Nicky: Toward A Virtual Assistant for Test and Measurement Instrument Recommendations

Robert Kincaid
Keysight Laboratories
Keysight Technologies
Santa Clara, USA
robert.kincaid@keysight.com

Graham Pollock
Keysight Laboratories
Keysight Technologies
Roseville, USA
graham_pollock@keysight.com

***Abstract*—Natural language question answering has been an area of active computer science research for decades. Recent advances have led to a new generation of virtual assistants or chatbots, frequently based on semantic modeling of some broadly general domain knowledge. However, answering questions about detailed, highly technical, domain-specific capabilities and attributes remains a difficult and complex problem. In this paper we discuss a prototype conversational virtual assistant designed for choosing test and measurement equipment based on the detailed measurement requirements of the test engineer. Our system allows for multi-stage queries which retain sufficient short-term context to support query refinement as well as compound questions. In addition to the software architecture, we explore an approach to ontology development that leverages inference from reasoners and minimizes the complexity of entering the specifications for a large collection of instruments. Finally, we provide insights into the issues of building this system and provide recommendations for future designs.**

Question answering; intelligent assistant; chatbot; ontology; semantic query; reasoning

I. INTRODUCTION

Question answering systems have been an active area of research for many years beginning with early systems such as BASEBALL [1], LUNAR [2] and ELIZA [3]. In the past such efforts were developed largely in the field of artificial intelligence and have focused on expert systems in relatively limited domains. More recently, the advent of scalable cloud computing and improvements in natural language processing, voice recognition, and voice synthesis as well as ontology-based reasoning systems has led to ubiquitous access to such systems as Siri [4], Cortana [5], Alexa [6] and Google Now [7]. These newer technologies enable conversational access to so-called virtual assistants for answering simple day-to-day questions or performing various actions. While these systems are engaging, their usefulness to-date has been primarily limited to answering direct questions for information retrieval (the weather, stock quotes, etc.) or invoking specific commands (phone dialing, reminders, appointments, alarm setting, music playback, etc.).

In the electronics industry, when new electronic parts or devices are designed, they need to be tested to validate that the design functions properly and to the intended specifications. This is true for single integrated circuits, circuit boards as well as complex devices such as mobile phones, televisions and other consumer electronics. Typically, one or more test engineers will have to design a test plan for the device under test that includes specifying the measurements and ranges required. These measurement requirements dictate the specific test equipment necessary. Creating this test setup can be a somewhat complex task. A typical test scenario will require many different measurements to be made over varying combinations of conditions and will usually involve multiple instruments and devices. Test and measurement companies such as Keysight have product lines consisting of hundreds of different instruments (both current and legacy models still in use) with a variety of potentially overlapping capabilities. Each device has its own characteristic measurement functions, measurement ranges, accuracy, software compatibility and more.

To begin to address this instrument selection task, we explored constructing a conversational recommendation agent we call Nicky (after Nikola Tesla), focused on recommending test and measurement instruments based on a variety of required attributes and functions typically specified by the test engineer. This effort included investigating the feasibility of using an ontology to represent the relationship between test and measurement equipment and their capabilities. In particular, we hope to apply reasoning over this semantic network to aid in choosing appropriate instruments based on test criteria provided by the user.

Our primary goal in this work was to explore the design space and feasibility of building an intelligent virtual assistant for a highly technically-focused task related to test and measurement. To this end we have built a proof-of-concept system and in the process have observed some interesting issues. Thus, the main contributions of this paper are:

- A high-level description of system architecture
- An efficient approach to ontology development that leverages reasoners to do much of the work
- Observations about issues implementing a virtual assistant in a very technical domain
- Lessons learned and recommendations with respect to handling complex specifications and how this informs design; particularly with respect to speech vs. text input/output.

