mediKanren: a System for Biomedical Reasoning

WILLIAM E. BYRD, GREGORY ROSENBLATT, MICHAEL J. PATTON, and THI K. TRAN-NGUYEN, Hugh Kaul Precision Medicine Institute, University of Alabama at Birmingham, USA
MARISSA ZHENG, Harvard University, USA
APOORV JAIN, Indian Institute of Technology Delhi, India
MICHAEL BALLANTYNE, Programming Research Laboratory, Northeastern University, USA
KATHERINE ZHANG, Harvard University, USA
MEI-JAN CHEN*, JORDAN WHITLOCK, MARY E. CRUMBLEY†, and JILLIAN TINGLIN‡, Hugh Kaul Precision Medicine Institute, University of Alabama at Birmingham, USA
KAIWEN HE, Department of Computer Science, University of Alabama at Birmingham, USA
YIZHOU ZHANG§, Harvard University, USA
JEREMY D. ZUCKER and JOSEPH A. COTTAM, Pacific Northwest National Laboratory, USA
NADA AMIN, Harvard University, USA
JOHN OSBORNE, Informatics Institute, University of Alabama at Birmingham, USA
ANDREW CROUSE and MATTHEW MIGHT, Hugh Kaul Precision Medicine Institute, University of Alabama at Birmingham, USA

mediKanren is a reasoning engine over biomedical knowledge, based on miniKanren. This paper gives a system description, along with examples of queries in low-level and medium-level query languages. We describe the implementation of mediKanren. We also illustrate the utility of mediKanren by describing how query results have resulted in treatment for a patient with a rare genetic disease. We also describe ways in which mediKanren might be improved or extended in the future. mediKanren is being developed as one of the reasoning engines for the NIH NCATS Biomedical Data Translator Program, and is compliant with standards and knowledge graphs produced by that program.

Additional Key Words and Phrases: relational programming, miniKanren, Racket, Scheme, relational programming, functional programming, precision medicine, drug repurposing, NCATS Biomedical Data Translator

1 INTRODUCTION

With over 1.5 million publications per year and more than 50 million total peer-reviewed articles, the rate and volume of novel discoveries has surpassed our ability to fully utilize and understand what is known [Jinha 2010]. This problem, described as the "unknown known," is characterized by two specific features: 1) "forgotten facts"—facts that are published but not widely known; and 2) "uninferred facts"—facts that have not been deduced

*now independent (current email: mjchen0098@gmail.com)
†now at University of Michigan Medical School (current email: mcrumble@med.umich.edu)
‡now at University of Alabama at Birmingham School of Medicine
§now at University of Waterloo (current email: yizhou.zhang@uwaterloo.ca)

miniKanren 2020, August 27 2020, Online

from existing published research. While all professional fields are subject to the effects of the "forgotten" or "uninferred" facts, the cost of the unknown known for healthcare providers is measured in human lives.

To tackle the issue of the uninferred medical knowledge, the University of Alabama at Birmingham's Hugh Kaul Precision Medicine Institute (UAB-HKPMI) has developed a software reasoning tool called *mediKanren* as a part of a multi-institutional grant funded by the National Center for Advancing Translational Science (NCATS). Since the tool's inception, mediKanren has been successful in finding novel FDA-approved therapeutic recommendations for disorders ranging from undiagnosed and purely symptomatic disease to genetically diagnosed metabolic disorders [Ross et al. 2019; Shepard 2019].

In this paper we give an overview of mediKanren, along with details of mediKanren's implementation and real-life example precision medicine queries. In Section 2 we provide background through an extensive glossary of all the main terms used in the paper. In Section 3 we give an example real-world scientific question, in which we want to change—or "budge"—the expression of a gene implicated in a disease. In Section 4 we answer this gene budging question using low-level and medium-level query languages, and using the mediKanren graphical user interface (GUI). In Section 5 we expand on the simple gene budging query to handle a set of genes implicated in Acute Respiratory Distress Syndrome (ARDS) in patients with COVID-19. In Section 6 we discuss how mediKanren performs concept normalization (sometimes called "synonymization") in order to find related concepts automatically, and to allow for cross-knowledge-graph queries. We also discuss tooling to automatically create maps of knowledge graphs. In Section 7 we discuss the core implementation of mediKanren. In Section 8 we discuss future directions. In Section 9 we conclude.

The example knowledge graphs and query answers in this paper are subject to change, as the knowledge graphs, mediKanren code, and underlying datamodel are improved.

2 BACKGROUND

mediKanren is being developed as part of the NIH NCATS Biomedical Data Translator Program. The following glossary defines common terms used in the NCATS Biomedical Data Translator Program, and in this paper. This glossary is designed to be read in order, and describes the essential background of the paper.

NCATS Biomedical Data Translator Program Multi-institutional program run by the National Center for Advancing Translational Sciences (NCATS), which is part of the National Institutes of Health (NIH). The program's goal is to combine structured knowledge with reasoning software that can be used by scientists/physician-scientists to accelerate the pace of *translational science*: translating basic scientific knowledge into treatments for patients.

Translator System The overall software system, structured knowledge, standards, and practices being developed by the NCATS Biomedical Data Translator Program.

Translator Architecture The architecture of the Translator System. The main components of the Translator Architecture are the autonomous relay service (ARS), autonomous reasoning agents (ARAs), and knowledge providers (KPs). The Translator Architecture also includes associated standards, including the Biolink Model, the Translator API (TRAPI), the KGX (Knowledge Graph eXchange) standard, and the *Smart API*.

NCATS Biomedical Data Translator Consortium A consortium of NIH NCATS-led research groups at multiple government institutions and universities working together to develop the Translator Architecture and System.

precision medicine Sometimes called *personalized medicine*. An approach to medicine in which treatment is tailored to each patient's genetic, environmental and lifestyle factors. For example, precision oncology might take into account specific mutations in the genome of the patient's tumors, along with the gene expression profiles in the cancer cells, using DNA and RNA sequencing. This information might be used to discover—or more likely, repurpose—drugs that might be especially effective for that patient.

drug discovery The process of finding a new chemical compound that treats one or more diseases. This process may involve laboratory bench work, computer simulations, large-scale robotic testing of thousands of compounds, or other expensive techniques.

clinical trial In the United States, a new drug must pass through multiple phases of clinical trials before being approved by the federal Food and Drug Administration (FDA). Bringing a newly discovered candidate drug to market is extremely expensive and usually takes many years, partly because many drugs are found to be unsafe in humans (fail the phase I of the clinical trial), or are found to be safe but ineffective in treating the target disease (fail phase II or phase III of the clinical trial).

drug repurposing The process of taking an existing drug that has been shown to be safe in humans (passed phase I of a clinical trial), and using that drug to treat a disease other than for which it was originally intended. A drug that passes phase I of a clinical trial, but fails phase II or phase III of that trial, might still be effective for treating some other disease that was not studied in the clinical trial. Since the drug has already passed phase I, and is considered safe, the time and cost to bring the drug to market can be greatly reduced. Furthermore, it may be possible for a physician to prescribe the drug to an individual patient even if the drug has not been shown to be effective at treating any specific disease, as long as there is evidence that the patient might benefit from the treatment, and that the risk of treatment is low.

gene expression level The amount of messenger RNA (mRNA) transcript produced for a given gene in a cell. This can be measured by techniques such as RNA sequencing, which can give an entire *gene expression profile*, showing which transcripts are being overexpressed or underexpressed in the cells of a patient, relative to a typical healthy cell. mRNA transcripts can be translated into proteins, which perform most of the functions within the cell.

protein expression level The amount of protein translated from a given gene in a cell. This can be measured by techniques such as proteomics, which can give an entire *protein expression profile*, showing which proteins are being overexpressed or underexpressed in the cells of a patient, relative to a typical healthy cell. Protein expression level is necessary, but not sufficient, for protein activity.

protein activity The rate at which a protein performs its function. Protein activity can be directly measured by bioassays, but these are, in general, not high throughput, so it is difficult to measure which proteins are active or inactive in the cells of a patient, relative to a typical healthy cell. The change in activity of proteins can cause disease symptoms (also known as *disease pheonotypes*), which can be treated in some cases through gene budging.

gene budging Using a drug to increase or decrease the activity of one or more proteins implicated in a disease or disease phenotypes, in order to try to treat the disease or its symptoms. Finding drugs that are safe in humans, and which can budge the activities of one or more target proteins in the desired direction (increasing or decreasing activity) is one approach to drug repurposing that we demonstrate in this paper, using mediKanren.

Semantic Web A set of related technologies, standards, and practices to develop interconnected knowledge sources that support automatic semantic reasoning. While originally proposed as an improvement to the World Wide Web, Semantic Web technologies and techniques can be used more generally. The NCATS Biomedical Data Translator Program has adopted many Semantic Web-related standards, such as CURIEs. The Semantic Web philosophy has influenced the development of new standards that are part of the Translator Architecture.

CURIE Compact Uniform Resource Identifiers (CURIEs) [W3C 2010] are machine-readable identifiers of the form `<knowledge-source>:<id>`. For example, `UMLS:C1425762` is a CURIE representing the human gene RHOBTB2. The prefix `UMLS` refers to the Unified Medical Language System (UMLS), which provides mappings between many different controlled vocabularies. The suffix `C1425762` is an identifier that is

unique within the UMLS knowledge source. CURIEs can be used to unambiguously refer to a concept or predicate, and to link concepts between different knowledge graphs.

concept An entity in the real world—for example, the drug *imatinib*, along with associated information about the drug. A concept may also contain metadata, such as provenance and context. A concept can be compactly represented as a *CURIE*.

concept category The type of a concept. For example, the concept *diabetes* might have the category *Disease*.

predicate A verb linking two concepts (the subject and the object). For example, the predicate *treats* might connect a *Drug* concept to a *Disease* concept.

edge A subject-predicate-object triple, where the subject and object are both concepts, along with associated data, representing an assertion—for example, the assertion *imatinib treats cancer*. An edge may also contain metadata, such as provenance and context.

knowledge graph (KG) A graph containing edges describing the relationship between concepts (nodes).

map of a knowledge graph A high-level graph showing only the concept categories, and predicates connecting concept categories, in a concrete knowledge graph. For example, if a concrete knowledge graph were to contain edges *imatinib treats cancer* and *insulin treats diabetes*, a map of that concrete knowledge graph might include the abstract edge *Chemical treats Disease*.

node normalization Also called *concept normalization*, *node synonymization*, or *concept synonymization*. The process of determining which distinct CURIEs actually represent the same concept in the real world.

edge normalization The process of determining which distinct subject-predicate-object edges actually represent the same assertion.

controlled vocabulary A curated set of terms describing concepts and/or predicates, usually from some specialized field, such as chemistry or medicine.

ontology A controlled vocabulary that also includes structured relationships between concepts, such as *is-a* (parent-child relationship) or *is-sibing-of* predicates.

