English Access to Structured Data

Kyle D. Richardson[1], Daniel G. Bobrow[1], Cleo Condoravdi[1], Richard Waldinger[2], Amar Das[3]
[1]Palo Alto Research Center, Palo Alto, CA; {krichard, bobrow, condorav}@parc.com
[2]SRI International, Menlo Park, CA; waldinger@ai.sri.com
[3]Stanford University, Stanford, CA; das@stanford.edu

Abstract— **We present work on using a domain model to guide text interpretation, in the context of a project that aims to interpret English questions as a sequence of queries to be answered from structured databases. We adapt a broad-coverage and ambiguity-enabled natural language processing (NLP) system to produce domain-specific logical forms, using knowledge of the domain to zero in on the appropriate interpretation. The vocabulary of the logical forms is drawn from a domain theory that constitutes a higher-level abstraction of the contents of a set of related databases. The meanings of the terms are encoded in an axiomatic domain theory. To retrieve information from the databases, the logical forms must be instantiated by values constructed from fields in the database. The axiomatic domain theory is interpreted by the first-order theorem prover SNARK to identify the groundings, and then retrieve the values through procedural attachments semantically linked to the database. SNARK attempts to prove the logical form as a theorem by reasoning over the theory that is linked to the database and returns the exemplars of the proof(s) back to the user as answers to the query. The focus of this paper is more on the language task; however, we discuss the interaction that must occur between linguistic analysis and reasoning for an end-to-end natural language interface to databases. We illustrate the process using examples drawn from an HIV treatment domain, where the underlying databases are records of temporally bound treatments of individual patients.**

Keywords: Natural language processing, Natural language interfaces to databases, Deductive question answering, Theorem proving, HIV drug resistance database.

I. INTRODUCTION

The dream of the semantic web is to provide access to the world's knowledge. Google and other search engines have made it easy to search for textual documents that are likely to contain statements that answer a user's query—implicit in keywords, or even laid out as a question. However, much of the information available online is in the form of structured databases rather than unstructured text.

Users wishing to access information in databases confront many barriers. A user may not know what sources exist, or how the information is encoded in any given source. Answers may not be present explicitly in any one source but may need to be inferred or computed from information obtained from several sources. Users must, in any case, be proficient in a database query language, such as SQL. As an alternative, natural language requires no special training and allows naïve users to compactly express complex information that would otherwise be hard to express in SQL or other formal query languages. Using natural language to access structured information has long been the goal of research on Natural Language Interfaces to Databases (NLIDB), although work in this area has diminished since the mid-1980s, in part because of the difficulty of the task [2], [7].

A major challenge for building an NLIDB is developing a high-precision system for linguistic analysis with enough coverage to handle a wide range of NL queries. As a workaround, recent work has limited itself to 'semantically tractable queries' [22] or pursued alternative 'hybrid' NLP approaches [13], [14], [15]. Even within more principled NLP approaches [11], [12], it is common to attempt to map natural language directly to SQL, in spite of well-known limitations on the expressive power of SQL [2, p.22] and the gap between the vocabulary of the query and that of the database.

In this paper, we describe our work in customizing a broad-coverage, general-purpose NL system for building an NLIDB. The premise is that by limiting the domain of discourse, we can use the semantics of the domain to enable interpretation of a much wider range of queries in noncontrolled English. Our domain of application is HIV treatment using the Stanford HIV Drug Resistance Database [24]. We have built a prototype system, called Quadri [3], [27], which answers English questions on the basis of the information in the Stanford databases. The databases we access are temporal [19], and describe drug regimens and tests that have been part of the treatment protocol for HIV. These curated databases are publicly available, but have seen less use than was expected by researchers and clinicians in the field. By providing an NL interface, we hope to significantly reduce the barriers to accessing this information and make the databases more widely available.

The databases provide the grounding for the domain. They provide tables of regimens, where a regimen for a given patient comprises a set of drugs and has associated with it a start date and an end date. They also specify times and results for tests given to the patient (e.g., the number of copies per milliliter of the virus in the blood, and characteristics of the genotype of the virus). However, NL questions by clinicians are posed at a more abstract level than asking directly for the retrieval of specific data. Although these questions can be answered with reference to grounded fields in the database, the desired information is often described qualitatively instead of quantitatively. The output of the linguistic analysis then would require extra reasoning to get at the corresponding values, since queries often include qualitative notions that need to be recast in quantitative terms, which is how the information is stored in the database. It is for this reason that we separate out the language analysis task from the reasoning and retrieval task. The linguistic analysis component is responsible for providing a high-level, 'implementation-independent' [12] description of the domain-specific information provided in the NL query. The reasoning and retrieval component is later responsible for fully aligning the language with the database structure and going into the database and pulling out the desired information requested in the NL query.

For instance, in our selected subject domain, we might have the query *Find all patients on a failing regimen for at least 30 weeks. The patients exhibited M184V.* Given the domain knowledge, we want to expand this to a conjunction of several conditions that the retrieved patients must satisfy: the patients must have had a regimen (or drug treatment) which eventually failed. Furthermore, we have a temporal constraint: the regimen must have a duration of *at least* 30 weeks. The second part of the query is construed as a filter: it seeks those patients satisfying the first part who, in addition, had a genotype test that indicated the presence of the mutation M184V. While some of this analysis can be embedded into the natural language component, some of it requires domain knowledge that goes beyond natural language understanding. A *failing regimen* in the domain model, for example, is one where the patient has a viral load test whose value is high after a certain amount of time from the onset of the regimen. The value for a viral load test is, in turn, high if its numerical result exceeds some specified threshold. In the data source this is simply indicated by a numerical viral load measurement for the patient and a time stamp. The fact that the viral load test occurred during the regimen has to be computed on the basis of the time stamps associated with the test and the regimen. The connection between a descriptive classification of regimens and the properties with which a regimen must be associated in a database in order to fall under a particular classification category is not part of the linguistic analysis but of the subsequent reasoning phase, carried out by the theorem prover.

