BioHackrXiv

Logic Programming Working Group

Chris Mungall[1], **Hirokazu Chiba**[2], **Shuichi Kawashima**[2], **Yasunori Yamamoto**[2], **Pjotr Prins**[3], **Nada Amin**[4], **Deepak Unni**[1], **and William E. Byrd**[6]

1 Environmental Genomics and Systems Biology, Lawrence Berkeley National Laboratory, Berkeley, CA, USA **2** Database Center for Life Science, Research Organization of Information and Systems, Japan **3** Department of Genetics, Genomics and Informatics, The University of Tennessee Health Science Center, Memphis, TN, USA. **4** Harvard University, USA **5** Berkeley Lab, USA **6** University of Alabama at Birmingham, USA

DOI:
10.21105/biohackrxiv.0XXXX

Software

- Repository
- Branch

Submitted: 17 Mar 2020
Published: pending

Introduction

As part of the one week Biohackathion 2019 in Fukuoka Japan, we formed a working group on Logic Programming for the biomedical sciencies.

Logic programming in the form of relational SQL queries on database tables and SPARQL queries on semantic web graph data stores, is well known to many bioinformaticians. More advanced logic programming, however, is underutilized in bioinformatics. Prolog, for example, is a high-level programming language that has its roots in first-order logic or first-order predicate calculus. Another example, miniKanren, is an embedded Domain Specific Language for logic programming. Core miniKanren is exceptionally simple, with only three logical operators and one interface operator (Friedman, Byrd, Kiselyov, & Hemann, 2018).

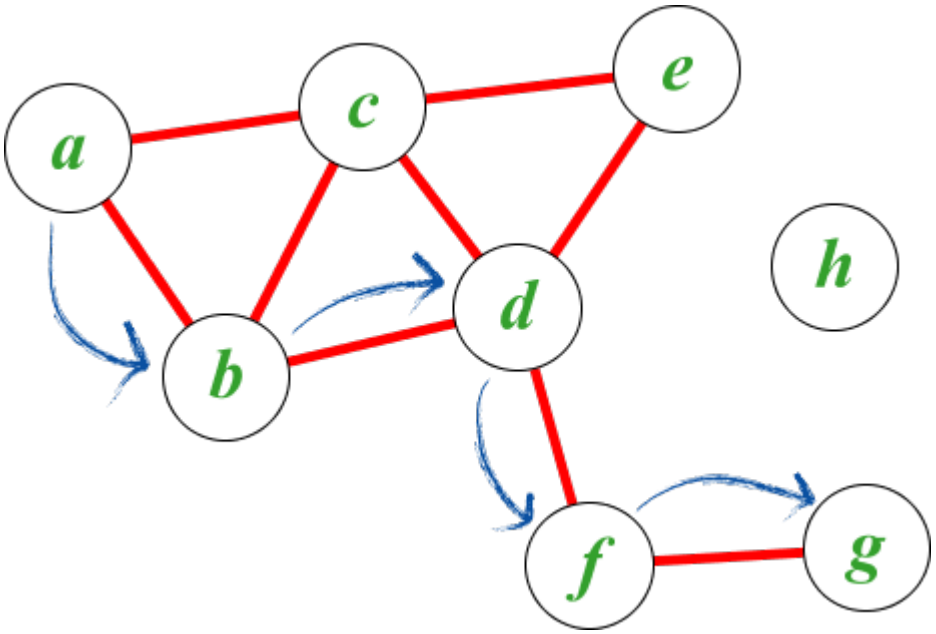

Figure 1: Logic programming resolver traverses the solution space to find all matches

The introduction of logic programming is particularly relevant in the context of multi-model data representations where data can be accessed in memory as free data structures, but also on disk where data can be represented as tables, trees (documents), and graphs. In bioinformatics

we can make use of all these different data sources and have a query engine that can mine them all efficiently.

Logic programming is well-suited for biological research. Essentially, a researcher writes a number of statements that include variables representing unknown information. The logic engine then goes through the solution space (all data) to find possible matches (see figure 1). Much more detail on the rationale and implementations of miniKanren and logic programming are well summarized in Byrd's book *The Reasoned Schemer, Second Edition* (Friedman et al., 2018), PhD thesis (Byrd, 2009), and online talks.

The 'Logic Programming' working group at the 2019 edition of the annual Japanese BioHackathon applied logic programming to various problems. The working group:

- researched state-of-the-art mapping between graph stores and logic programming;
- created methods for bridging between SPARQL and in-memory data representations using Prolog;
- extended the Biolink model;
- and added Relational Biolink type inference for mediKanren.