***n*-hop query** A query between two concepts with precisely *n* unknown edges in the path between those concepts.

Biolink Model A standard datamodel for biomedical data used in the Translator System. The knowledge graphs used by mediKanren are in Biolink Model format.

knowledge provider (KP) A component of the overall Translator System that can provide structured knowledge to an autonomous reasoning agent (ARA), either dynamically in response to a API-based query, or as a bulk transfer of a knowledge graph via KGX (Knowledge Graph eXchange). Different KPs may specialize in different types of knowledge—for example, genomic information.

autonomous reasoning agent (ARA) A component of the overall Translator System that can perform reasoning over the structured knowledge provided by one or more knowledge providers (KPs) in response to a query. The mediKanren system described in this paper is one of several ARAs being developed as part of the Translator System. An ARA can be run in "stand-alone" mode, in which the ARA directly interacts with user queries, and directly provides responses to the user. For example, the GUI described in Section 4.3 allows a user to directly interact with mediKanren in stand-alone mode. In the context of the complete Translator System, a query would normally be sent to an ARA from the central autonomous relay service (ARS), or perhaps from another ARA, using Translator API (TRAPI) messages. Similarly, an ARA would respond to the query with a TRAPI-compliant message. Different ARAs may specialize in different types of reasoning, and may be implemented using any technology, as long as they abide by the standards of the Translator Architecture.

autonomous relay service (ARS) A component of the overall Translator System that acts as intermediary between an end user and the autonomous reasoning agents (ARAs). The ARS interprets a user query,

turning that query into Translator API (TRAPI) messages that are sent to one or more ARAs. Unlike the multiple ARAs and multiple knowledge providers (KPs) in the Translator System, there is a single ARS.

mediKanren One of several autonomous reasoning agents (ARAs) being developed as part of the NCATS Biomedical Data Translator System. mediKanren is the ARA described in this paper.

Translator API (TRAPI) An application programming interface (API) standard being developed as part of the NCATS Biomedical Data Translator Program. The TRAPI allows for the ARS, ARAs, and KGs to exchange information about queries.

KGX (Knowledge Graph eXchange) An NCATS Biomedical Data Translator Program standard—and set of tools—for bulk data exchange of knowledge graphs between KPs and ARAs.

Knowledge graphs (KGs) provide a unified way of organizing relations between concepts that exist in disparate locations and have been utilized in a variety of fields ranging from medicine to biodiversity and ecology [RDM 2019; Rotmensch et al. 2017].

As a part of a collaboration between UAB's Hugh Kaul Precision Medicine Institute and the NCATS Biomedical Data Translator Consortium, mediKanren make use of several knowledge graphs provided by other teams. Any knowledge graph compliant with the Translator KGX (Knowledge Graph eXchange) format can be readily imported into mediKanren. To give an example of a knowledge graph used by mediKanren, the Semantic MEDLINE Database (SemMedDB) [**?**] knowledge graph has more than 107 million assertions summing up more than 31 million publications (as of mid-2020). mediKanren also incorporates the *RTX*, *ROBOKOP*, and *Orange* knowledge graphs; the names of these knowledge graphs are derived from the team names in the early stage of the Translator project.

Figures 10 and 11 on pages 23 and 24 outline the mapped edges contained within the RTX and ROBOKOP KGs.

Knowledge graphs consist of medical concepts, represented as graph vertices, and relationships between them, represented as directed graph edges. Edges between concepts have a subject-predicate-object structure, as in the Resource Description Framework (RDF) data model. Both concepts and edges are attributed. Edge attributes include metadata about provenance and evidence supporting the relationship claim. For example, a triple of subject-predicate-object is "imatinib inhibits KIT gene" and the evidence would link to publications supporting that claim.

A subgraph containing further claims around "imatinib" is shown in Figure 12 on page 25. A bigger subgraph containing further claims around "imatinib" is shown in Figure 13 on page 26. In both cases, we are inferring that "imatinib treats asthma," a new finding based on known mechanisms.[1]

3 AN EXAMPLE SCIENTIFIC QUESTION: RHOBTB2 2-HOP QUERY

The question attempts to find a therapeutic option for UAB-HKPMI participants with rare mutations resulting in over-expression/gain-of-function of the RHOBTB2 gene. The clinical manifestations of RHOBTB2 over-expression include seizures, severe global developmental delay, movement disorders, ataxia, and generalized decrease in muscle tone. Thus, the query aims to find FDA-approved drugs `X` that directly (1-hop) or indirectly (2-hop, through some gene or protein `Y`) inhibit the over-production of the RHOBTB2 gene.

To find a safe drug, we go through an edge, `drug-safe`, that encapsulates whether a drug has a tradename. This is not a property of the drug, but is in the form of an edge/claim with the subject being the drug, the predicate being "has tradename," and the object being the trademark.

We will see how to answer this query in the next section, using low-level and medium-level languages.

[1] NCATS provided the example problem of trying to find mechanistic connections between imatinib and asthma during the initial Translator Feasibility Phase.

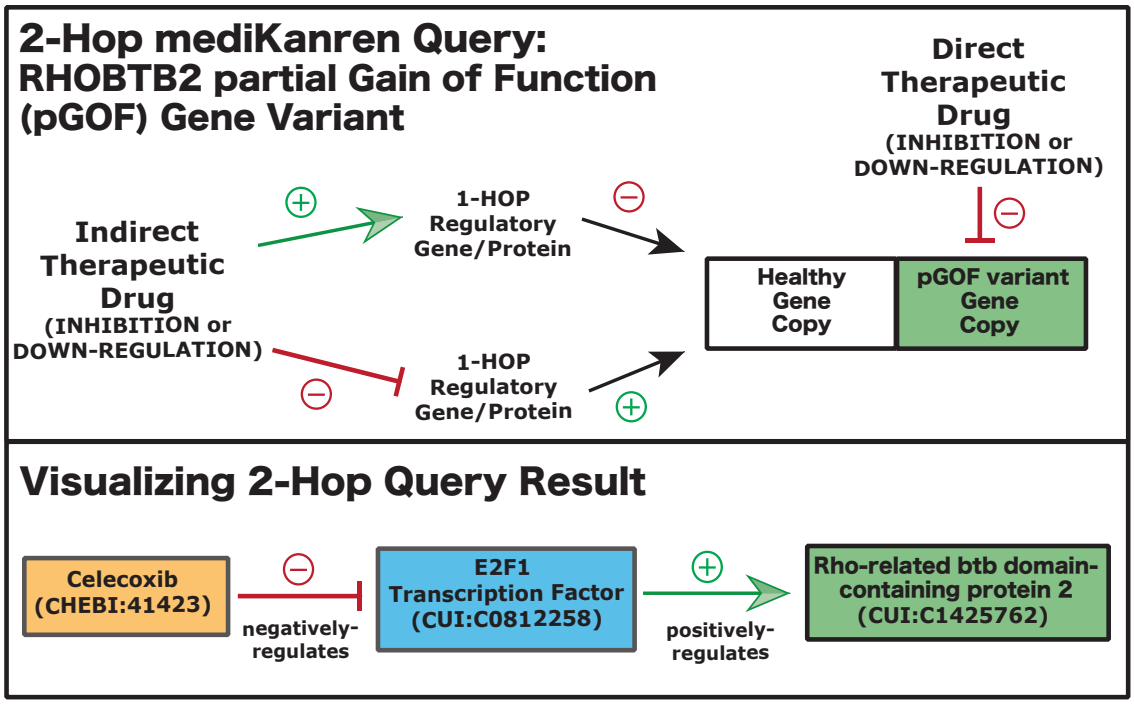

Fig. 1. The top panel of this diagram shows the therapeutic strategy for partial gain-of-function (pGOF) genetic variants, such as the RHOBTB2 case from UAB-HKPMI. The bottom panel serves as a visualization aid for the 2-hop mediKanren query result. The FDA-approved drug celecoxib was found to be an indirect inhibitor of the E2F1 transcription factor, which is responsible for expression of the RHOBTB2 gene. Concept CURIEs are notated in parentheses.

4 QUERYING KNOWLEDGE GRAPHS WITH MEDIKANREN

mediKanren currently supports two query languages: a low-level language that is essentially raw miniKanren, extended with relations to traverse knowledge graphs; and a more abstract interface, `query/graph`, that makes standard queries easier to write and to read.[2]

We assume the reader is familiar with the basics of miniKanren and Racket. For readers unfamiliar with miniKanren, the language is described in Friedman et al. [2018] and Byrd [2009]. A high-level overview of Racket is given by Felleisen et al. [2015].

4.1 Low-level Query Language (miniKanren)

Knowledge graphs can be queried directly in miniKanren, with the help of relations in mediKanren's miniKanren/graph database interface library. Graph database relations `~cui-concepto` and `edgeo` can be used to find

[2]mediKanren also supports `run/graph`, which is effectively a variant of `query/graph` without concept/edge normalization.

concepts (drugs, genes, diseases, etc.) and to find edges in a knowledge graph that unify against a given subject-predicate-object triple, for example.[3]

While edgeo is a pure relation, ~cui-concepto requires that its first argument be a ground string. This is typical of the design of mediKanren—while we try to preserve relationality when possible, in some cases it doesn't seem useful or scalable to make a relation operate in "all modes."

As a first step towards answering the scientific question of Section 3, we ask what are the concepts that positively or negatively regulate RHOBTB2. This is a 1-hop query. We will see later how to do 2-hop queries.

```
(define rhobtb2-curie "UMLS:C1425762")

(run* (q)
  (fresh (db edge eid subject object pred eprops
          scid scui sname sdetails
          pred-id pred-name
          ocid ocui oname odetails)
    (== q `(,db ,sname ,pred-name ,oname))
    (== edge `(,eid ,subject ,object ,pred . ,eprops))
    (== pred `(,pred-id . ,pred-name))
    (== subject `(,scid ,scui ,sname . ,sdetails))
    (== object `(,ocid ,ocui ,oname . ,odetails))
    (conde
     ((== pred-name "positively_regulates"))
     ((== pred-name "negatively_regulates")))
    (~cui-concepto rhobtb2-curie `(,db . ,object))
    (edgeo `(,db . ,edge))))
```

The query uses miniKanren's unification to pattern match on an edge, a predicate, a subject, and an object. The matching uses the underlying list representation of these entities. Because the object is known when the edge query edgeo is called, the interface to the database will use the "by object" index.

This query produces 12 answers:

```
'((uab-pmi "E2F1 gene" "positively_regulates" "RHOBTB2 gene")
  (uab-pmi "E2F1 gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "Tumor Suppressor Genes" "negatively_regulates" "RHOBTB2 gene")
  (semmed "E2F1 gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "E2F1 gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "UBE2L3 gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "muristerone" "positively_regulates" "RHOBTB2 gene")
  (semmed "ERVK-10 gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "PRH2 gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "PR@ gene cluster" "negatively_regulates" "RHOBTB2 gene")
  (semmed "PGR gene" "negatively_regulates" "RHOBTB2 gene")
  (semmed "TMEM37 wt Allele" "negatively_regulates" "RHOBTB2 gene"))
```

This query is tedious and error-prone to write by hand—we will see a more abstract interface in the next section.