The idea of using an underlying specification of the database model to zero in on a domain-specific NL interpretation is a key part of the Quadri system. We use the general-purpose, ambiguity-enabled NLP system BRIDGE [4], which parses and maps NL queries to an abstract knowledge representation (AKR). By providing an abstract specification of the database model, which identifies sortal relations in the database, temporal information, and other relevant facts, we build a custom component on top of BRIDGE that maps queries from AKR to an unambiguous domain-specific logical form. This is passed to SRI's theorem-prover SNARK [25], which is equipped with an axiomatic theory of the HIV drug-resistance domain that serves as our data model. SNARK attempts to prove the theorem (or theorems), invoking *procedural attachments* that provide a semantic link to the various data resources. SNARK also has special procedures that accelerate temporal reasoning. In what follows, we describe each component in turn and show how it applies to sample problems.

II. Language Analysis

A. Mapping to Domain-Specific Logical Forms

As an initial step, an English question is linguistically analyzed. For the language task, we are using a deep understanding system called BRIDGE [4], which provides a general-purpose linguistic analysis. In other words, it provides generic information about the language and preserves ambiguity throughout the parsing and interpretation process. Since our goal is to map unrestricted English to a precise logical form that can be used by a first-order theorem prover, using a robust system like BRIDGE that is already equipped to handle a variety of linguistic phenomena is essential. The chief challenge then is fitting the system to the target domain, and this requires integrating knowledge of the domain and using this information to guide the interpretation process.

To do the fitting, we develop a *Language-Use Model* (LUM). A LUM is an abstract model of the subject domain that provides a list of the sorts of objects in the domain, along with a specification of how designated expressions in the language map to these objects and relations. The customization of BRIDGE involves building a LUM, and going from a general-purpose semantic representation to a domain-specific representation. Crucial to our LUM is a set of *argument signatures,* which are specifications of the relations that occur in the domain. These provide frames, loosely speaking, for selecting local patterns that support the domain-specific interpretation.

Argument signatures are defined in terms of domain concepts and domain relations, which include concepts like *patient, medical-test, treatments*, and *retroviral-drug,* and relations such as *regimen-contains-drug,* and *patient-has-test.* Corresponding to the domain concepts are vocabulary items in the domain that must be identified. Relations are defined in terms of the sort of their arguments. They might be related to other relations and have additional properties. In our domain, we are particularly interested in temporal relations. These are indicated by linguistic structure, including tense, aspect, prepositional phrases, and specific keywords. Implicit concepts are also included, allowing limited reasoning to occur on the language side, e.g., knowing that M184V indicates the existence of a genotype test in the first example.

The LUM is integrated into the BRIDGE pipeline, as shown in Figure 2. The initial component of the pipeline is a finite-state machine that recognizes named entities [16]. We have augmented these entities to include specialized biomedical vocabulary and multiword expressions such as *M184V (mutation)* and *viral load (medical-test)*. The general-purpose parser then creates a dependency structure that identifies linguistic functions and arguments using the LFG language engine [18]. All syntactic ambiguities are found and preserved using a compact notation. Semantic processing then occurs to normalize the dependency structure using an XFR rewrite system described in [10]. These structures are later mapped to an Abstract Knowledge Representation [5],[9], which has facilities for time and date, entailment and contradiction detection (ECD), among other features (see [4]). The LUM is

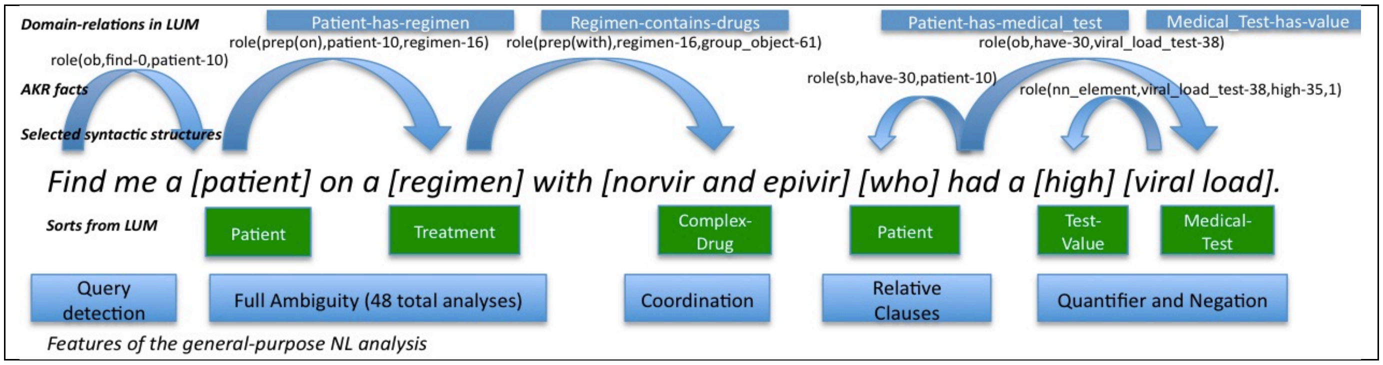
Domain-relations in LUM
Patient-has-regimen
Regimen-contains-drugs
Patient-has-medical_test
Medical_Test-has-value
role(ob,find-0,patient-10)
role(prep(on),patient-10,regimen-16)
role(prep(with),regimen-16,group_object-61)
role(ob,have-30,viral_load_test-38)
role(sb,have-30,patient-10)
role(nn_element,viral_load_test-38,high-35,1)
AKR facts
Selected syntactic structures
Find me a [patient] on a [regimen] with [norvir and epivir] [who] had a [high] [viral load].
Sorts from LUM
Patient
Treatment
Complex-Drug
Patient
Test-Value
Medical-Test
Query detection
Full Ambiguity (48 total analyses)
Coordination
Relative Clauses
Quantifier and Negation
Features of the general-purpose NL analysis

