Proposal to build a translational biomedical Reasoning Tool that would leverage a markov network-generation approach

Summary vision statement

Our proposal describes a Reasoning Tool in the spirit of Wolfram|Alpha [2]: the user will query the system in natural language and the Reasoning Tool will parse the input, generate search strategies to fulfill the user question, execute user-selected search strategies by traversing the relevant Knowledge Sources, and finally present results to the user. Similar to the highly successful Wolfram|Alpha, our Reasoning Tool will learn from repeated interactions with users to return the most relevant search strategies for a given type of natural language query.

This system will handle a variety of user queries spanning the spectrum of biomedical entity types in available Knowledge Sources. In particular, our proposed solution will be able to process queries involving a number of biological entities, with example questions including: *How are disease X and Y related?*, *How does variant X cause disease Y and protect against disease Z?*, *What is the clinical outcome pathway for the treatment of disease X by drug Y?*, *For all drugs used to treat disease X, which ones have the least well-studied clinical outcome pathway?*, etc.

Our approach (detailed in the project description) would advance current capabilities for biomedical knowledge querying in several respects. First, the process of generating and executing search strategies will be automated, removing the need to write custom Python notebooks for each query type. Our system, based on a probabilistic Markovian architecture, will both learn from repeated user interaction and also dynamically incorporate uncertainty in all steps of the proposed process. Relevant Knowledge Sources will also be automatically identified, removing the need for users to be experts in the variety of biological databases available. Lastly, by returning a variety of ranked search strategies and answers, our Reasoning Tool will present a dossier of results allowing the user to appreciate subtlety and complexity when it naturally arises from an input query (eg. some questions have more than one “correct” strategy for connecting Knowledge Sources).

Our team has the requisite depth and breadth of expertise to successfully construct the proposed Reasoning Tool within the requested 10-month time-frame. This includes significant experience in software engineering for distributed systems and significant research experience advancing the state-of-the-art in systems biology [4, 15, 16, 30], natural language processing [12–14, 24], mathematical algorithms for big data problems [11, 18–21], as well as database interactions and knowledge discovery [17, 25–27]. Individual components of this Reasoning Tool have been implemented for different applications (as detailed in the previous references) and similar paradigms are employed by a variety of previously implemented knowledge engines with high-profile examples including IBM’s Watson [9] and the aforementioned Wolfram|Alpha [2].

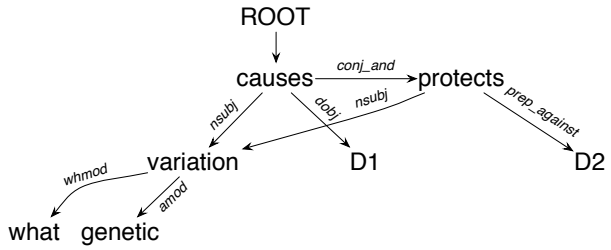
Finally, the Reasoning Tool we propose will be built in a modular, extensible fashion allowing it to interact with new Knowledge Sources. Since we employ a Markov chain architecture to generate search strategies and a Dijkstra-like algorithm to execute them, new biological entities can easily be incorporated. The flexibility of this framework allows for quick integration of additional constraints and new query types with the advantage of not needing to re-train the entire system whenever a new extension is added (in contrast to deep learning/neural network architectures).

Project plan

We propose a three-component system that will (i) process natural language user queries, (ii) generate a dossier of ranked search strategies, and (iii) execute the desired search strategies.

Natural language processing

A critical functionality of a Reasoning Tool is the ability to parse and understand free-form user input. This process involves two steps: (a) **syntactic parsing**, which converts the natural language query into a tree or directed graph structure modeling syntactic dependencies, and (b) **semantic parsing**, which converts the syntactic structure into a first-order logic formula. For syntactic parsing, to start with, we will use the widely-used Stanford parser which converts the sentence into a directed acyclic graph known as the "Stanford dependency" [5] (Figure 1(a)). Then for semantic parsing, we will write our own converter that transforms the Stanford dependency graph into first-order logic formula (Figure 1(b)). A potential problem with this approach is that the Stanford parser is mostly trained on news text and may not perform the best for the biomedical domain, but we will retrain the Stanford parser with additional in-domain data from BioNLP contests [28]. The NLP step will result in a first-order logic formula (Fig. 1(b)). This representation of the user query will be used by the Reasoning Tool to generate a dossier of search strategies.