[3]The ~cui-concepto relation should really be named ~curie-concepto, since it is used to find concepts corresponding to a given CURIE (Compact Uniform Resource Identifier). In contrast to a CURIE, a *CUI* is a UMLS (Unified Medical Language System) concept identifier, which is used in SemMedDB. The name ~cui-concepto is a holdover from when mediKanren only used SemMedDB.

4.2 Medium-level Query Language

As can be seen in the code above, raw miniKanren is a low-level, and often clumsy, interface for expressing biomedical queries. To make it easier, faster, and less error-prone to write and read queries, mediKanren also supports a higher-level query interface, `query/graph`.

With this higher-level language, we can concisely express the two-hop query described in Section 3:

```
(query/graph
  (;; concepts
   (X       drug)
   (Y       gene-or-protein)
   (rhobtb2 "CUI:C1425762")
   (T       #f))
  (;; edges
   (X->Y       negatively-regulates)
   (Y->rhobtb2 positively-regulates)
   (X->T       drug-safe))
  ;; paths
  (X X->Y Y Y->rhobtb2 rhobtb2)
  (X X->T T))
```

The query asks for a safe drug that negatively regulates a gene or protein which in turn positively regulates the RHOBTB2 gene. The query is structured in three parts: declarations of concepts, edges, and paths. Concept and edge declarations include a name which will be used in path declarations, as well as a constraint. A path is comprised of concepts and edges, alternating between them.

Here we indicate that the `rhobtb2` concept set consists of a single element, identified by its CURIE. The concepts `X` and `Y` are constrained by sets of concept categories that abstract over the variation between category names in the various datasets (see Section 6.2). Similarly, we constrain each edge using a set of edge predicate names. The query uses two paths: one for the two-hops of downregulation and upregulation, and another for the drug-safe edge relating the drug to its trademark with FDA approval.

In general, a use of `query/graph` has the following structure:

```
(query/graph
  ((<concept-identifier> <curie-or-category>) ...)
  ((<predicate-identifier> <predicate> [<filter-procedure>]) ...)
  (<concept-identifier> <predicate-identifier> <concept-identifier>
   [<predicate-identifier> <concept-identifier>] ...) ...)
```

A `<curie-or-category>` must be a `<curie-string>`, or a list of `<category-string>`s, or `#f`, where `#f` stands for the full set of categories. A `<predicate>` must be a list of `<predicate-string>`s, or `#f`, where `#f` stands for the full set of predicates. A `<predicate>` may be followed by an optional procedure for filtering edges. The other substructures follow the definitions in Section 2.

The query above returns 11 results. Of the 11 "safe" drugs, 8 of them are scored with a "confidence" score of 0.925, including celecoxib. (This confidence score should not be taken too literally—this does not mean we have 92% confidence in the results! However, we can use relative confidence scores to help rank results.)

```
(0.925
  (("CHEBI:41423" . "CUI:C0812258") ("CUI:C0812258" . "CUI:C1425762"))
  (("celecoxib" . "E2f transcription factor 1")
   ("E2f transcription factor 1" . "Rho-related btb domain-containing protein 2")))
```

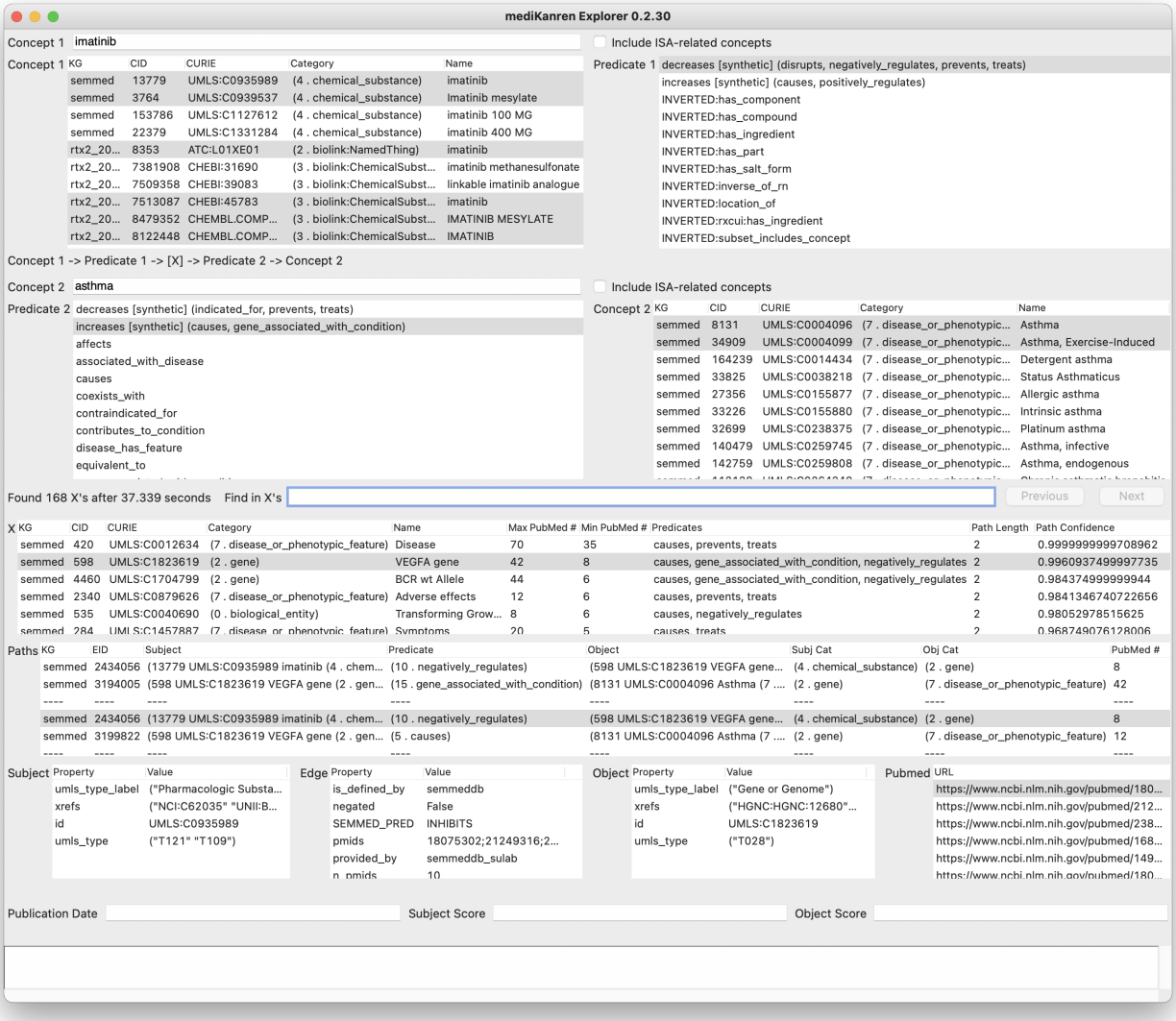

Fig. 2. mediKanren Graphical User Interface, showing a 2-hop query to determine if the cancer drug imatininb might treat asthma, through regulation of some unknown concept, "X." The user enters "imatinib" in the "Concept 1" text field, which populates the "Concept 1" list box with concepts containing the string "imatinib." The user selects several of these concepts, along with a set of predicates representing the notion of "Concept 1" "decreasing" the unknown concept "X." The user then enters "asthma" in the "Concept 2" text field, and selects two related concepts from the "Concept 2" list box, along with a set of predicates representing the notion of unknown concept "X" "increasing" "Concept 2." The list box "X" is then populated with 168 concepts that satisfy the query. The user selects a candidate "X" to inspect—in this case, the concept "VEGFA gene." The "Paths" list box is then populated with all the 2-hop paths that connect the selected imatinib-related concepts with the selected asthma-related concepts, through "VEGFA gene." The user selects the first edge in one of the 2-hop results: "imatinib negatively_regulates VEGFA gene." (The second edge in this 2-hop result is "VEGFA gene causes Asthma.") The boxes below the "Paths" list box include information about the edge, including clickable URLs for publications supporting that edge.

4.3 Graphical User Interface

As can be seen in Section 4, the query languages supported by mediKanren require knowledge of Racket and of relational programming. It is easy to make mistakes when hand-writing queries in these languages. Also, while expressive, these programmatic queries can take a while to write.

To support users who are non-programmers (or non-Racket/non-miniKanren programmers), we have created a graphical user interface (GUI) that is easy and fast to use. This GUI only supports a limited subset of mediKanren queries:

(1) 1-hop queries of the form *known subject concept S is related to unknown object concept O through known predicate P*;
(2) 1-hop queries of the form *unknown subject concept S is related to known object concept O through known predicate P*;
(3) and 2-hop queries of the form *known subject concept S1 is related to unknown concept X through predicate P1, and concept X is also related to object O2 through known predicate P2.*

Figure 2 is a screenshot of the mediKanren GUI, showing a simple 2-hop drug repurposing query provided by NCATS, which was described at the end of Section 2. This query is trying to determine if the cancer drug imatinib may also treat asthma through some unknown mechanism. Effectively, the query asks, "is there some concept X such that imatinib reduces the severity of/inhibits X, and where X causes/increases the severity of asthma?" One of the X's in this case is the gene VEGFA. The user can click on one of the PubMed identifier links in the lower-right of the GUI window to open up that paper in a Web browser.

5 RANKING GENE BUDGING RESULTS FOR A SET OF GENES

A common use of mediKanren in the Hugh Kaul Precision Medicine institute is to find FDA-approved drugs that "budge" the expression of a gene, either increasing or decreasing the amount of protein produced by that gene. This task is referred to as *gene budging*.

5.1 Simple Gene Budging

Our running query from Section 3 is an example of simple gene budging: we are looking to regulate a gene via a drug.

As described in Figure 1, the FDA-approved compound celecoxib that was returned is an indirect inhibitor of RHOBTB2. Upon sharing this result with the participants' physician, celecoxib therapy was initiated by admission into clinical trials. As of January of 2020, the participants have reported a decreased number of ataxic events, with an increase in focus and attention span.

5.2 Ranking Gene Budging Results for Acute Respiratory Distress Syndrome (ARDS)

Gene budging finds drugs that upregulate or downregulate some gene. In practice, we might have a profile of genes that are over-expressed and under-expressed, and we want to find drugs that downregulate the over-expressed genes and up-regulate the under-expressed genes. Done naively, this process can result in an explosive cocktail of drugs.

Instead, it helps to know which genes have the most regulatory influence over the list of abnormally expressed genes. This is done by determining the number of abnormal genes that another gene, which may or may not be on the list of interest, influences. The genes that regulate the most other abnormal genes and fewest number of normal genes can be targeted to minimize the number of genes that need to be budged directly in order to return all abnormally expressed genes to normal levels while simultaneously minimizing the number of unintended adverse effects. Furthermore, this regulation should be done per cell tissue type to provide more precise anatomical context for targeting.