Figure 1: An example query with selected patterns

used on top of the full BRIDGE analysis as a filter. Domain-specific terms are identified in the analysis, and local syntactic patterns that correspond to pairs satisfying domain events are selected and rewritten into domain-specific relations. Alternative structures are discarded.

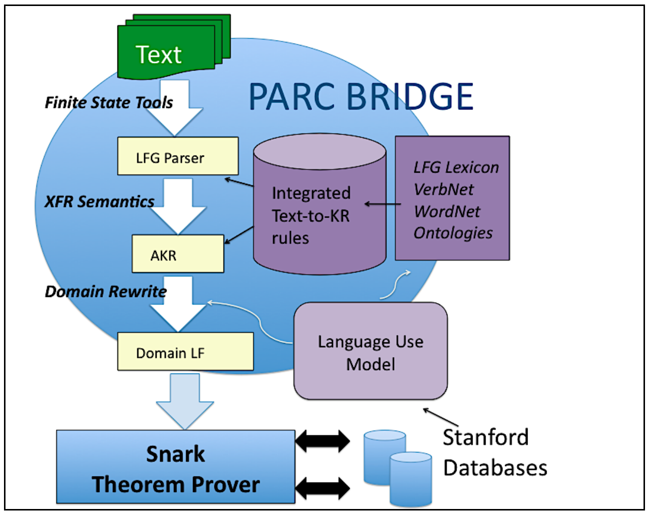

Figure 2: Quadri architecture

Figure 1 provides an analysis of an example query. Particular words in the sentence are identified and associated with a domain sort specified in the LUM. Syntactic patterns indicated by the blue arrows are chosen, since the sorts of the corresponding terms relate to underlying relations in the domain listed above. In this case, the system initially produces 48 analyses, and alternative analyses not interpretable in the domain are eliminated. On the edges of the blue arrows are the corresponding AKR facts that relate the terms. Note that relative clauses and coordination, among other complex linguistic phenomena, are independently handled in the system, and interpreted appropriately in the AKR. *Who,* for example, is resolved to refer to the *patient*, and conjunction between *Norvir* and *Epivir* is recognized to be a *group object.*

As another example, in the first query we have the fragment *patients on a failing regimen*, which can be syntactically analyzed either as [patients] [on a failing regimen] or [patients [on a failing regimen]]. In the LUM, there is the relation *patient-has-regimen* between a *patient* and a *treatment.* Here *regimen* is identified as an instance of a *treatment*, and the two terms are linked by the preposition *on*. Alternative analyses, including the possible attachment of *failing regimen* to the verb *find,* are eliminated, and the particular relation is derived since the preposition in this case is consistent with the interpretation. Similarly, the LUM might indicate that treatments can *fail,* which has the consequence in this example of identifying *failing regimen* as an instance of this relation in the language.

B. Temporal and Implicit Concepts

Temporal information plays a critical role in the drug resistance domain, since most queries relate to treatment intervals and events (loosely speaking) that occur at different stages of these intervals. A crucial part of the LUM then is information about the temporal properties of these events, including whether they are *punctual* (i.e., occurring at a point in time), *durative* (i.e., occurring over an interval of time), and how events are related to other events in time. Here, too, we provide frames that help select the appropriate interpretation. These frames come in two varieties. Some relate explicit events and temporal concepts expressed in the language. Others relate events with temporal concepts left implicit by the language but overtly represented in the logical form.

An example of the first type is illustrated in the fragment *patients on a failing regimen for at least 30 weeks.* A regimen (treatment) is a durative event that has a start date and an end date in the database. Since it is durative, the LUM indicates that it can be associated with a time interval that specifies the duration of the desired treatment. The system therefore looks for a pattern in the language between a treatment and a time interval, and finds an analysis that links the regimen with the interval *week* via the preposition *for.* Alternative analyses are eliminated, and the preposition *for* is interpreted as indicating the duration of the regimen. The modified numeral *at least 30* is independently interpreted in the BRIDGE system as a complex cardinality on *week*, and is normalized to be 'greater than or equal to'. This allows us to conclude in the analysis that the *treatment* occurred over some time period, and that the duration of this period is greater than or equal to 30 weeks.

Punctual events have implicit time points associated with them. Having a *medical test*, for example, is associated with a date stamp in the database. In order to bring out this fact, we expand the relation to have a time variable. The time of the test might also relate to a specified point in a given time period. The intended meaning of the query *The patient had a high viral load after 24 weeks on the regimen* is that the patient had a medical test measuring viral load shortly after the first 24 weeks of the regimen. The frame indicates a relation between a medical test and a time period, and the system attempts to find an instance of this pattern. The generated time point associated with the test is then placed in relation with the interval. We also want to conclude that the *24 weeks* is part of the total duration of the regimen. This fact is derived independently in the BRIDGE system in a special module that handles data and time specification. Figure 3 gives a portion of the AKR analysis with this information. In bold are the selected role relations [5] in the AKR that provide the basis for the domain relations, which are indicated in blue. Alternative analyses are struck through to indicate that they are eliminated.