978-1-5090-4284-5/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSC.2017.11

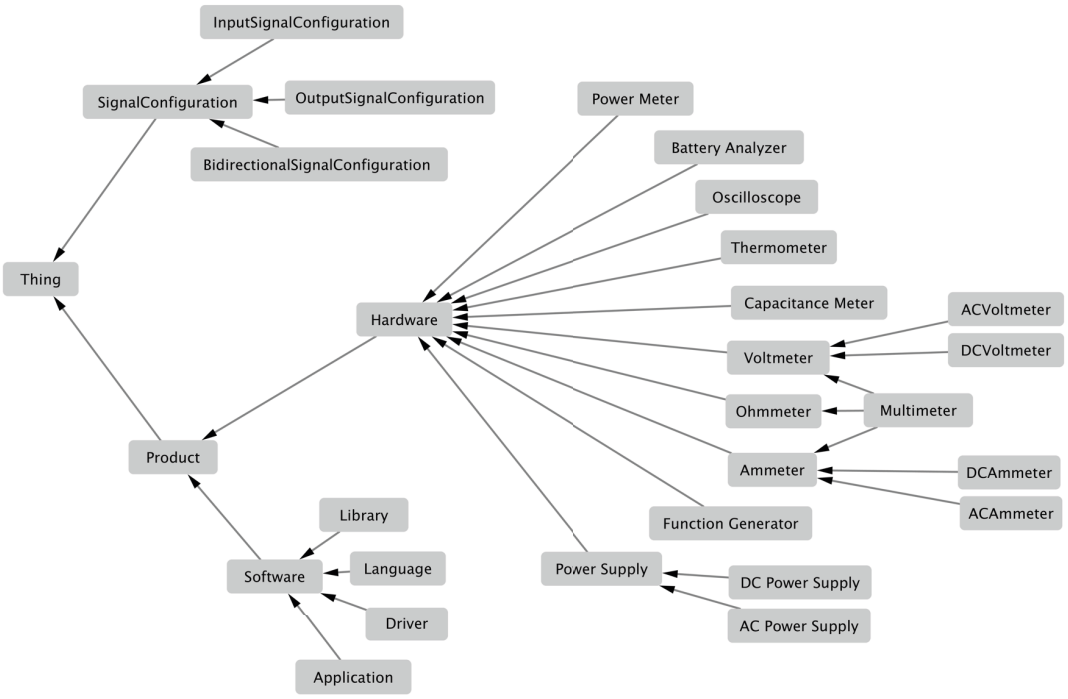

Figure 1. Basic class structure of the instrument ontology. Arrows indicate "is a" relationships.

II. Research Questions

In considering how to design and use a question answering system in our domain we focused on several research questions:

Q1: Can an ontology represent the complexity of test and measurement instruments and capabilities?

A central issue in our work was determining if it was possible to efficiently represent the inherent complexity of instrument characteristics using an ontology. This representation needed to allow the characteristics of instruments to be modeled in sufficient detail to allow easily extensible complex queries. We were also interested in the level of difficulty in authoring such an ontology and whether suitable SPARQL [8] queries could be successfully formed at this level of detail and complexity. While descriptive logic systems should in principle be capable of such tasks, a key concern was how practical authoring and using such a domain-specific ontology might be.

Q2: For our targeted use case, can reasoners provide useful benefit beyond a traditional relational database?

One of the main benefits of an ontology-based system is the ability to compute over the relations and attributes within the ontology. In particular, reasoners can be applied and potentially provide matching results that might not be easily obtainable in a traditional table-based model using specific, targeted queries that match only attributes.

Q3: Can a conversational interface efficiently support answering detailed, technical instrument queries and provide query refinement?

Implementing a dialog-based system for a highly technical use case presents a number of challenges, which we will discuss in more detail later in this paper. Therefore, another key question for our research was to what extent is a conversational approach useful, what strategies are necessary and what limits might there be on such approaches.

Q4: Can a domain-specific virtual assistant be built with modest resources and open source software?

While a number of well-known, robust virtual assistants are already deployed as scalable, high-performance cloud-based services, we were interested in exploring the feasibility of more modest systems with a narrow task focus and based on client-side hardware. Could we build a standalone application that replicated many of the features of these more extensive systems if the targeted task was sufficiently narrow?