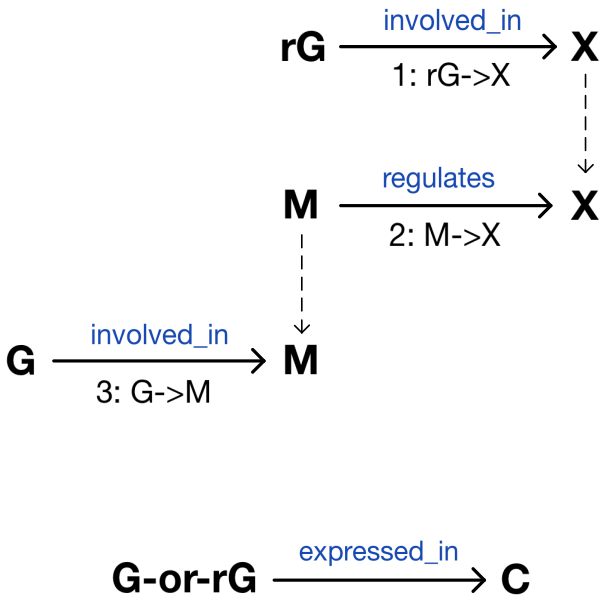

Fig. 3. Queries to find genes that regulate genes of interest. G are genes, M and X are pathways, rG are regulated genes (genes of interest), and C are cell/tissue types. Dotted arrows denote how one query feeds another. The flow starts at the top with the provided genes of interests rG. The first query out of the regulated genes rG finds the involved pathways X. The second query then finds the pathways M that regulate pathways X. Then the third query finds the genes G involved in pathways M. The indirection via pathways causes a loss of precision; however, the pathway information is very reliable, since it comes from the heavily curated Gene Ontology (GO). Finally, the independent bottom query finds the cell/tissue type C of each gene involved.

To implement ranked gene budging for a set of genes, we first use queries to extract the relevant data from the knowledge graphs. Subsequently we use Racket code to aggregate and sort information for each tissue type and regulating gene.

Figure 3 depicts the queries we use to extract the raw data. We start with a regulated gene *rG* of interest. We look at the pathways *X* involving the gene. We look at pathways *M* that regulate pathways *X*. We look at genes *G* that are involved in pathways *M*. We constrain per tissue type for each gene *G* or regulated gene *rG*.

The queries are as follow. We use `run/graph`, a variant of `query/graph` without concept normalization (see Section 6.2). We disable concept normalization to stay within the Gene Ontology (GO) [?], which contains curated information about genes, gene products, and pathways. We could have run all those queries in one big query, but it is more efficient to perform multiple smaller queries.

```
;; rG->X hop
(match-define
  (list rG/X=>concepts rG/X=>edges)
  (run/graph
    ((rG g) (X #f)) ;; g is a gene of interest
    ((rG->X involved_in))
    (rG rG->X X)))
```

```
;; M->X hop
(match-define
  (list M/X=>concepts M/X=>edges)
  (run/graph
    ((M #f)
     (X (hash-ref rG/X=>concepts 'X)))
    ((M->X (set-union positively_regulates negatively_regulates subclass_of)))
    (M M->X X)))
;; G->M hop
(match-define
  (list G/M=>concepts G/M=>edges)
  (run/graph
    ((G #f)
     (M Ms-in-GO))
    ((G->M involved_in))
    (G G->M M)))
;; query that finds cell expression information
(define q
  (query/graph
    ((G-or-rG g) ;; g is any regulator gene that was found by the previous query or
                 ;; any gene of interest in the given list
     (C #f))
   ((G-or-rG->C '("expressed_in")))
   (G-or-rG G-or-rG->C C)))
```

Loosely then, the genes *G* regulate the gene *rG*. For a particular gene *G*, we can do a more precise 1-hop query to confirm that it regulates gene *rG*, without the indirection of the pathways. Still, the multi-hop query through GO can find possible connections between genes and how they influence each other and/or a particular condition that may not have been studied before.

We process the query results using Racket code. For each tissue type, we create a map from each gene expressed in the tissue to a list of genes that the given gene regulates. If we sort the map by the number of genes regulated, this already gives us a good idea of the most regulating genes. Based on this map, we can rank gene budging results based on a number of criteria.

When we take the initial set of genes to be the genes involved in ARDS (Acute Respiratory Distress Syndrome), these ranked gene budging results has the potential to help COVID-19 patients, for whom ARDS can be fatal.

6 HARMONIZATION OF KNOWLEDGE GRAPHS

Multiple knowledge graphs (RTX, Orange, SemMedDB, ROBOKOP) each of which may reference overlapping source databases must be “harmonized” such that equivalent concepts and relations are identified. This yields a more computationally tractable graph and allows queries to traverse all accessible links that may have otherwise been missed due to source knowledge graph naming conventions. Currently mediKanren supports harmonization through the manual inspection of knowledge graph maps and the effective normalization of knowledge graph concepts and predicates. In the future, a complete harmonization will include meta-data such as the provenance of all input relations. For example, if an asserted gene-disease relationship is found in 2 input graphs (but based on the same publication) the harmonized knowledge graph should give such a relation a lower weight than if each knowledge graph derived its assertion from different publications.

6.1 Automatic Generation of Knowledge Graph Maps

To facilitate harmonization of input knowledge graphs, we assess the relevant concepts and relation classes of each input knowledge graph using maps. Initially, knowledge graph maps such as those in Figures 10 and 11 were hand drawn. While they look nice, they are time-consuming to create and they might be selective with respect to to the underlying knowledge graph triples.

For any knowledge graph, we can generate a map of triple types (subject prefix, predicate, object prefix), where a concept prefix takes into account the CURIE name up to the column. Now, we have a complete specification of what triple types are allowed by the knowledge graphs. Now, the question is how do we visualize the resulting map of triple types?

We are still looking into options, but we can compare a hand-drawn map of the Orange knowledge graph and a complete automatically generated map of the same knowledge graph in Figures 4 and 5. As we can see, the hand-drawn map is more readable; however, it is missing many concepts and edges. Inspection of these maps and the underlying knowledge graph data allows the creation of rules for concept and predicate expansion to effectively normalized graphs.

6.2 Normalization by CURIE and Predicate Expansion

Concept normalization traditionally involves the selection of a canonical representation of a concept or predicate to represent equivalent identifiers or synonyms for that concept. We achieve the same effect in mediKanren by collecting all known equivalent (but differently represented) CURIEs for a single concept by finding all identifiers for that concept across all publicly available databases. Comparing the two KGs, RTX and ROBOKOP, reveals a design similarity in using the HGNC CURIE to index gene concepts (see green highlighted CURIES in Figures 10 and 11); however, it also reveals critical differences. The ROBOKOP KG omitted cross-reference indexes to other commonly used gene indexes—NCBIGene, Ensembl, CUI, NCIT and MESH (Figure 11). Moreover, the ROBOKOP KG chose the PANTHER-FAMILY CURIEs to index protein concepts instead of UniProtKB. We effectively normalize all concepts in mediKanren's knowledge graphs by adding these omitted cross references.

Beyond concept CURIE differences, the graphs differ significantly with respect to predicate designations. ROBOKOP's `part_of` predicate (as opposed to RTX's encodes predicate) is used to traverse between a specific gene and its respective coding gene/protein family. Similarly with concepts, these equivalent predicates are added to the underlying knowledge graph. These differences may at first glance appear trivial, but the underlying differences in these graphs make cross-knowledge graph queries inoperable. We discuss further how concept and predicate normalization are currently performed in the implementation section (Section 7.1).

7 IMPLEMENTATION

Rather than build on an existing relational or graph database, we chose to implement our system using miniKanren and Racket, which are both flexible and expressive languages. We made this choice to reduce the risk of painting ourselves into a corner as we discovered requirements during development, as we were more confident in our ability to adapt miniKanren and Racket as necessary. In particular, the faster-miniKanren[4] implementation is fairly small and simple, making it quite hackable.

We use Racket to implement a simple graph database using a mixture of files containing either plain text s-expressions or data in a custom binary format. These files represent the underlying data, as well as indices supporting fast lookup needed for common queries over the graph structure. Aside from indices for graph structure lookup, we also include indices for full-text search over concept names, supporting responsive lookup in our GUI.

[4]https://github.com/michaelballantyne/faster-miniKanren

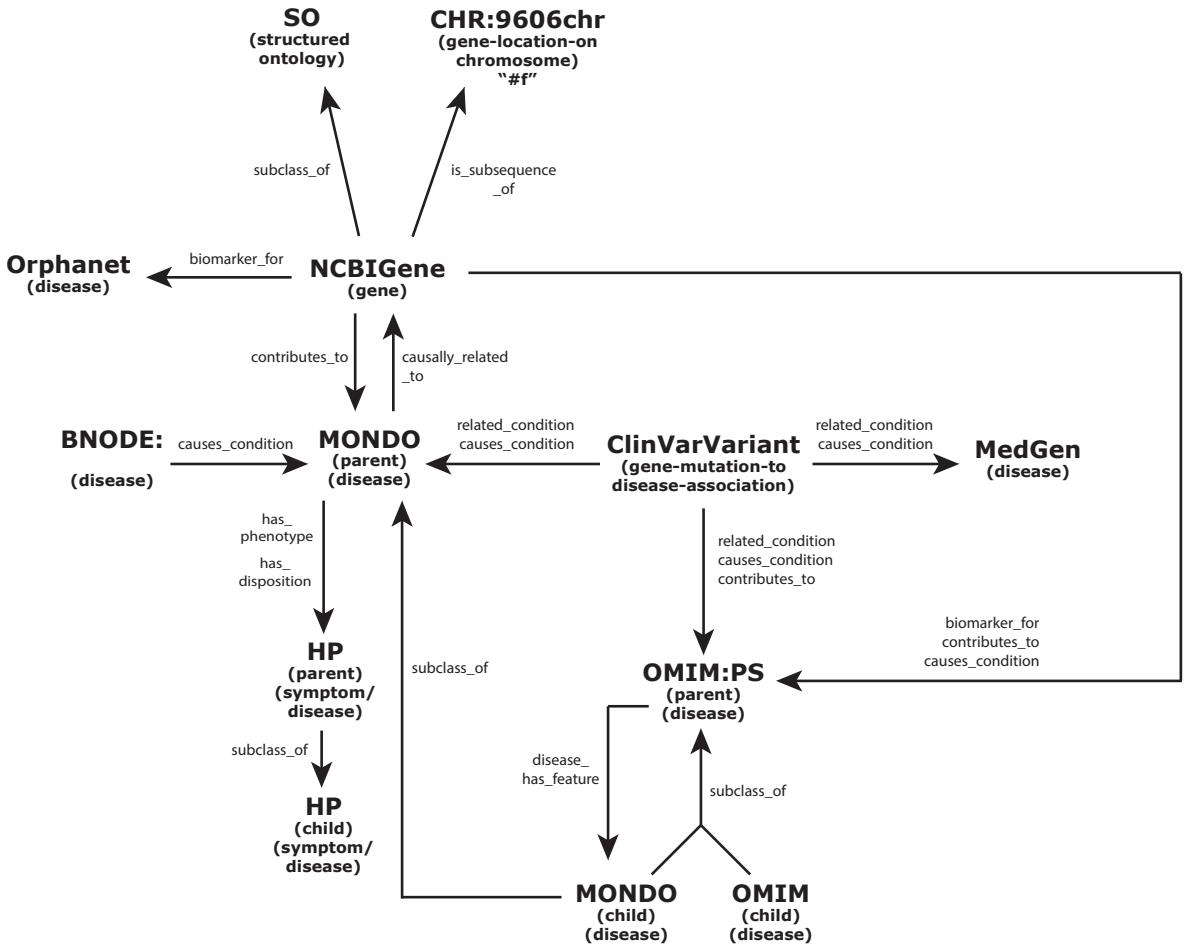

Fig. 4. Hand-drawn map of Orange knowledge graph.