Query: Find patients with a high viral load after 24 weeks on a regimen.
Choice Space: xor(A1,A2,A3,…,A45) iff 1 *(45 analyses or 'choices')*
(or(A32,or(or(or(or(A33,A34),A27,A25),A26….)…. *(place in 'choice space')*
  role(cardinality_restriction,week-26,24) ***(Time-Period, Cardinality)***
    ➔ ***interval-has-duration(week-26,24 weeks)***
or(or(or(A45,or(A41,A42),A39)….)….
  role(nn_element,viral_load_test-16,high-11,1) ***(Medical-Test, Test-Value)***
    ➔ ***medical-test-has-value(viral_load_test-16, high-11)***
~~or(or(or(A28,A29),A24),A23):~~
~~role(prep(after),patient-7,24-22)~~
~~or(or(or(A42,A43),or(A38,A39),A33,A32),A10,A9):~~
~~role(prep(after),patient-7,week-26)~~
~~or(or(A30,A31),A22,A21):~~
~~role(prep(after),viral_load_test-16,24-22)~~
~~or(or(or(A41,or(A44,A45)),A40,or(A35,A36,A37),A34)…~~
  role(prep(after),viral_load_test-16,week-26) ***(Viral-load, Time-Period)***
    ➔ ***medical-test-has-time(viral_load_test-16, time_point-1)***
    ➔ ***occurs-after(time_point-1, week-26)***
or(A36,or(or(or(A27,A28,A29),…)
  role(prep(during),week-26,regimen-37) ***(Time-period, Treatment)***
    ➔ ***occurs-during(week-26,regimen-37)***
~~or(or(A17,A18),or(or(A13,A14),A11,A12),A4,or(A1,A2,A3)):~~
~~role(prep(with),find-1,viral_load_test-16)~~
~~or(or(or(or(A28,A29),A24,….)~~
  role(prep(with),patient-7,viral_load_test-16) ***(Patient, Medical-Test)***
    ➔ ***patient-has-test(patient-7, viral_load_test-16)***
…)

Figure 3. AKR analysis of a query

Temporal prepositions such as *after* and *before* are interpreted as particular places in time relative to the intervals they modify. More complex modifiers include *at the end of X,* or *at the start of X,* and pinpoint particular places within interval *X*. Combinations of these types of modifiers also occur, such as *after the end of X, near the start of X,* and so on. The LUM provides a schema for how to represent this information (similar in spirit to [20]). In *The patient had a viral load after the end of 24 weeks,* we are trying to identify a place in time that is *after* an interval of *24 weeks.* As before, we know that it is the time point of the genotype test that is located at this particular space in time given the frame that relates the medical test with the time period.

The language analysis is filled out in other ways using the LUM. For example, some events expressed in the language are part of larger events that we want to identify. Having a mutation M184V means having a genotype test that indicates the M184V mutation. This is detectable in the database, since the field corresponding to the test result or mutation is embedded into the larger medical-test table. The same is true for drugs, which always have a related treatment. If the query does not explicitly mention a treatment, we generate a treatment variable in order to make this explicit. In *The patient failed norvir after 24 weeks,* we assume that there is an implicit treatment variable containing the drug *Norvir* and that the treatment is at least *24 weeks* long. In general, these expansions are done in order to align as closely as possible the language with the database structure, which the LUM ultimately models.

C. Quantifiers and Multiple Sentences

Quantifiers and logical connectives are an important part of doing a high-precision analysis, and quantifier scope must be specified in order to do the subsequent theorem-proving. We must be able to distinguish *Patients not all of whose regimens contain Norvir from Patients all of whose regimens do not contain Norvir*, for example.

In the AKR analysis, semantic facts are represented as "flattened" clauses [9], [10]. The scope of operators like negation that is grammatically fixed is specified in the flat representation, but the relative scope of quantifiers that is not grammatically fixed is not specified. The nesting information therefore needs to be built out of these structures. Figure 4 shows the AKR for the example *Every patient is on some regimen with Norvir*, and the full domain-specific interpretation with quantifier structures is displayed.

In the AKR, quantifiers are specified as cardinality restrictions on the terms. Terms that do not have overt quantifiers are treated by default as existential. From the cardinality relation, a quantifier structure is built that relates each quantifier with its term, and specifies the type of the term. "Scopes-over" relations are created that describe nesting

Every patient is on some regimen with norvir.

AKR
role(cardinality_restriction,norvir-1,mass)
*role(cardinality_restriction,patient-2,**all**(pl))*
*role(cardinality_restriction,regimen-3,**some**(sg))*
role(copula_subj,be-3,patient-2)
~~*A1:*~~
~~*role(prep(with),be-3,norvir-1)*~~
A2:
role(prep(with),regimen-3,norvir-1)
role(tprep(on),patient-2,regimen-3)

DOMAIN-INTERPRETATION
((top_level patient_2 2)
(quant exists norvir_1 sort drug)
(quant all patient_2 sort patient)
(quant exists regimen_3 sort treatment)
(exists_group ex_grp_4 (regimen_3 norvir_1))
(scopes_over nscope patient_2 ex_grp_4)
(in nscope ex_grp_4 (**patient-has-regimen** patient_2 regimen_3))
(in nscope ex_grp_4 (**regimen-has-drug** regimen_3 norvir_1))….)