(a) step 1: syntactic parsing

cause (G?, D1) $\wedge$ protect_against (G?, D2)

(b) step 2: semantic parsing

Figure 1: Natural language parsing of the input query *"What genetic variant causes D1 and protects against D2?"* where D1 and D2 stand for two diseases. (a) syntactic parsing: from query to Stanford dependencies. (b) semantic parsing: from dependencies to first-order logic.

Generate a dossier of search strategies

Given a parsed natural language query as input, our Reasoning Tool will return a *dossier* of possible *search strategies.* A search strategy is a tree with vertices being biological entity types (including genetic variants, drugs, genes, molecules, pathways, diseases, diagnoses, etc.) and edges being query templates of Knowledge Sources to identify relationships between pairs of those entities. Terminal vertices in this tree correspond to the entity types of the constants provided by the user's parsed natural language query. To illustrate, we show the tree for three simple search strategies that the Reasoning Tool would include in the dossier for two user queries: *How are disease DX and disease DY connected?* (Fig. 2a-b) and *How does variant G cause disease DX and protect against disease DY?* (Fig. 2c).

How the Reasoning Tool will generate and prioritize search strategies

In implementing the Reasoning Tool, we will use a Markov chain approach to generate trees with the constant terms from the user's query as terminal vertices in the tree. This approach enables the prioritization of search strategies with fewer Knowledge Source lookups. To illustrate, recall the hypothetical user query *How are disease DX and disease DY connected?* With two constants (i.e., terminal vertices in the tree), the Reasoning Tool would generate graphs according to the Markov process shown in Fig. 3a). Such Markov chains will be constructed for each possible one-

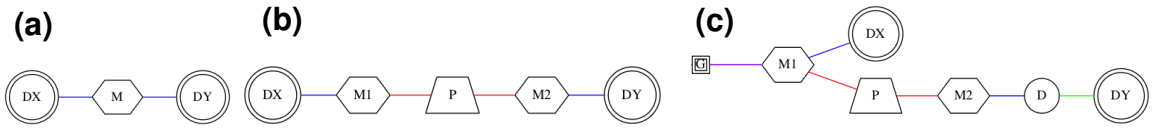


Figure 2: Three example search strategies returned by the Reasoning Tool: (a) and (b) are for a user query *How are disease DX and disease DY connected?* Search strategy (c) is for a query *How does variant G cause disease DX and protect against disease DY?* Double-shapes denote constants in the query. Shapes denote bioentity type (circle: disease, hexagon: molecule, trapezoid: pathway, box: variant). Internal vertex labels like "M" and "P" representing the entity *type,* not a specific entity. Edge colors denote the type of association (blue: disease-molecule, red: molecule-pathway, green: disease-disease, purple: variant-molecule).

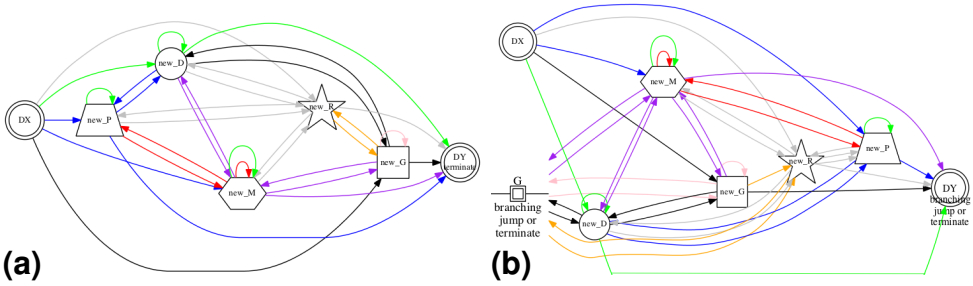


Figure 3: Markov chain state transition graphs for generating ranked query strategies for (a) an example two-constant query (diseases DX and DY) and (b) a three-constant query (variant G and diseases DX and DY). Double-shape vertices are the constants of the natural-language query. Colored arcs represent searches for entity-entity associations in various database sources [blue: disease-molecule association (e.g., DisGeNET, ClinVar), red: molecule-pathway association (e.g., Pathway Commons 2, BioCyc), green: disease-disease association (e..g, Monarch, DNetDB), purple: variant-molecule association (e.g., ClinVar, BeFree), pink: variant-variant association (SNAP or IGSR), orange: drug-variant interaction (PharmGKB), black: variant-disease association (ClinVar, DisGeNET)]. Each shape corresponds to a different type of biological entity (M: molecule, P: pathway, R: drug, G: variant, D: disease). In (b), the "branching jump or terminate" vertices enable the generation of a branched tree.

constant, two-constant, and three-constant natural-language query (and are trivially extensible). Each realization of the Markov chain will generate a possible search strategy, i.e., a tree, and will assign a probability to each search strategy by penalizing weak associations and excessively long chains of Knowledge Source lookups.