III. Related Work

Question answering systems have a long history in user interface research and artificial intelligence, dating back to such very early systems as BASEBALL [1], LUNAR [2] and ELIZA [3]. Frequently these types of systems are built as a dialog system or a conversational agent [9]. Conversational agents can also be utilized for imperative commands used for device control such as voice activated phone dialing, music selection or controlling televisions. Brennan [10] even makes a case for considering conversation as a form of direct manipulation [11]. Interfacing to such agents has been accomplished via voice recognition and synthesis [4-7] as well as through purely text-based dialog systems [3, 12].

Advances in speech processing, natural language processing and artificial intelligence have led to the advent of modern virtual assistants which employ increasingly humanlike discourse. Siri [4] is a well-known pioneering example that was

largely based on the CALO project [13] from SRI and there are several other similar well-known systems [5-7]. However, these systems typically require extensive infrastructure to support a massive body of web-derived knowledge as well as a high volume of user queries. Additionally, query refinement through user context and discourse have begun to appear in such systems as Hound [14] and Viv [15].

Semantic networks, frequently in the form of ontologies, provide many affordances [16] that are often central to the reasoning systems of virtual assistants and conversational agents [17]. Such networks provide a rich infrastructure with which to express some knowledge domain and provide a computable representation for inference and query. Many of these systems are fact-based and provide means for querying a knowledge base for related facts or relationships. A rarer case is an ontology constructed around a highly technical and complex domain. A recent prominent example is IBM's work with the Watson platform to provide a knowledge base of oncology aimed at providing recommendations about cancer treatment options [18]. Wolfram Alpha [12] incorporates a large body of largely fact-based technical knowledge covering various mathematical and scientific domains as well as the ability to perform symbolic math. There has even been early work by Freiling [19] which endeavored to describe test and measurement instruments and their operation which is particularly relevant and motivational to the domain of our inquiry.

Finally, our targeted use case of device selection has similar intentions to previous work on virtual shopping assistants [20] and virtual sales clerks [21, 22]. These systems involve searching for relevant items for purchase based on attributes important to the user, such as style, price, etc. However, these attributes are generally well structured and easily searched by more traditional database technologies, although semantic approachs have begun to appear.

Despite all these recent advances, crafting an efficient, accurate and useful virtual assistant is still challenging [23-25], particularly for a complex, technical use case.

IV. Ontology Development

The instrument ontology was developed in OWL [26] using Protégé 5.0.0 for both editing as well as SPARQL [8] query prototyping. The Pellet reasoner [27] was used for inferring otherwise unspecified relationships and properties. We have implemented a collection of relevant instruments models as part of our "proof-of-concept". This included a selection of AC and DC power supplies, function generators, multimeters, power meters and oscilloscopes. We intended this collection to display a level of representative complexity as well as give reasonable coverage over a variety of typical instruments with different input and output channels, each with a variety of functions, properties and ranges.

To aid in the construction of the ontology we compiled a list of competency questions which were generally of the form:

What instruments measure voltage?

Which devices can produce current?

What can measure 0 to 10 volts DC?

What can produce 200 milliamps?

What instruments work with MATLAB?

Etc.

Broadly, the questions fell into several classes based on the following:

- What devices can measure some property (voltage, current, resistance, temperature, etc.), potentially over some range.
- What devices can produce such properties (usually a signal of some kind).
- Additionally, we included software as a category in the ontology so queries can also be constructed to select compatible software/hardware combinations.

One of the main objectives of our ontology design was to simplify the entry of instrument specifications into the ontology. To this end we created a relatively simple hierarchy of hardware and software classes (Figure 1) with axioms sufficient to infer subclass assignments directly from instrument properties. Note that for our use case, there is only one instance of an instrument model and so we chose to limit the class hierarchy to generic instrument types rather than include specific models as part of the class hierarchy directly.