Figure 4. Domain interpretation from AKR

between the quantified terms. Scope ambiguity is resolved by the domain information. In the relation *patient-has-regimen*, patient is treated as the head of the relation in the frame, and is therefore assumed to outscope whatever its modifier is. This leads to the interpretation that each patient had that patient's

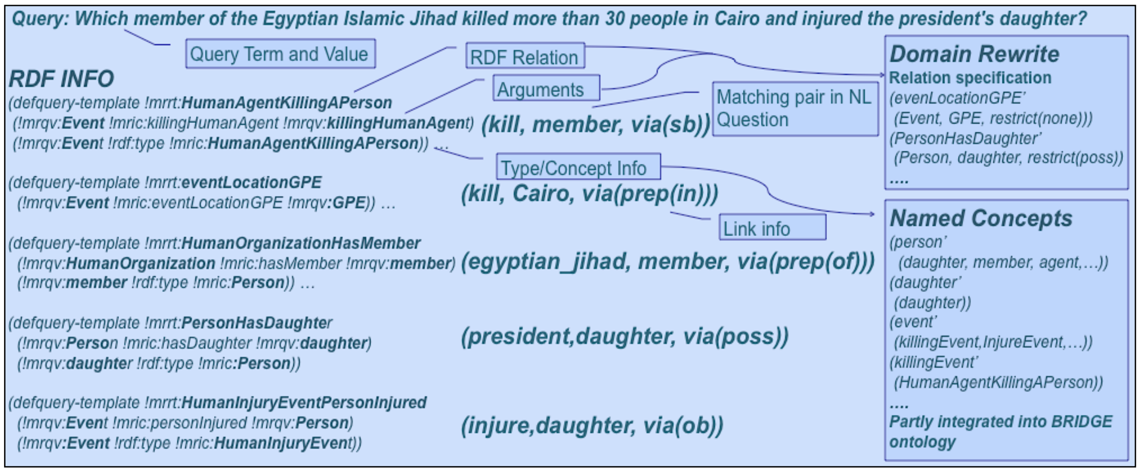
Query: Which member of the Egyptian Islamic Jihad killed more than 30 people in Cairo and injured the president's daughter?
Query Term and Value
RDF Relation
Arguments
Matching pair in NL Question
Type/Concept Info
Link info
RDF INFO
(defquery-template !mrrt:HumanAgentKillingAPerson
(!mrqv:Event !mric:killingHumanAgent !mrqv:killingHumanAgent)
(!mrqv:Event !rdf:type !mric:HumanAgentKillingAPerson)) …
(defquery-template !mrrt:eventLocationGPE
(!mrqv:Event !mric:eventLocationGPE !mrqv:GPE)) …
(defquery-template !mrrt:HumanOrganizationHasMember
(!mrqv:HumanOrganization !mric:hasMember !mrqv:member)
(!mrqv:member !rdf:type !mric:Person)) …
(defquery-template !mrrt:PersonHasDaughter
(!mrqv:Person !mric:hasDaughter !mrqv:daughter)
(!mrqv:daughter !rdf:type !mric:Person))
(defquery-template !mrrt:HumanInjuryEventPersonInjured
(!mrqv:Event !mric:personInjured !mrqv:Person)
(!mrqv:Event !rdf:type !mric:HumanInjuryEvent))
(kill, member, via(sb))
(kill, Cairo, via(prep(in)))
(egyptian_jihad, member, via(prep(of)))
(president,daughter, via(poss))
(injure,daughter, via(ob))
Domain Rewrite
Relation specification
(evenLocationGPE'
(Event, GPE, restrict(none)))
(PersonHasDaughter'
(Person, daughter, restrict(poss))
….
Named Concepts
(person'
(daughter, member, agent,…))
(daughter'
(daughter))
(event'
(killingEvent,InjureEvent,…))
(killingEvent'
(HumanAgentKillingAPerson))
….
Partly integrated into BRIDGE ontology

Figure 5. Information extracted from an RDF specification of another domain

own drug, as opposed to there being a unique drug that each patient has, which is not the intended meaning. Since the different orderings of existentials does not cause ambiguity, we place all the existential variables into a list, which the patient variable scopes over in this case. The relations are placed within the *nscope* (nuclear scope) over the existential list.

Quantifier scope ambiguity continues to be a major area of research in computational semantics, and underspecification formalisms, such as *Minimal Recursion Semantics* (MRS, [8]), have become the standard approach. The basic idea is to rely on a single compact and underdetermined semantic representation, rather than having to enumerate each fully specified interpretation [21]. From these underspecified representations, particular interpretations can be selected (or constraint solved, [6]) in accordance with the context. Our approach is distinct, in that we avoid having any information about quantifier scope in the AKR, which eliminates the need for developing an underspecified representation altogether. Quantifier information is represented uniformly throughout each AKR analysis, and nesting structures are built later using the domain model directly. Depending on the domain, quantifier scope for a single sentence can still be interpreted differently.

Scopes-over assertions are also used to specify the scope of negation, and detect definiteness. Definiteness is an important feature in doing multi-sentence queries, which our system can handle. Multi-sentence queries allow a user to incrementally specify constraints on the answers that user wants to see. Definite reference (e.g. saying *the patients*, or *those regimens*) is used to link later elements to those mentioned earlier. Quantified concepts mentioned in a first sentence are remembered as possible targets for definite reference. Definite references create a new quantified variable, but because it is indicated as definite (referring to a previously known element), a search is made backward to find the most recent mention of a variable of the same sort. In the linguistic analysis, we create an expression `equal(target-variable, definite-variable)`. The meaning of this is that the target variable must satisfy all the constraints specified for the definite variable. In the description about the theorem prover, we explain how we implement this.

D. Portability and Future Work

Porting to a new domain requires being able to create a LUM for that domain. Since the LUM is largely an abstraction of the data source, many argument signatures can be inferred from the database schema. For example, the relation *patient-has-regimen* can be inferred from there being a join relation between the patient and treatment tables (see [14] for details on how to generate queries in this way). However, the domain vocabulary must be identified, and in specialized domains such as medicine, many of the words (e.g, M184V) do not occur in the basic WordNet ontology and must be manually added or learned from a corpus of documents.