The datasets we deal with, while many GB in size, are still small enough that an expensive computing cluster is unnecessary. In fact, mediKanren has been designed to run on fairly modest laptops to allow scientists to use their personal machines to explore the data without needing internet access or other supports. For this reason, our custom database implementation performs most data retrievals directly from disk without prefetching into RAM. This choice reduces both memory usage and startup time. Despite this tradeoff, mediKanren is still fast enough for interactive use during case reviews or other situations where it's necessary to perform many simple queries quickly. The simplicity of this database implementation has allowed us to quickly prototype new features, but trades off some performance and compact data encoding.

To improve interoperability, the knowledge graphs we work with have had their structure standardized according to the Biolink Model [Team 2020b]. This standardization allows us to ingest and process new data from different sources using the same pipeline. This pipeline begins by converting a data source to CSV (or TSV) format using the Knowledge Graph eXchange (KGX) tool [Team 2020a], which was developed by a collaborative

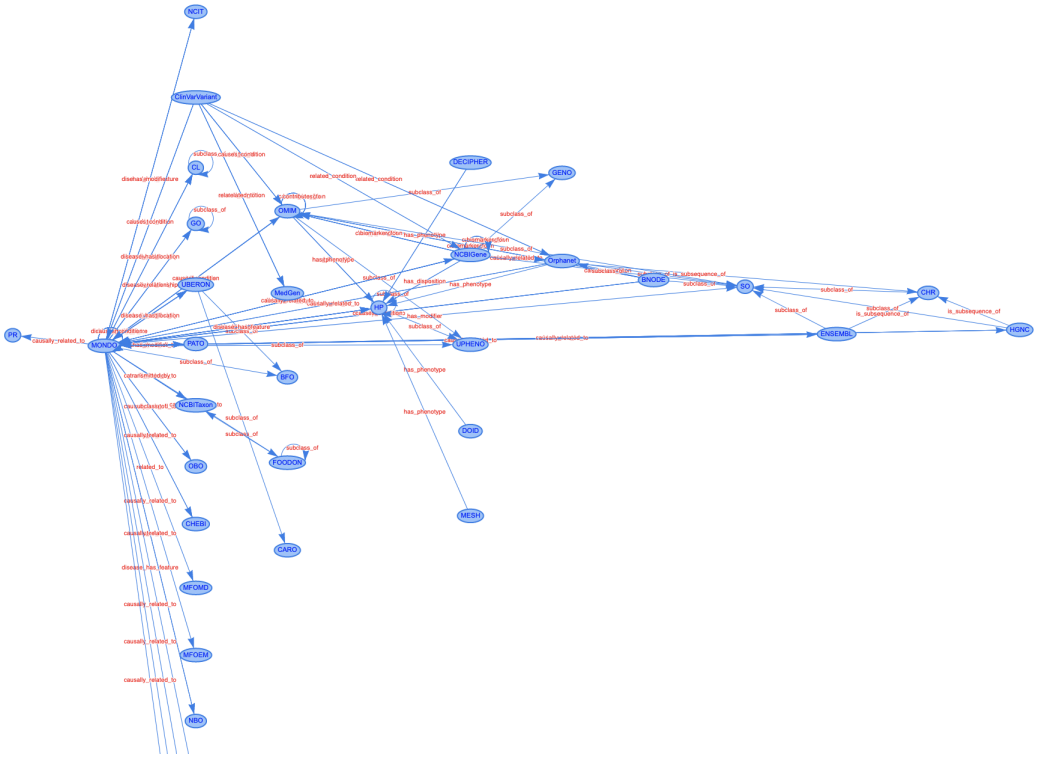

Fig. 5. Autogenerated map of the Orange knowledge graph. When compared with the hand-drawn map in Figure 4, it is obvious that we need to improve the layout of the map, and also need better ways to navigate the map. In practice, it is more useful at the moment to look at the text table summarizing the knowledge graph by type, from which the map is generated.

effort for general use. Given the resulting CSVs (or TSVs), we then run a second processing step specific to our implementation, to convert this input data to our database representation.

We expose concept and edge information to a miniKanren program by defining the relations `conceptO` and `edgeO`. These relations are defined using miniKanren's `project` construct, which allows us to implement them with Racket code that accesses our database representation, choosing an appropriate index to use based on groundness of different portions of the concept or edge argument.

7.1 Concept Normalization Implementation

As described in Section 6.2, we want to be able to write queries that connect data across knowledge graphs. Unfortunately, we cannot do this naively because many of the concepts we work with are given different CURIEs in each knowledge graph. Rather than normalizing each input graph to a canonical CURIE for each entity type (for example, using *only* HGNC CURIEs for all genes) we normalize concepts to equivalence classes. Each concept is assigned an equivalence class by computing the set of identifiers reachable from a concrete concept either through special properties or by following special edges (such as edges containing the Biolink predicate `same_as`), guided by manual inspection of the underlying source knowledge graph or generated knowledge graph map. These special properties and edges are defined using easy-to-change rules that allow us to experiment with different

notions of equivalence. Because concept normalization is an expensive operation, we support pre-building a cache of each concept's equivalence class.

7.2 Propagators

The `query/graph` DSL is implemented on top of a constraint propagation architecture inspired by Radul and Sussman [2009]. Concept and edge sets specified in a query are represented as cells containing sets of intermediate results coming from underlying miniKanren queries. These cells are connected by propagators that define constraints on those cells. Whenever a cell's constraint is updated, the set of results the cell contains may shrink. Anytime a set shrinks, any propagators watching that set's cell will be scheduled to update its constraints. This update process continues until quiescence, by which point all constraints will be arc-consistent. Final results are then extracted from cells by finding all globally consistent combinations.

8 FUTURE WORK

8.1 dbKanren

We have started a new implementation of our database with a more compact data representation, higher performance, and a more direct integration with miniKanren, including the ability to precompute and persist finite relations. The new implementation will also support stratified aggregation and computing fixed points, in the sense of Datalog, allowing us to reduce the amount of ad hoc Racket code we have to write when expressing complex queries. Data ingestion and processing will be possible to express relationally, rather than requiring ad hoc Racket code. At some point, we would also like to support temporally stratified relations, allowing us to describe dynamic systems, such as the UI layer, relationally as well, reducing the effort needed to prototype new UIs.

8.2 Ontologies

We aim to improve on intelligent uses of ontology data sources. Ontologies have been used extensively in biomedical and translational science to organize and structure knowledge with child-parent relationships (for example, symptoms comprising a disease) and/or functional class information (for example, a drug or gene classified by its mechanism of action or function) [Denny et al. 2018; Martínez-Romero et al. 2017]. A key issue in gathering information from ontologies are the inherent asymmetries that exists between data sources. Figure 6 outlines the structural differences in the Medical Subject Headings (MESH) and the Chemicals Entities of Biological Interest (CHEBI) ontologies [de Matos et al. 2010] with respect categorizing anticoagulant drug concepts. To date, mapping concepts across ontologies remains an ongoing struggle and area of active research [Groß et al. 2016]. In the future, we hope to develop a systematic way of identifying and reconciling differences across ontologies during import without losing data integrity.

8.3 High-Level Query DSL

We are creating a more intuitive query language based on feedback from biomedical researchers. Our existing query DSL designs are not ideal for biomedical researchers without experience in Racket or logic programming. Based on such users' feedback, we are improving the `query/graph` interface in several ways. We have designed a new syntax reminiscent of SQL, with a "select" clause for accessing node and edge attributes and a "where" clause for filtering results based on these attributes. Concepts and edges are specified in named clauses to make the syntax more readable to novices. The output of a query is structured as a list of rows for easy import to Excel for further analysis. The new DSL also checks syntax more carefully to provide informative error messages. We expect this DSL to elaborate to the next-generation mediKanren logic language. Experts will be able to mix

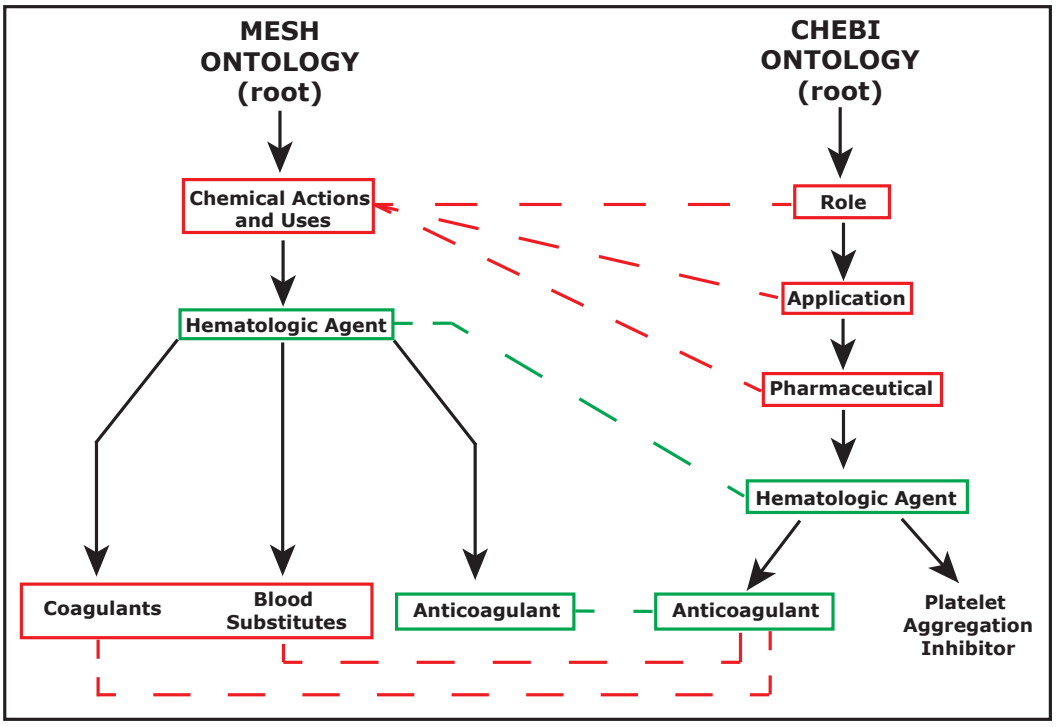

Fig. 6. MESH and CHEBI chemical ontologies display some similarities (dashed green) but many more structural differences (dashed red) for organization of drug class architecture.

high-level queries with lower-level relational code to express complex queries. Figure 7 shows how we imagine the query from Section 4.2 will be expressed.

