.; Feret, J.; Cristescu, I.; Forbes, A. G.; and Fontana, W. 2018. The Kappa platform for rule-based modeling. *Bioinformatics*, 34(13): i583–i592.

Cardelli, L.; Češka, M.; Fränzle, M.; Kwiatkowska, M.; Laurenti, L.; Paoletti, N.; and Whitby, M. 2017. Syntax-Guided Optimal Synthesis for Chemical Reaction Networks. In Majumdar, R.; and Kunčak, V., eds., *Computer Aided Verification*, 375–395. Cham: Springer International Publishing. ISBN 978-3-319-63390-9.

Cropper, A.; and Dumancic, S. 2022. Inductive Logic Programming At 30: A New Introduction. *J. Artif. Intell. Res.*, 74: 765–850.

Dalchau, N.; Murphy, N.; Petersen, R.; and Yordanov, B. 2015. Synthesizing and Tuning Chemical Reaction Networks with Specified Behaviours. In Phillips, A.; and Yin, P., eds., *DNA Computing and Molecular Programming*, 16–33. Cham: Springer International Publishing. ISBN 978-3-319-21999-8.

Dumancic, S.; Guns, T.; and Cropper, A. 2021. Knowledge Refactoring for Inductive Program Synthesis. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 7271–7278. AAAI Press.

Fisher, J.; and Henzinger, T. A. 2007. Executable cell biology. *Nat. Biotechnol.*, 25(11): 1239–1249.

Fisher, J.; Köksal, A. S.; Piterman, N.; and Woodhouse, S. 2015. Synthesising Executable Gene Regulatory Networks from Single-Cell Gene Expression Data. In Kroening, D.; and Păsăreanu, C. S., eds., *Computer Aided Verification*, 544–560. Cham: Springer International Publishing. ISBN 978-3-319-21690-4.

Fisher, J.; Piterman, N.; and Bodik, R. 2014. Toward Synthesizing Executable Models in Biology. *Frontiers in Bioengineering and Biotechnology*, 2.

Harris, L. A.; Hogg, J. S.; Tapia, J.-J.; Sekar, J. A. P.; Gupta, S.; Korsunsky, I.; Arora, A.; Barua, D.; Sheehan, R. P.; and Faeder, J. R. 2016. BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, 32(21): 3366–3368.

Hocquette, C.; Dumancic, S.; and Cropper, A. 2023. Learning Logic Programs by Discovering Higher-Order Abstractions. *CoRR*, abs/2308.08334.

Köksal, A. S.; Beck, K.; Cronin, D. R.; McKenna, A.; Camp, N. D.; Srivastava, S.; MacGilvray, M. E.; Bodík, R.; Wolf-Yadlin, A.; Fraenkel, E.; Fisher, J.; and Gitter, A. 2017. Synthesizing Signaling Pathways from Temporal Phosphoproteomic Data. *bioRxiv*.

Koksal, A. S.; Pu, Y.; Srivastava, S.; Bodik, R.; Fisher, J.; and Piterman, N. 2013. Synthesis of biological models from mutation experiments. *SIGPLAN Not.*, 48(1): 469–482.

Konur, G. M. . K. N., S. 2023. Verifiable biology. *Journal of the Royal Society*, 20(202).

Trajano, I. A.; Ferreira Filho, J. B.; de Carvalho Sousa, F. R.; Litchfield, I.; and Weber, P. 2021. MedPath: A process-based modeling language for designing care pathways. *International Journal of Medical Informatics*, 146: 104328.