We have done an experiment in porting our system to a new domain [23]. In the DARPA-sponsored Machine Reading project, an ontology has been provided (encoded in RDF) dealing with terrorists and events in which they may be involved (demonstrations, bombings and killings). We used the RDF class definition triples that provide a specification of the domain relations and their arguments, to construct relation-argument signatures for this new LUM. The concepts in this domain are more common and most have corresponding elements in our WordNet-based taxonomy (e.g., for *Geo-Political-Entity, Person, Human-Organization*). The event hierarchy was more specialized, with events such as *HumanKillingEvent,* with roles of *AgentKilling* and *PersonKilled*. Using the Quadri system with minimal modifications except for the change of ontology (and adding 39 new relation-argument signatures) we successfully ran a number of examples.

Figure 5 provides an example query in this domain with the RDF entries. In the RDF template, relations are specified with their argument types. A *HumanInjuryEventPersonInjured* relation, for example, occurs between a *Person* and a *HumanInjuryEvent.* By including this information in BRIDGE, our system finds local patterns that link these two concepts together. In the example query, "Which member of … *injured of the president's daughter* …" the word *injure* is recognized as indicating a *HumanInjuryEvent,* and *daughter* is a person, which fits the RDF signature.

Relations in the domain often have associated restrictions. A *person-has-daughter* relation is keyed by a possessive relation in the parsed structure (i.e., a possessive verb, or genitive). It is not keyed by just having a local pattern between two words in the sentence. If you say *John knows the daughter*, it does not follow that John has a daughter. In this case, the RDF naming conventions were uniform enough to give clues into what the restrictions are in the language. If a relation has the word *have* in it, we assume that in the language we need to look for a possessive pattern between the concepts. So, in addition to providing information about domain relations, we were able to extract some information about the language structure, which is a key part of the LUM. Further work will look at other ways of inferring the restrictions on relations in the language from domain corpora, and in general on learning LUMs from other sources.

III. Reasoning Component

The deductive component of Quadri consists of the theorem-proving system SNARK, equipped with an axiomatic theory of the HIV drug-resistance domain. Although in principle any sufficiently powerful theorem prover can play this kind of role, SNARK [25], a first-order theorem prover specially intended for applications in software engineering and artificial intelligence, is particularly appropriate. It contains many of the most successful features of automatic theorem provers, including resolution (for general reasoning), paramodulation (for reasoning about equality), and term rewriting (for representing definitions and simplifications), as well as procedures that perform accelerated inference for selected concepts (e.g., numerical computation, temporal and spatial reasoning). We rely on SNARK's mechanisms for answer extraction (obtaining answers to questions from proofs) and procedural attachment. A sort mechanism keeps track of the sorts of all terms and prevents a sorted variable from being replaced by a term of an incompatible sort; this rules out many dead ends in the search. SNARK has strategic control features that allow us to tailor it to exhibit high performance in a selected subject domain. It is mature software that has been applied in a number of successful systems (e.g., NASA's Amphion [17], and SRI's Quark [28])

Temporal reasoning is of particular importance in medical subject domains. SNARK has a version of the Allen calculus [1] for reasoning about temporal relations between entities. While the original Allen calculus deals only with temporal intervals, SNARK's version deals with both time points and time intervals. Time points can be dates and times, and a procedural attachment can perform date and time arithmetic. This is quite a bit faster than if the computation were carried out axiomatically. Let us examine the behavior of the reasoning component of Quadri on a simple example. Then we can turn to a multisentence example.

A. Simple Example.

Consider the query *What patients had a high viral load?* (This question is simpler than a researcher is likely to ask; virtually all patients in the database have a high viral load at some point.) The semantic representation for this sentence includes the following:

```
top_level(patient_5, 1)
quant(exists patient_5 sort patient)
quant(exists viral_load_test_2 sort viral_load_test)
exists_group (ex_grp_6 (patient_5 viral_load_2 ...))
in nscope (ex_grp_6
  patient-has-test(patient_5, viral_load_test_2))
in nscope (ex_grp_6
  test-has-value(viral_load_test_2, high_3)))
```

This tells us that patient_5 and viral_load_test_2 are both existentially quantified and have the relations patient-has-test and test-has-value within their scope.

From this and other components of the semantic representation, Quadri constructs a conjecture:

```
exists(patient_5 sort patient)
  exists(viral_load_test_2 sort viral_load_test)
    exists(high_3 sort test_result)
      patient-had-test(patient_5, viral_load_test_2) &
      test-has-value(viral_load_test_2, high_3) &
      within-range(high_3, high)
```

In other words, we must show the existence of a patient, a viral load medical test, and a test value such that the patient had the viral-load-test with a numerical value that was within the range regarded as high. This is the conjecture that is passed to SNARK.

SNARK employs its axiomatic theory for the HIV-drug-resistance subject domain with appropriate procedural attachments to prove that the conjecture is a theorem in this theory. The answer-extraction mechanism keeps track of the substitutions made for all the existentially quantified variables in the conjecture necessary to complete the proof; these constitute the answer that is extracted. In general, there will be many proofs for the same theorem, each of which may yield a different answer.

In particular, the relation patient-has-test has a procedural attachment to the Stanford HIV Drug Resistance Database. This database knows the list of medical tests for each patient; the procedural attachment yields an identifier for each patient and that patient's corresponding medical tests.