Using the query language, we can search for over-represented biological motifs in the knowledge graph:

- confounders(A,B) = $A \longleftarrow ?X \longrightarrow B$
- colliders(A,B) = $A \longrightarrow ?X \longleftarrow B$
- feedback(A) = $A \longrightarrow ?X \longrightarrow A$
- feedForward(A,B) = $A \xrightarrow{v_1} ?X \xrightarrow{v_2} B, A \xrightarrow{v_3} B$

and enable some ranking over the query results, for example with a belief maintenance system introduced next.

```
(query/graph
  (select (curie X) (curie Y) (pubmed X->Y) (pubmed Y->rhobtb2))
  (concepts
    (X       drug)
    (Y       gene-or-protein)
    (rhobtb2 "CUI:C1425762"))
  (edges
    (X negatively-regulates Y       #:as X->Y)
    (Y positively-regulates rhobtb2 #:as Y->rhobtb2))
  (where (safe-drug X)))
```

Fig. 7. RHOBTB2 query using our future query language design. Sections 4.1 and 4.2 show the same query expressed using mediKanren's existing low-level and medium-level query languages, respectively.

8.4 Truth Maintenance Systems and Causal Reasoning

We are looking at non-monotonic reasoning based on truth maintenance systems [Doyle 1979; Forbus and de Kleer 1993] and belief maintenance systems [Falkenhainer 1988; Ramoni and Riva 1993].

Truth maintenance systems (TMS) have been incorporated in mediKanren for modeling the knowledge base as a directed graph with possible benefits of introducing the "beliefs" for an edge $A \rightarrow B$ (where A and B are concepts) in terms of the antecedents. These beliefs are essentially the support for and against a given assertion (for example, "imatinib cures cancer") and are incorporated by adding a layer of *belief maintenance* (BMS) on top of the TMS. The TMS, with its ability to identify the conclusions of a set of antecedents, helps in easily updating the beliefs. This has applications in drug ranking. For example, a query such as $A \rightarrow_{\text{increases}} X \rightarrow_{\text{cures}} B$ (A increases some intermediary X which cures B) could be ranked by the BMS in decreasing order of support, which would thus help identify the Xs having the greatest support for the assertion above.

In addition, we have extended truth maintenance systems to support deterministic and probabilistic causal reasoning so that we can leverage knowledge of regulatory mechanisms to draw inferences about causal effects [Pearl 2009].

To illustrate the kinds of reasoning that this extension permits, let's examine an example from biochemistry.Consider the following diamond motif in Figure 8, which was shown to be over-represented in mammalian signal transduction pathways [Ma'ayan et al. 2005]. In this motif, U is a stimulus, such as a signaling ligand, and C is a receptor. A and B are downstream signaling molecules, both of which are activated in response to U binding to C, and D is a transcription factor that can be activated by the activity of either A or B.

In this case, we would like to perform the following kinds of deterministic causal reasoning:

- **Prediction** ($\neg A \implies \neg D$): If signaling molecule A is not active, then transcription factor D is not active.
- **Abduction** ($\neg D \implies \neg C$): If transcription factor D is not active, then the receptor C is not bound to the ligand.
- **Transduction** ($A \implies B$): If signaling molecule A is active, then so is signaling molecule B.
- **Action** ($\neg C \implies D_A \wedge \neg B_A$): If the receptor C is not bound, but we intervene to activate A with a drug, signaling molecule B will not be active but the transcription factor D will still be activated.
- **Counterfactual** ($D \implies D_{\neg A}$): If transcription factor D is active, then it will still be active if we intervene to inhibit A with a drug.

We have further extended the TMS to handle probabilistic causal reasoning. For example suppose U can bind to C with probability p and A can spontaneously activate with probability q. We now wish to know $P(\neg D_{\neg A}|D)$,

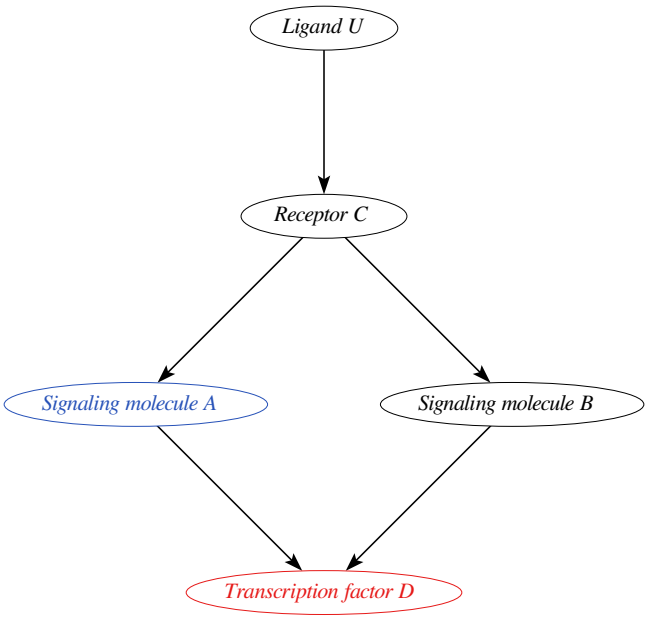

Fig. 8. Causal diagram for diamond signaling motif.

that is, the probability that if transcription factor D was observed to be active, it would not be active if we were to inhibit A.

As a further extension of the TMS, we would like to reason about cases where there may be unmeasured common causes of the variables in our model.

For example, consider the case in Figure 9 where we wish to predict the causal effect of some observable phenotype, such as cholesterol serum levels, on a disease such as cancer. Unfortunately, naive attempts to predict this causal effect may be biased because there are many common causes of both cholesterol levels and risk for cancer, such as diet, that could introduce confounding. However, if it is also known that certain alleles of a gene, such as apoE, are genetic determinants of low serum levels, then this knowledge can be used to predict an unbiased causal effect of cholesterol on cancer using a methodology called *Mendelian randomization* [Katan 2004].

To generalize this reasoning capability, we implemented sound and complete algorithms in the TMS that can identify which causal effects can be identified given a causal knowledge graph containing a set of observed and latent nodes [Shpitser and Pearl 2008]. In those cases where the causal effect is identified, the TMS will derive a formula for estimating the unbiased causal effect from observational data.

8.5 A Methodology for Improving COVID-19 Clinical Outcomes

COVID-19 has a complex etiology that starts in the lungs, and—through cytokine-induced inflammation and virus-induced endothelial tissue damage—activates the coagulation cascade, leading to thrombotic events [Voutouri

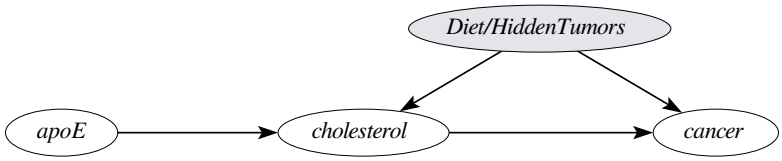

Fig. 9. A causal model with unobserved confounders (grey). Mendelian Randomization is one method for using prior knowledge networks to predict causal effects in the presence of unobserved confounders.

et al. 2021]. Doctors calibrate an implicit model of the COVID-19 disease pathway using clinical lab values to select medical countermeasures that guide the patient back to health. Clinical data from hundreds of thousands of de-identified patients has been collected and recently made available through NCATS National COVID Cohort Collaborative (N3C) [n3c [n.d.]]. We are currently using causal knowledge extracted from mediKanren to connect these clinical lab values to an explicit model of the COVID-19 disease pathway to account for potential sources of bias in the clinical data. By applying causal reinforcement learning to the N3C dataset, we aim to learn optimal dynamic treatment regimes that result in improved clinical outcomes [Zhang and Bareinboim 2020].

9 CONCLUSION

In this paper we introduced *mediKanren*, a combination of miniKanren, a graph database, knowledge graphs describing relationships between medical concepts, multiple query languages of varying degrees of abstraction, and a graphical user interface (GUI) to simplify data exploration and common queries. All features of the faster-miniKanren implementation are available for queries, including typical constructs like `run`, `conde`, `fresh`, various constraints, and (potentially recursive) user-defined relations. We provide the database as a set of miniKanren relations. To make queries fast, we represent the data backing these relations as specially formatted files on disk, with indexes for fast retrieval. We use the miniKanren `project` syntax to write Racket code that interfaces with this representation. Performance is sufficient for the GUI to support low-latency querying in the common case.

ACKNOWLEDGMENTS

Support for this work was provided by the National Center for Advancing Translational Sciences, National Institutes of Health, through the Biomedical Data Translator Program, awards OT2TR003435 and OT2TR002517. Any opinions expressed in this document are those of the Translator community at large and do not necessarily reflect the views of NCATS, individual Translator team members, or affiliated organizations and institutions.

Support for Marissa Zheng and Katherine Zhang was provided by the Harvard College Research Program (HCRP) through the Harvard College Office of Undergraduate Research and Fellowships.

Support for Jeremy Zucker and Joseph Cottam was provided by the PNNL Laboratory-directed R&D Data-Model Convergence Initiative. PNNL is operated for the DOE by Battelle Memorial Institute under Contract DE-AC05-76RLO1830.

We thank Jim Cimino and Jake Chen of UAB's Informatics Institute for their ideas and suggestions during the Translator Feasibility Phase.

We also thank Chris Austin, Christine Colvis, Noel Southall, and the rest of the NCATS for their vision and leadership of the Biomedical Data Translator Program. We also thank the other Translator teams for their support

and friendly working relationships. We have benefited especially from the advice of: Andrew Su at Scripps; Maureen Hoatlin and Matt Brush at OSHU; Chris Mungall at LBNL; Steve Ramsey at Oregon State University; Chris Bizon at RECNI/UNC, Melissa Haendel at the University of Oregon; and Paul Clemons at the Broad Institute. We are also grateful to all of the teams that have provided us knowledge graphs, beginning with Andrew Su and Greg Stupp, who supplied the initial Biolink-compatible versions of SemMedDB. The ROBOKOP knowledge graph from the ROBOKOP team at RENCI has also been invaluable. Steve Ramsey and his team at Oregon State University have created and refined multiple versions of their RTX 2 knowledge graph specifically for our needs.

Will Byrd, Greg Rosenblatt, and Michael Patton greatly benefited from visiting other Translator teams; we thank everyone for their hospitality, especially: Maureen Hoatlin, Melissa Haendel, and Matt Brush for our OSHU visit; Chris Mungall, Marcin Joachimiak, Deepak Unni, and Seth Carbon for our LBNL visit; and Paul Clemons, Jason Flannick, Vlado Dancik, Mathias Wawer, and Marcin von Grotthuss for our visits to the Broad Institute. We also thank Paul Clemons, Steve Ramsey, Melissa Haendel, Chris Bizon, and Eugene Muratov for visiting us at UAB. We thank the teams and institutions that hosted in-person Translator hackathons for their hospitality, as well: Scripps (La Jolla), RENCI (Chapel Hill), OSHU (Portland), NCATS (Bethesda), and ISB (Seattle).

We thank George Bonheyo for helpful edits. We also thank the anonymous reviews from the miniKanren Workshop for their helpful suggestions.