Individual instrument models are primarily represented by a collection of signal configurations corresponding to the input and/or output modes the instrument supports. These signal configurations define the capabilities of the instrument via object and data property assertions but do not directly classify the instrument type. For a simple example, the U1273AX Handheld Digital Multimeter is defined as being composed of five individual input signal configurations as seen below in compact Turtle syntax:

```
:U1273AX rdf:type owl:NamedIndividual , owl:Thing ;
  rdfs:label "U1273AX Handheld Digital Multimeter,
              4.5 digit"^^xsd:string ;
  :modelNumber "U1273AX"^^xsd:string ;
  :hasInputSignalConfiguration
  :U1273AX_AC_Current ,
  :U1273AX_AC_Voltage ,
  :U1273AX_DC_Current ,
  :U1273AX_DC_Resistance ,
  :U1273AX_DC_Voltage .
```

All five of these signal configurations actually belong to a single physical channel. Channels, i.e. the physical connections capable of measuring or providing signals, can often be capable of more than one signal type. Further, an instrument can and often does contain multiple channels. Our example multimeter has a single DC Voltage mode on channel 1 defined by the following signal configuration definition:

```
:U1273AX_DC_Voltage rdf:type owl:NamedIndividual ,
                    owl:Thing ;
  :minDcVoltage "0.0"^^xsd:float ;
  :channelNumber "1"^^xsd:nonNegativeInteger ;
  :maxDcVoltage "1000.0"^^xsd:float ;
  :signalConfigurationName "DC Voltage"^^xsd:string ;
  :canMeasure :DcVoltage .
```

Each signal configuration is also unique to the instrument model. There is no hierarchy of signal channels or configurations

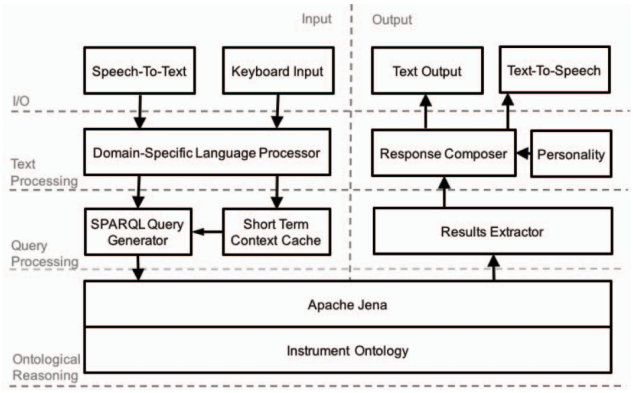

Figure 2. A high level overview of the Nicky architecture

since they are unique properties of the instrument model and not shared with other similar instruments. Further, it is useful to split this composition between instrument and channels so that we can enumerate and search by signal configuration properties.

Class axioms are used to infer subclass associations directly from these signal configurations. For example, an instrument that has a signal channel that produces voltage would map to a power supply, whereas a signal channel that measures voltage would map to a voltmeter as shown below:

```
:Voltmeter rdf:type owl:Class ;
  owl:equivalentClass
  [
    rdf:type owl:Restriction ;
    owl:onProperty :hasInputSignalConfiguration ;
    owl:someValuesFrom [
      rdf:type owl:Restriction ;
      owl:onProperty :canMeasure ;
      owl:hasValue :Voltage
    ]
  ] ;
  rdfs:subClassOf :Hardware ,
  [
    rdf:type owl:Restriction ;
    owl:onProperty :typeOfHardware ;
    owl:hasValue :Voltmeter
  ] .
```

In this way, the editor of the ontology does not have to manually determine with which instrument classes to associate a new entry, and our class hierarchy alone can now establish high-level relationships. All the multimeter instances will have several signal configurations that map through axioms to voltmeter, ammeter and ohmmeter and the following class definition will automatically assign such instruments to the class multimeter:

```
:Multimeter rdf:type owl:Class ;
   owl:equivalentClass [ rdf:type owl:Class ;
       owl:intersectionOf ( :Ammeter
                            :Ohmmeter
                            :Voltmeter )
   ] ;
   rdfs:subClassOf [
       rdf:type owl:Restriction ;
       owl:onProperty :typeOfHardware ;
       owl:hasValue :Multimeter
   ] .
```

Using this strategy, a modest amount of class definitions and axioms greatly simplifies the effort needed to specify and classify the numerous instruments we are interested in defining.

V. Design and Implementation

Nicky was developed as a stand-alone application written in Java (JDK 1.8) and has been verified to operate on Microsoft Windows, MacOS and Linux. Figure 2 depicts a block diagram of the main components of the architecture. In the following sections we describe these components in greater detail.