One procedural attachment reveals that patient Mr. A had a viral load test AV881130 (the data in this discussion is not real). Another indicates that the result of this test was a viral load of 5, on the logarithmic scale. The proof, however, is not completed by procedural attachment alone; it requires the use of axioms in the HIV drug resistance theory. The axiom

```
in-range(viral-load, log-scale(?number)) ⇔ ?number >= 4
```

tells us that, in the logarithmic scale, 5 is regarded as high, since it is more than 4. (Note that symbols with question marks are variables, which can be replaced by other terms during the proof. The name of the variable indicates its sort; e.g. ?number is a variable of sort *number* and can be replaced only by a term, such as *5*, which is also of sort number.)

When the proof is complete, SNARK can extract an answer. For this proof, the desired answer is Mr. A. SNARK will also report the values found for the other existentially quantified variables in the theorem, i.e., the identifier of the test, AV881130, and its result, 5 on the logarithmic scale. Other proofs will yield different answers. Most of these correspond to other patients, but some will correspond to other viral load tests that Mr. A has taken.

B. Multipart Example.

Consider the multipart query we have used as an example: *Find all patients on a failing regimen for at least 30 weeks. The patients exhibited M184V.* The sentence is transformed by Bridge into a semantic representation, which is a flat, unordered set of conditions that describe the logical form(s).

The initial set of forms for the query is:

```
top_level(patient_3, 5)
top_level(patient_2, 45)
quant(exists patient_3 sort patient)
quant(exists patient_2 sort patient)
definite(patient_2)
equal(patient_3, patient_2)
```

This initial set tells Quadri that, corresponding to the two parts of the query, there are two logical forms, one with top-level quantifier exists(patient_3*)* and the other with top-level quantifier exists(patient_2*),* each of sort patient. The indexes on the top-level variables, 5 and 45, respectively, tell Quadri that the logical form for patient_3 must be solved before the logical form for patient_2, because its index is less. Another condition in the representation, definite(patient_2)*,* tells us that variable patient_2 is the same as an earlier variable in the query, and the condition equal(patient_3, patient_2) says that patient_2 is actually the same as patient_3. This corresponds to the linguistic intuition that the patients referred to in the second part of the query, *The patients exhibited M184V*, are the same as those mentioned in the first part, *Find all patients on a failing regimen for at least 30 weeks*.

Other conditions describe the scoping of the quantifiers and their relationship to the atomic propositions of the logical form. For instance, the condition

```
in scope(ex_group_10,
          patient-has-regimen(patient_3, regimen_4))
```

tells us that a certain group of quantifiers (defined elsewhere) has the proposition *patient-has-regimen* in its scope.

From this information, Quadri pieces together the first logical form,

```
exists(patient_3 sort patient)
   exists(regimen_4 sort regimen)
      exists(week_5 sort time-interval)
         patient-has-regimen(patient_3, regimen_4) &
         failing(regimen_4) &
         duration(regimen_4, week_5) &
time_measure(week_5,complex_card(>=,30), week)
```

In other words, we must seek a patient with a failing regimen whose duration is at least 30 weeks. This logical form is passed as a conjecture to be proved by SNARK.

For instance, Mr. A has two regimens, Regimen A.1 and Regimen A.2. Each of these is returned to SNARK, which will resume two separate branches of the proof search, with regimen_4 replaced by Regimen A.1 and Regimen A.2, respectively. The relation time_measure has no procedural attachment itself, but the condition

```
time_measure(week_5, complex_card(>=, 30), week))
```

can be transformed according to the following axiom (a rewrite rule) in the domain theory:

```
time_measure(?time-interval,
                  complex_card(>=, ?number),   ?unit)
  ⇔
 duration(?time-interval) >= ?unit(?number)
```

taking ?time-interval to be week_5*,* ?number to be 30, and ?unit to be week. The resulting formula is

```
duration(week_5) >= week(30)
```

i.e., the duration of time interval week_5 must be at least 30 weeks. This formula is transformed by other axioms in the axiomatic theory; for instance the duration of a time interval is the arithmetic difference between its finish point and its start point, i.e.,

```
duration(?time-interval) =
  finish-time(?time-interval) - start-time(?time-interval)
```

While the database does not store the duration of each regimen explicitly, it does know its start date and finish date. Once the time interval week_5 is replaced by a concrete time interval, with dates as its end points, procedural attachments will find its endpoints, compute its duration, and check if it is greater than 30 weeks. As it turns out, Regimen A.1 is less than 30 weeks long, but Regimen A.2 is longer.

As a result, Patient Mr. A is returned by SNARK as one of many answers to the first part of the query, with Regimen A.2 as the specified failing regimen.

The second part of the query requires us to discard from the set of answers patients who do not exhibit the mutation M184v, corresponds to the second logical form,

```
exists(patient_2 sort patient)
   exists(genotype_test_7 sort genotype_test)
     exists(time_point_8 sort time-point)
       exists(m184v_1 sort mutation)
         patient-has-test(patient_2, genotype_test_7) &
         test-has-value(genotype_test_7, m184v_1) &
         m184v_1 = m184v &
         test-has-time(genotype_test_7, time-point_8) &
         patient_2 = patient_3.
```

In other words, we seek a patient who has had a genotype test on a certain date that revealed the presence of a mutation M184v.

Note that the logical form specifies that patient_2 = patient_3, because the semantic representation told us that patient_2 must be equal to patient_3. Note that the variable patient_3 is outside the scope of the quantifier exists(patient_3… from the first logical form. This second logical form is conjoined with the first logical form and submitted to SNARK. Special treatment is given to the quantifiers during this conjunction to ensure that the variable is pushed within the scope of the appropriate quantifier.