Training of the Markov chain transition probabilities

The transition probabilities (weights of the colored edges in Fig. 3) will be determined empirically using a set of 100 natural-language queries that we will assemble from template queries from online NCATS-Translator resources and using biological mechanisms (and successful search strategies for these queries) for drug-variant interactions, variant-disease mechanisms, etc. We will select transition probabilities that minimize the ranks of the 100 successful search strategies for the 100 example queries, among the dossiers produced for these queries (using the principal axes and eigenvalues of the Hessian to prioritize important linear combinations of the log-probability weights to be fine-tuned in the training pocedure).

Demonstrating translational utility of integrating different knowledge sources

Suppose a user wishes to find a common mechanism underlying the causal effect of variant rs334 for sickle-cell anemia and that same variant's protective benefit against malaria. The natural-language query has three constants, and one possible search strategy is given in Fig. 2c. If executed against the databases Pathway Commons (red), Disease Ontology (green), DisGeNET

(blue), and ClinVar (purple), the search strategy would return an explanatory mechanism, as shown in Fig. 4, that is consistent with current knowledge of the role of heme in cerebral malaria [8].

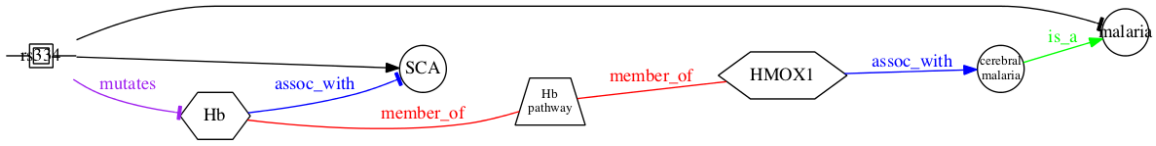


Figure 4: Example of how the Reasoning Tool-returned search strategy of Fig. 2c would find entity-entity associations that would suggest a common mechanism underlying the protective benefit of the rare variant rs334 against malaria and the causal effect of rs334 for sickle-cell anemia (SCA). Hb, hemoglobin b; HMOX1, heme oxygenase 1. Edge colors: see Fig. 3b.

Executing search strategies

After the search strategy (tree generated by the Markov chain) is selected, we will employ a modified Dijkstra algorithm to find paths between source and target entities that minimize a sum of edge-weights, where smaller edge-weights indicate stronger evidentiary value. The straightforward modification will ensure that, for causal paths of influence, the overall "sign" of the path (activating or inhibiting) is consistent with the semantics of the user's natural-language query.

Anticipated functionality for the proof-of-concept (POC) tool

For the POC questions, the system we build will allow the user to select from one of two questions: "which genetic condition(s) protect(s) from a given disease D" and "what is the outcome pathway for drug R and condition C". The condition C and/or drug R will be user-specified. Pre-computed search strategies for these two questions will be implemented. The modified Dijkstra algorithm will be implemented and will search the Knowledge Sources in real time, based on the search query. If funded, our full Reasoning Tool will allow for natural language input (Section 1.1), automatic generation of search strategies (Section 1.2) and will account for empirically determined evidentiary weights for different types of entity-entity associations (Sec. 1.3).

Components of the proposed software stack

We will implement our system in Python, using `graph-tool` [29] or `SNAP` [22] for graph searching and querying, `Boost.Python` [3] for implementing the modified Dijkstra algorithm, `Py4J` [1] for accessing Java-only APIs as needed (see Sec. 1.6), `Flask` [10] for providing a REST API, and `Requests` and `json` to access the blackboard REST API. We will use a Python API for Stanford CoreNLP [23]. We will publish on GitHub all code and models developed for the Reasoning Tool.

How proposed software will interact with Translator

We will use the Java-based NCATS Knowledge Beacon Aggregator to obtain metadata (i.e., schemas and entity counts) from Knowledge Sources via RESTful queries leveraging the common Translator Knowledge Beacon Application Programming Interface (KBAPI). Our system will provide the functionality of converting a natural-language-query to a dossier of search strategies (JSON format), via a REST API complying with the NCATS Knowledge Beacon Web API.

Beyond Markov chains—Markov Logic Networks

We will additionally explore using Markov Logic Networks [6, 7] that would probabilistically represent possible connections between different types of biological entities in different Knowledge Sources, as well as the nature of those connections (e.g., inhibitory, causal, etc.). MLN is a natural generalization of the undirected graphical model approach described in Sec. 1.2. As such, the use of Markov Logic Networks could improve the ability to map parsed natural-language queries to the search strategies that are most likely to yield biologically relevant information.