A. Query Input

Two modes of interaction are provided for conversing with the system:

Speech input is obtained via the computer's microphone input. The captured audio is sent to a third-party cloud-based speech-to-text service for conversion to text. A virtual button on the display (Figure 3) is used for a "push-to-talk" function (PTT). When the PTT button is released, the captured audio is remotely processed and returned as text in less than 1-2 seconds. While an always-listening mode might be more natural for the user, the use of a PTT function disambiguates "turn taking" often problematic in conversational agents. For our prototype, we are relatively immune to extended pauses in utterances that might otherwise be taken as a turn marker.

For comparison to speech input, we also implemented a keyboard-based interface in the same display. The user can, therefore, optionally type directly into the display to enter questions without using the PTT function. Such keyboard input eliminates the need for external calls to cloud-based speech processing and allows execution to take place entirely on the user's computer.

B. Domain-Specific Language Processor

Once a question is entered and the text acquired, it is submitted for further processing to the "Domain-Specific Language Processor". Since our use case has a reasonably limited vocabulary, we chose to not employ a full natural language processing system and instead relied on developing a compact domain-specific language processor with ANTLR 4 [28, 29].

We started with the basic competency questions used for ontology development and constructed ANTLR expressions to match most of the reasonable variations for formulating these questions. In this way we could handle syntax variations such as:

"What measures voltage?"

"Which instruments measure DC voltage?"

"What can measure 0 to 10 VDC?

etc.

In addition, we could process some normalizations at the lexical level. In particular, recognizing VDC or VAC as "volts DC" and "volts AC" respectively. Unit conversions were also handled within the ANTLR-constructed parser. For example, "mv" is interpreted as "millivolts" and properly converted to 0.001 volts when used in a query against the ontology. For our use case, one of the critical aspects of handling flexible text input

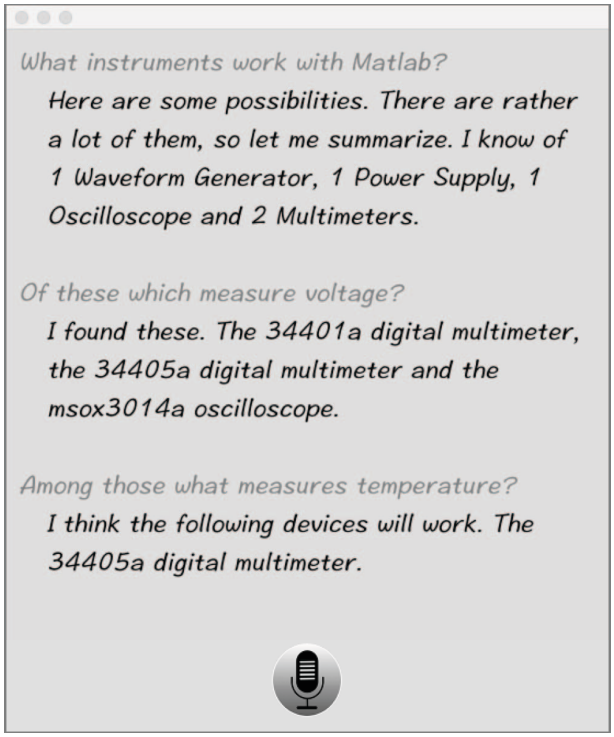

Figure 3. A typical session illustrating the layout of the prototype's user interface. Note the "push-to-talk" button at the bottom of the display.

is this ability to understand and convert various methods of specifying the same value in different units and/or abbreviations.

Once the user's sentence is fully parsed and normalized, entity extraction is performed to identify the key verb and object pairs such as "measure voltage" or "produce current" as well as any range restrictions such as "0 to 10 volts DC". This extracted, normalized query data is stored in an extensible context cache. If the parser detects specific continuation phrases such as "from those" or "which of these" etc. the extracted information is appended to the context cache. In the absence of a continuation phrase the cache is cleared before storing the current information and a new query sequence is begun.

C. SPARQL Query Generation and Execution

A SPARQL query is assembled and prepared for execution by enumerating the context cache containing one or more user supplied criteria. As our examples illustrate, signal requirements result in a hasSignalConfiguration statement, a canMeasure or canProduce statement and a corresponding filter restriction. Software requirements add a similar compatibleWithSoftware statement and matching filter. Additional instrument related attributes are added for retrieval. Our narrow use case permits all queries to take the same extensible form and can be assembled in a consistent, straightforward manner. We used Apache Jena [30, 31] as our run-time ontology processor.