Research existing logic programming facilities for SPARQL

The working group researched current solutions for combining logic programming with SPARQL. ClioPatria is an in-memory RDF quad-store tightly coupled with SWI-Prolog by Jan Wielemaker, the main author of SWI-Prolog (Wielemaker, Beek, Hildebrand, & Ossenbruggen, 2016). SWI-Prolog is published under a BSD license, and there even exist bindings for ClioPatria and Python, for example, although we were unable to locate the source code. We think ClioPatria and SWI-Prolog are particularly useful for teaching, and for (in-memory) semantic web applications. SWI-Prolog comes with client libraries for SQL and SPARQL queries.

Application of SPARQLProg to biological databases

SPARQL provides a subset of what logic programming can do. The working group added to SPARQLProg which provides a way to define modular query components using logic programming.

A number of biological databases make their data available in RDF format, supporting SPARQL access—for example, Uniprot, NCBI Pubchem and the EBI RDF platform. Complicated SPARQL queries are required to effectively extract and combine information from multiple RDF databases.

SPARQL queries lack the property of composability, there is no way to reuse modular components across queries. For example, to execute a range query on a genomic region using the FALDO model (Bolleman et al., 2016) requires authoring a complex query over many triples. If we then wish to reuse parts of that query in a more complex query, we have to manually compose this together.

For example, a 4-part predicate `feature_in_range` can be composed with a binary `has_mouse_ortholog` predicate:

```
feature_in_range(grch38:'X',10000000,20000000, HumanGene),
has_mouse_ortholog(HumanGene, MouseGene)
```

This will compile down to a more complex SPARQL query, and execute it against a remote endpoint.

SPARQLProg includes bindings for many common biological SPARQL endpoint. As part of this hackathon we developed wrappers for RDF databases of MBGD (Uchiyama, Mihara, Nishide, Chiba, & Kato, 2019), KEGG OC, TogoVar, JCM, Allie, EBI BioSamples, UniProt, and DisGeNET. Future work includes using these Prolog codes as building blocks for integrative analysis.

SPARQLProg is written in SWI-Prolog and has a Python interface library. All code has been made available in the example directory of SparqlProg which provides sophisticated mapping of logic queries to SPARQL.

Extending the Biolink Model

The Biolink Model is a data model developed for representing biological and biomedical knowledge. It includes a schema and generated objects for the data model and upper ontology. The BioLink Model was designed with the goal of standardizing the way information is represented in a graph store, regardless of the formalism used. The working group focused on extending this model to support representation of a wide variety of knowledge.

The following tasks were accomplished as part of the BioHackathon:

1. represent datasets, and their related metadata
2. represent family and pedigree information, to support clinical knowledge
3. Make the provenance model more rich and descriptive

For future work, the group will ensure that the new classes added to the model will have appropriate mappings to other schemas and ontologies.

Relational Biolink type inference for mediKanren

miniKanren is an embedded Domain Specific Language for logic programming. The goal was to implement a relational type inferencer for the Biolink model in miniKanren, which can be integrated into mediKanren. The working group added a `yaml` subdirectory to the mediKanren GitHub page, and created multiple files in https://github.com/webyrd/mediKanren/yaml where `yaml2sexp.py` generates the `biolink.scm` file which contains an s-expression version of the Biolink yaml file. `yaml.scm` contains miniKanren relations, and Chez Scheme code that generates miniKanren relations based on `biolink.scm`. These are giant miniKanren `conde` clauses that can be though of as relational tables. `yaml.scm` also contains tests for the relations.

Future work:

1. integrate this work into the Racket mediKanren code
2. integrate with the data categories in the KGs
3. create query editor with decent type error messages, autocomplete, query synthesis, etc.

Discussion

The working group concluded that there is ample scope for logic programming in bioinformatics. Future work inludes expansion of accessing semantic web databases using SPARQLProg, expanding the BioLink model, and adding dynamic SPARQL support to miniKanren.

BioHackrXiv

References

Bolleman, J. T., Mungall, C. J., Strozzi, F., Baran, J., Dumontier, M., Bonnal, R. J., Buels, R., et al. (2016). FALDO: a semantic standard for describing the location of nucleotide and protein feature annotation. *J Biomed Semantics*, *7*, 39. doi:10.1186/s13326-016-0067-z

Byrd, W. E. (2009). *Relational programming in miniKanren: Techniques, applications, and implementations* (PhD thesis). Indiana University, Bloomington, IN, USA.

Friedman, D. P., Byrd, W. E., Kiselyov, O., & Hemann, J. (2018). *The Reasoned Schemer* (second edition.). Cambridge, MA, USA: MIT Press.

Uchiyama, I., Mihara, M., Nishide, H., Chiba, H., & Kato, M. (2019). MBGD update 2018: microbial genome database based on hierarchical orthology relations covering closely related and distantly related comparisons. *Nucleic Acids Res*, *47*(D1), D382–D389. doi:10.1093/nar/gky1054

Wielemaker, J., Beek, W., Hildebrand, M., & Ossenbruggen, J. van. (2016). ClioPatria: A SWI-Prolog infrastructure for the Semantic Web. *Semantic Web*, *7*(5), 529–541. doi:10.3233/SW-150191