References

[1] Py4J. https://www.py4j.org/index.html. Accessed: 2017-09-22.

[2] Wolfram|Alpha. http://www.wolframalpha.com/. Accessed: 2017-09-22.

[3] Boost.Python. http://www.boost.org/libs/python/doc/. Accessed: 2017-09-22.

[4] Pedro De Atauri, David Orrell, Stephen Ramsey, and Hamid Bolouri. Evolution of design principles in biochemical networks. *Systems biology*, 1(1):28–40, 2004.

[5] Marie-Catherine de Marneffe and Christopher D. Manning. Stanford typed dependencies manual, 2008.

[6] P M Domingos and W A Webb. A Tractable First-Order Probabilistic Logic. *AAAI*, 2012.

[7] Pedro Domingos and Matthew Richardson. 1 markov logic: A unifying framework for statistical relational learning. *Statistical Relational Learning*, page 339, 2007.

[8] Ana Ferreira, Ivo Marguti, Ingo Bechmann, Viktória Jeney, Ângelo Chora, Nuno R Palha, Sofia Rebelo, Annie Henri, Yves Beuzard, and Miguel P Soares. Sickle hemoglobin confers tolerance to Plasmodium infection. *Cell*, 145(3):398–409, April 2011.

[9] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[10] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2014.

[11] Andreas Holzinger, Matthias Hörtenhuber, Christopher Mayer, Martin Bachler, Siegfried Wassertheurer, Armando J Pinho, and David Koslicki. On entropy-based data mining. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, pages 209–226. Springer, 2014.

[12] Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 144, 2007.

[13] Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086. Association for Computational Linguistics, 2010.

[14] Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics, 2012.

[15] Daehee Hwang, Alistair G Rust, Stephen Ramsey, Jennifer J Smith, Deena M Leslie, Andrea D Weston, Pedro De Atauri, John D Aitchison, Leroy Hood, Andrew F Siegel, et al. A data integration methodology for systems biology. *Proceedings of the National Academy of Sciences of the United States of America*, 102(48):17296–17301, 2005.

[16] Daehee Hwang, Jennifer J Smith, Deena M Leslie, Andrea D Weston, Alistair G Rust, Stephen Ramsey, Pedro de Atauri, Andrew F Siegel, Hamid Bolouri, John D Aitchison, et al. A data integration methodology for systems biology: experimental verification. *Proceedings of the National Academy of Sciences of the United States of America*, 102(48):17302–17307, 2005.

[17] HV Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 13–24. ACM, 2007.

[18] David Koslicki and Daniel J Thompson. Coding sequence density estimation via topological pressure. *Journal of mathematical biology*, 70(1-2):45–69, 2015.

[19] David Koslicki and Hooman Zabeti. Improving min hash via the containment index with applications to metagenomic analysis. *bioRxiv*, page 184150, 2017.

[20] David Koslicki, Simon Foucart, and Gail Rosen. Quikr: a method for rapid reconstruction of bacterial communities via compressive sensing. *Bioinformatics*, 29(17):2096–2102, 2013.

[21] David Koslicki, Simon Foucart, and Gail Rosen. Wgsquikr: fast whole-genome shotgun metagenomic classification. *PloS one*, 9(3):e91784, 2014.

[22] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[23] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[24] Haitao Mi and Liang Huang. Forest-based translation rule extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 206–214. Association for Computational Linguistics, 2008.

[25] Arnab Nandi and HV Jagadish. Assisted querying using instant-response interfaces. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1156–1158. ACM, 2007.

[26] Arnab Nandi and HV Jagadish. Effective phrase prediction. In *Proceedings of the 33rd international conference on Very large data bases*, pages 219–230. VLDB Endowment, 2007.

[27] Arnab Nandi and HV Jagadish. Guided interaction: Rethinking the query-result paradigm. *Proceedings of the VLDB Endowment*, 4(12):1466–1469, 2011.

[28] Claire Nedellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. Overview of bionlp shared task 2013. In *Proc. of BioNLP*, 2013.

[29] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: $10.6084/\mathrm{m9.figshare.}1164194$. URL `http://figshare.com/articles/graph_tool/1164194`.

[30] Stephen A Ramsey, Elizabeth S Gold, and Alan Aderem. A systems biology approach to understanding atherosclerosis. *EMBO molecular medicine*, 2(3):79–89, 2010.