Nada Amin thanks Elena Glassman and Grzegorz Kossakowski for brainstorming.

Nada Amin and Will Byrd thank Gerald Jay Sussman at MIT for validating that mediKanren is a fine application for exploring and pushing the boundaries of truth maintenance systems.

We thank everyone who has used mediKanren and given us feedback over the past 3 years. The Biomedical Data Translator Programs's subject-matter experts have been especially helpful, especially Maureen Hoatlin from OHSU. The online mini-hackathons arranged by Karamarie Fecho from RENCI also helped us understand the strengths and limitations of our tools and knowledge sources.

REFERENCES

[n.d.]. About the National COVID Cohort Collaborative |National Center for Advancing Translational Sciences. https://ncats.nih.gov/n3c/about

William E. Byrd. 2009. *Relational Programming in miniKanren: Techniques, Applications, and Implementations.* Ph.D. Dissertation. Indiana University.

P de Matos, R Alcántara, and A Dekker et al. 2010. Chemical Entities of Biological Interest: An update. *Nucleic Acids Res* (2010), D249-D254. Issue 38. doi:10.1093/nar/gkp886.

P Denny, M Feuermann, DP Hill, RC Lovering, H Plun-Favreau, and P Roncaglia. 2018. Exploring autophagy with Gene Ontology. *Journal Biomedical Semantics* 14 (2018), 419–436. doi:10.1080/15548627.2017.1415189.

Jon Doyle. 1979. *A Truth Maintenance System.* Technical Report. MIT AI Memo 521.

Brian Falkenhainer. 1988. Towards a General-Purpose Belief Maintenance System. In *Uncertainty in Artificial Intelligence*, John F. LEMMER and Laveen N. KANAL (Eds.). Machine Intelligence and Pattern Recognition, Vol. 5. North-Holland, 125 – 131. https://doi.org/10.1016/B978-0-444-70396-5.50017-0

Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi, Eli Barzilay, Jay McCarthy, and Sam Tobin-Hochstadt. 2015. The Racket Manifesto. In *1st Summit on Advances in Programming Languages (SNAPL 2015) (Leibniz International Proceedings in Informatics (LIPIcs))*, Thomas Ball, Rastislav Bodik, Shriram Krishnamurthi, Benjamin S. Lerner, and Greg Morrisett (Eds.), Vol. 32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 113–128. https://doi.org/10.4230/LIPIcs.SNAPL.2015.113

Kenneth D. Forbus and Johan de Kleer. 1993. *Building Problem Solvers.* MIT Press, Cambridge, MA, USA.

Daniel P. Friedman, William E. Byrd, Oleg Kiselyov, and Jason Hemann. 2018. *The Reasoned Schemer* (2nd ed.). The MIT Press, Cambridge, MA, USA.

A Groß, C Pruski, and E Rahm. 2016. Evolution of biomedical ontologies and mappings: Overview of recent approaches. *Comput Struct Biotechnol J.* 14 (2016), 333–340. doi:10.1016/j.csbj.2016.08.002.

Arif E. Jinha. 2010. Article 50 million: an estimate of the number of scholarly articles in existence. 23 (2010), 258–263. doi:10.1087/20100308.

M B Katan. 2004. Apolipoprotein E isoforms, serum cholesterol, and cancer. 1986. *International Journal of Epidemiology* 33, 1 (feb 2004), 9. https://doi.org/10.1093/ije/dyh312

Avi Ma'ayan, Sherry L Jenkins, Susana Neves, Anthony Hasseldine, Elizabeth Grace, Benjamin Dubin-Thaler, Narat J Eungdamrong, Gehzi Weng, Prahlad T Ram, J Jeremy Rice, Aaron Kershenbaum, Gustavo A Stolovitzky, Robert D Blitzer, and Ravi Iyengar. 2005.

Formation of regulatory patterns during signal propagation in a Mammalian cellular network. *Science* 309, 5737 (12 aug 2005), 1078–1083. https://doi.org/10.1126/science.1108876

Marcos Martínez-Romero, Clement Jonquet, Martin J. O'Connor, John Graybeal, Alejandro Pazos, and Mark A. Musen. 2017. *Journal Biomedical Semantics* 8 (2017), 21. doi: 10.1186/s13326-017-0128-y PMCID: PMC5463318.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference* (2nd ed.). Cambridge University Press, USA.

Alexey Radul and Gerald Jay Sussman. 2009. The art of the propagator. In *Proceedings of the 2009 International Lisp Conference*. 1–10.

Marco Ramoni and Alberto Riva. 1993. Belief maintenance with probabilistic logic. In *Proceedings of the AAAI Fall Symposium on Automated Deduction in Non Standard Logics, Raleigh, NC*.

Page RDM. 2019. Ozymandias: a biodiversity knowledge graph. *Scientific Reports* 7 (2019), e6739. doi:10.7717/peerj.6739.

C Ross, H Empinado, and R Robbins. 2019. An AI expert's toughest project: writing code to save his son's life - STAT. https://www.statnews.com/2019/07/25/ai-expert-writing-code-save-son/.

M Rotmensch, Y Halpern, A Tlimat, S Horng, and D Sontag. 2017. Learning a Health Knowledge Graph from Electronic Medical Records. *Scientific Reports* 7, 1 (2017), 5994. doi:10.1038/s41598-017-05778-z.

B. Shepard. 2019. Diagnosis in 2.127 seconds: Solving a years-long vomiting mystery using AI, research and brain power - News. https://www.uab.edu/news/health/item/10703-diagnosis-in-2-127-seconds-solving-a-years-long-vomiting-mystery-using-ai-research-and-brain-power.

Ilya Shpitser and Judea Pearl. 2008. Complete Identification Methods for the Causal Hierarchy. *Journal of Machine Learning Research* 9, 64 (2008), 1941–1979. http://jmlr.org/papers/v9/shpitser08a.html

Biomedical Data Translator Tangerine Team. 2020a. Knowledge Graph Exchange tools for Biolink Model compliant graphs. https://github.com/NCATS-Tangerine/kgx.

Biolink Model Team. 2020b. Biolink Model. https://biolink.github.io/biolink-model/.

Chrysovalantis Voutouri, Mohammad Reza Nikmaneshi, C Corey Hardin, Ankit B Patel, Ashish Verma, Melin J Khandekar, Sayon Dutta, Triantafyllos Stylianopoulos, Lance L Munn, and Rakesh K Jain. 2021. In silico dynamics of COVID-19 phenotypes for optimizing clinical management. *Proceedings of the National Academy of Sciences of the United States of America* 118, 3 (19 jan 2021). https://doi.org/10.1073/pnas.2021642118

W3C. 2010. CURIE Syntax 1.0: A syntax for expressing Compact URIs. https://www.w3.org/TR/curie/.

Junzhe Zhang and Elias Bareinboim. 2020. Designing Optimal Dynamic Treatment Regimes: A Causal Reinforcement Learning Approach. PMLR. http://proceedings.mlr.press/v119/zhang20a.html

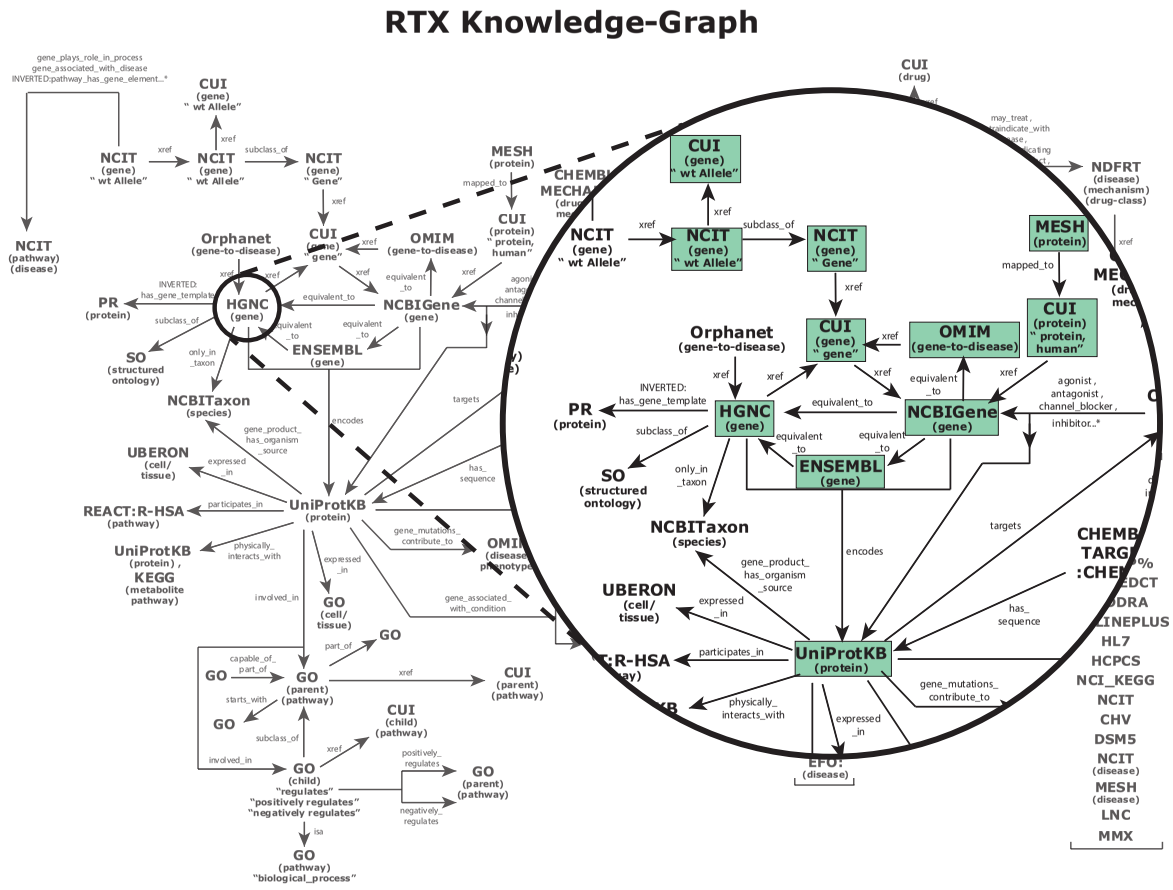

Fig. 10. Zoomed-in and green highlighted concepts represent gene and protein concept specific CURIES, and the predicate paths that connect them. Concept types are noted in parenthesis below each database name (bold). Edge predicates are located above each arrow, directing the relation between concepts.