The list of answers for the multipart query includes Mr. A, along with his failing regimen (A.2) and the date for his genetic test. The user may then ask further questions, to restrict the set of answers still further or to request additional information.

IV. CURRENT STATUS

The Quadri prototype is now capable of handling queries at the level of the examples in this paper, which includes limited anaphoric reference and multi-sentence questions. It provides feedback to the user of the translation of the logical form, can prove the associated theorems, and can query a snapshot of the database to identify cohorts of patients that satisfy stated user criteria. Next steps include enabling users to provide feedback to choose among alternative interpretations, and querying the databases in full.

Our planned evaluation of the Quadri system encompasses multiple dimensions. The first is the scope of queries that can be handled, and how these cover what potential users would want to ask. For this purpose, we have gathered questions from researchers at the Stanford Biomedical Informatics Research group. We have also collected statements from articles that described HIV cohorts being used for clinical trials, and adapted them so that they could be answered from the database set. The second dimension is the language coverage, that is, how easy it is to use language that the system would understand. For this we had people create different paraphrases of a number of the queries on our first list. The third dimension is correctness – that is, did the system return all and only those patients specified in the query. For this we have created a gold standard based on an expert creating and running an SQL query for some of the queries.

ACKNOWLEDGMENTS

We thank Robert Shafer and Soo-Yon Rhee for consultation on HIV Drug Resistance in general and for information about the Stanford HIV Drug Resistance Database in particular; Mark Stickel for sharing expertise on the use of the SNARK theorem proving system; Will Bridewell for comments, suggestions, and discussion. This work was supported by Award Number RC1LM010583 from the National Library of Medicine. The content is soley the responsibility of the authors and does not necessarily represent the official views of the National Library of Medicine or the National Institutes of Health.

REFERENCES

[1] Allen, J. "Maintaining knowledge about temporal intervals," *Communications of the ACM.* 1983.

[2] Androutsopoulos, I. et al. "Natural language interfaces to databases," *Natural Language Engineering,* 2(1):29-81, 1995.

[3] Bobrow, D.G. et al. "Deducing answers to English questions from structured data," in *Proc. IUI'11,* 2011.

[4] Bobrow, D.G. et al. "PARC's BRIDGE and question answering system," in *Proc. GEAF '07,* 2007.

[5] Bobrow, D.G. et al. "A basic logic for textual entailment," in *Proc. AAAI Workshop on Inference for Textual Question Answering,* 2005.

[6] Burchardt, A. et al. "Computational Semantics," course text, ESSLLI 2004. http://www.coli.uni-saarland.de/projects/milca/courses/esslli04/

[7] Copestake, A. et al. "Natural language interfaces to databases," *The Knowledge Engineering Review*. 5:225-249. 1990.

[8] Copestake, A. et al. "Minimal recursion semantics: an introduction," *Research on Language and Computation.* 3:281-332. 2005.

[9] Crouch, R. "Packed rewriting for mapping semantics to KR," in *Proc. Sixth International Workshop on Computational Semantics*, 2005.

[10] Crouch, R. et al. "Semantics via f-structure rewriting," in *Proc. LFG06,* 2006.

[11] Frank, A. et al. "Question answering from structured knowledge sources," *Journal of Applied Logic*, 5, 20-48. 2007.

[12] Frank, A. et al. "Querying structured knowledge sources," in Proc. *AAAI Workshop on Question Answering in Restricted Domains,* 2005.

[13] Hallett, C. et al. "Composing questions through conceptual authoring," *Computational Linguistics,* 33(1):105-133, 2007

[14] Hallett, C. "Generic querying of relational databases using natural language generation techniques," in *Proc. 4th International Natural Language Generation Conference,* 2006.

[15] Kang, I. et al. "Lightweight natural language database interfaces," in *Proc. NLDB'04,* 2004.

[16] Kaplan, R. et al. "Integrating finite-state technology with deep LFG grammars," in *Proc. Workshop on Combining Shallow and Deep Processing for NLP (ESSLI)*, 2004.

[17] Lowry, M.R. et al. "AMPHION: automatic programming for scientific subroutine libraries," in *Proc. ISMIS'94*, 1994.

[18] Maxwell, J.T. et al. "An efficient parser for LFG," in *Proc. LFG'96,* 1996.

[19] O'Connor, M. et al. "The Chronus II temporal database mediator," in *Proc. AMIA02,* 2002.

[20] Pratt, I. et al. "The expressive power of the English prepositional system," in *Proc. Time-94'*

[21] Pinkal, M. "On semantic underspecification" in H. Bunt and R. Muskens (eds) *Computing Meaning.* Kluwer Academic Publishers. 1999.

[22] Popescu, A. et al. "Towards a theory of natural language interfaces to databases," in *Proc. IUI'03,* 2003.

[23] Richardson, K. et al. "Using a general-purpose NLP system for mapping English to RDF," poster, DARPA *Machine Reading Phase III Kickoff Meeting.* 2011.

[24] Shafer, R.W. "Rationale and uses of a public HIV drug-resistance database," *Journal of Infectious Disease,* 194 Suppl 1:S51-8. 2006.

[25] Stickel, M. et al. "A guide to SNARK," www.ai.sri.com/snark/tutorial.html, 2000.

[26] Stickel, M. et al. "Deductive composition of astronomical software from subroutine libraries," *Automated Deduction,* 12, 1994.

[27] Waldinger, R. et al. "Accessing structured health information through English queries and automatic deduction," in *Proc. of AAAI Spring Symposium on Health Communications,* 2011.

[28] Waldinger, R. et al. "Deductive question answering from multiple resources," in *New Directions in Question Answering.* Maybury, M (ed) MIT Press. 2004.