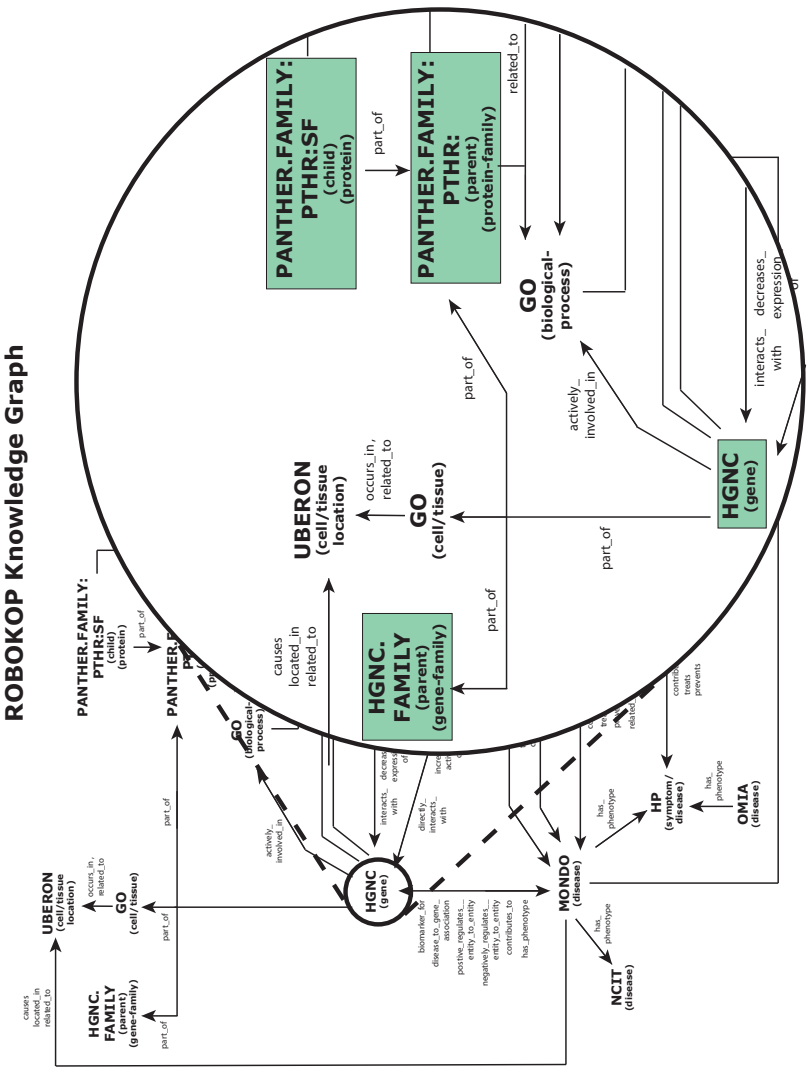

Fig. 11. Zoomed-in and green highlighted concepts represent gene and protein concept specific CURIES, and the predicate paths that connect them. Concept types are noted in parenthesis below each database name (bold). Edge predicates are located above each arrow, directing the relation between concepts.

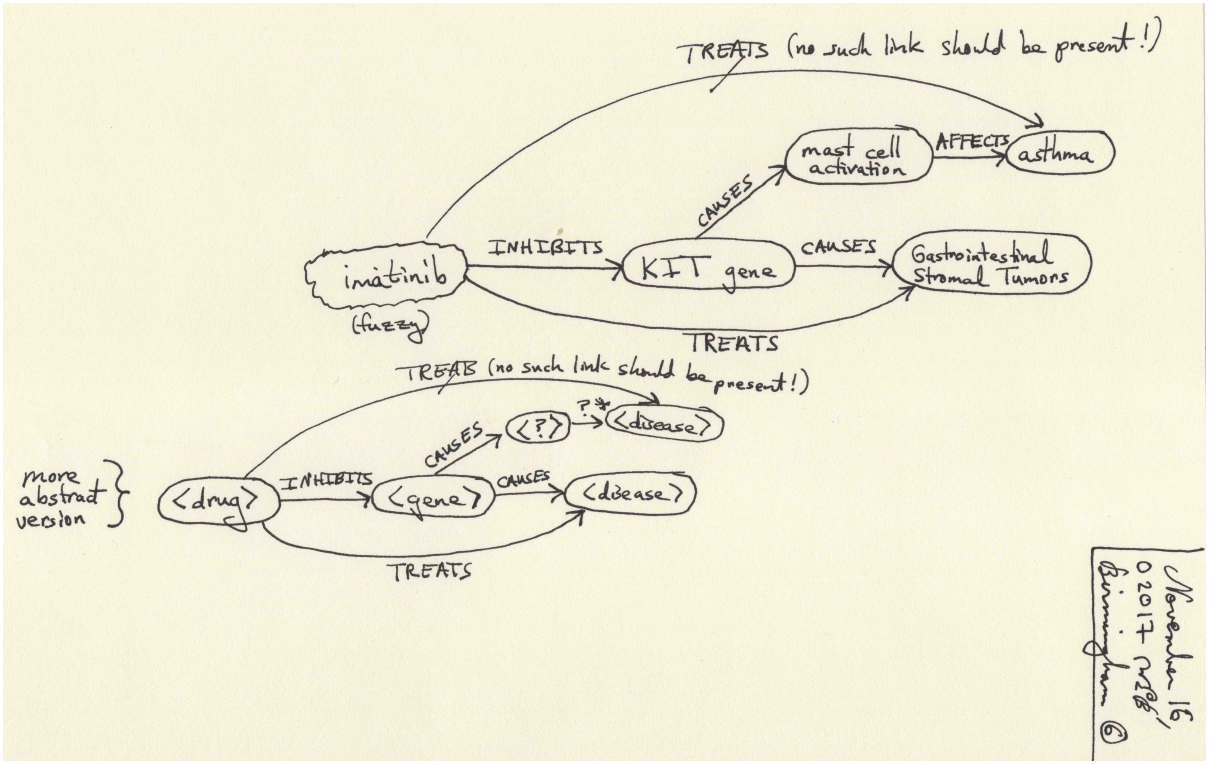

Fig. 12. Hand-drawn diagram from the early development of mediKanren. The upper-part of the diagram shows a subgraph from the Semantic MEDLINE Database (SemMedDB), based on the NCATS-provided drug-repurposing example query: "is there evidence that the cancer drug imatinib also treats asthma?" Even though imatininb is not known to treat asthma (at least in the medical literature), we can weigh the strength of the evidence that imatininb might treat asthma through some known mechanism of action, such as by inhibiting expression of a gene implicated in causing asthma. The subgraph shows that drugs in the "imatinib" class are known in the literature to treat gastrointestinal stromal tumors, and that (one) mechanism of action for this treatment is inhibition of KIT gene, whose over-expression is known to cause the tumors. KIT gene is also known to cause mast cell activation, which affects asthma. The overall subgraph provides evidence that imatinib might treat asthma, inferring potential new knowledge. Any algorithm weighing this evidence should take into account that the "affects" predicate is weaker than the "causes" predicate, and that the path from imatinib to asthma through KIT gene is one "hop" longer than the path from imatinib to gastrointestinal stromal tumors. The lower-part of the diagram shows a generalized schema of the concrete imatinib subgraph, which can be used as the basis of a query for discovering drug-repurposing candidates.

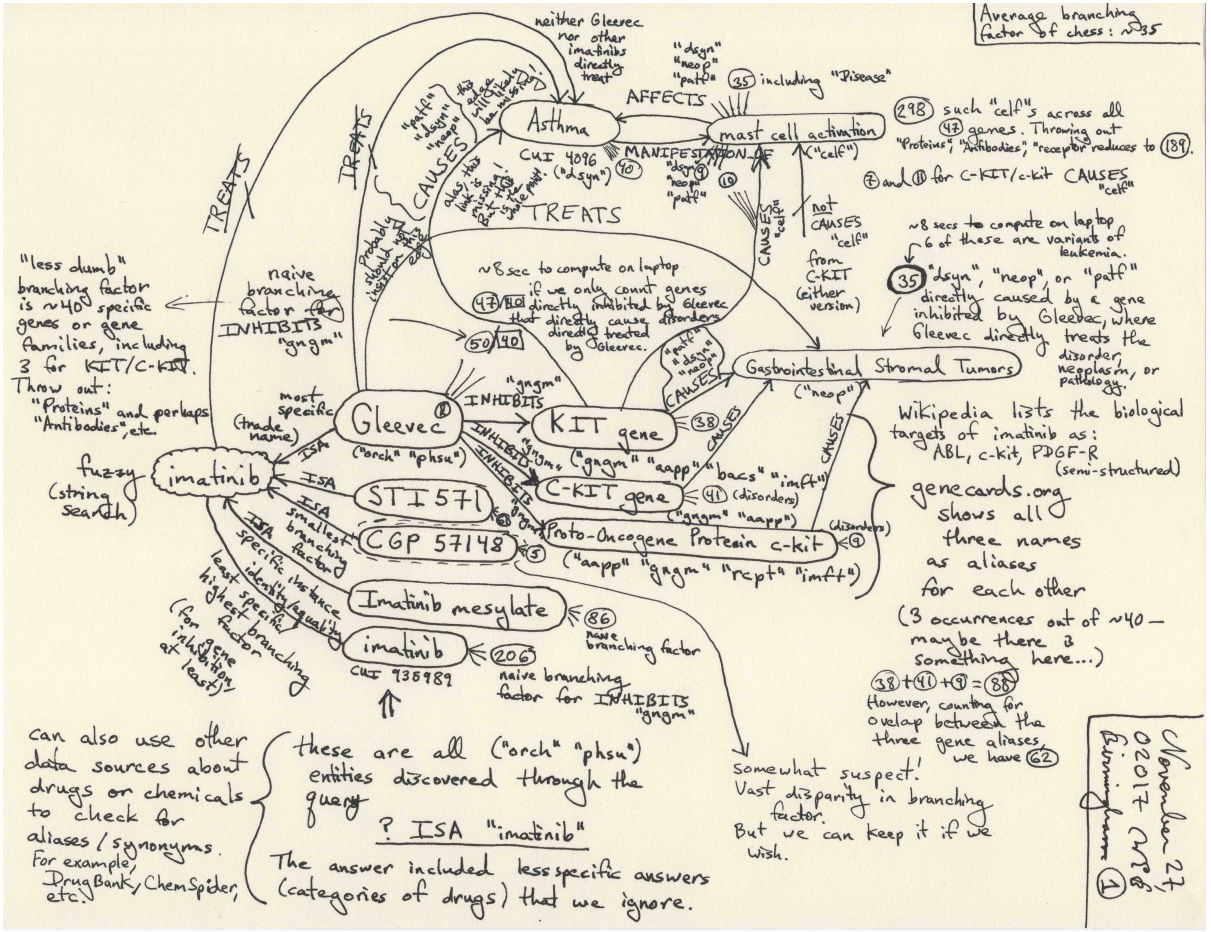

Fig. 13. Hand-drawn diagram from the early development of mediKanren. This diagram is a refined version of the Semantic MEDLINE Database (SemMedDB) subgraph from Figure 12, based on the NCATS-provided drug-repurposing example query: "is there evidence that the cancer drug imatinib also treats asthma?" This expanded subgraph shows different drugs that might match the search term "imatinib," along with multiple genes implicated in causing gastrointestinal stromal tumors. The figure also shows the branching factors of the edges that start with each node in the subgraph: for example, the C-KIT gene is known to cause 41 different disorders. For several nodes the branching factor is over 200; in comparison, the average branching factor in a game of chess is around 35.