D. Results Extractor

After a SPARQL query is executed, Jena returns a list of strings containing the query results. These results must be parsed to extract the relevant data, typically an instrument model number and type.

E. Response Composer

The extracted response elements are assembled into a reasonable English sentence as the answer to the user's question. When there is a large list of compatible instruments, summary statistics are computed to list the type and number of instruments found. If the list is small, a concatenated list of instruments is generated. To invoke a bit of personality into the system and avoid monotonous replies, a randomly selected prefix is prepended to the response such as "I know of" or "The" as well as a randomly selected preamble. See Figure 3 for example sentence constructions.

F. Response Output

If the PTT function is invoked and we are operating in voice mode, the text response is sent to a third-party cloud-based text-to-speech service for conversion to high quality audio output. This output is then streamed to the computers speakers for an audible response. The net effect is a conversational refinement of searching for an instrument.

In contrast, if the text interface is being used by simply typing, then the result is immediately output and no speech-to-text is applied.

VI. Examples

In this section we provide several illustrative examples of how the system functions.

Example 1: Query refinement through dialog

Figure 3 shows the display and a discourse to find an instrument that can measure voltage, temperature and is compatible with the software platform MATLAB. In this simple example, we first begin by looking for instruments with interfaces that are compatible with MATLAB. Since there are a number of them, rather than enumerate them all with voice synthesis, the system provides a brief summary. We then narrow the list first by asking a follow-up question for an instrument that measures voltage. We finally refine our query to arrive at a device that can also measure temperature. The corresponding final SPARQL query combining all questions is:

```
SELECT DISTINCT ?instrumentLabel ?typeLabel
WHERE {
   ?instrument :hasSignalConfiguration ?sig1 .
   ?instrument :hasSignalConfiguration ?sig2 .
   ?instrument rdfs:label ?instrumentLabel .
   ?instrument :compatibleWithSoftware ?software1 .
   ?software1 rdfs:label ?swtitle1 .
   ?sig1:canMeasure ?measure1.
   ?sig2:canMeasure ?measure2.
   ?instrument :productWebPage ?link .
   ?instrument :typeOfHardware ?type .
   ?type rdfs:label ?typeLabel .
   FILTER
   (
     ( regex(str(?swtitle1),"matlab","i") )
     &&
     ( regex(str(?measure1),"voltage","i") )
     &&
     ( regex(str(?measure2),"temperature","i") )
   )
} ORDER BY ?instrument
```

Note that the keywords “Of these” and “Among those” are the key phrases used to indicate to the query generator that these secondary queries should be combined with the previous query. In this way, while the SPARQL query retains the same basic form, it accumulates additional signals, software requirements and filter statements.

The basic structure of the query is to collect the relevant attributes such as signal configurations and software compatibility. These attributes are then used to filter the results based on query criteria. In this instance, it is necessary to obtain duplicate signals so we can use two different criteria (voltage and temperature). This construction allows compiling SPARQL queries to obtain any arbitrary combination of requirements retained in the context cache. Selecting items based on filters is potentially inefficient. However, since our instrument collection is never likely to evolve past a few thousand individual instruments, the flexibility afforded by this query construction for building extensible, complex queries is a useful trade-off.

Example 2: Compound queries

Nicky supports direct compound queries with Boolean conjunctions. An example equivalent to Figure 3 is:

What works with MATLAB and can measure voltage and can measure temperature?

This generates exactly the same SPARQL query shown in Example 1, and thus retrieves the same result, the 34405a digital multimeter.

Example 3: Query over ranges

When we introduce value ranges on instrument attributes, both the human and machine queries become slightly more complicated. An example is the question and answer:

What can generate 2 amps AC and 1 to 5 volts AC?
How about this? The ac6801a basic AC power source.

The corresponding SPARQL query is generated as:

```
SELECT DISTINCT ?instrumentLabel ?typeLabel
WHERE {
   ?instrument :hasSignalConfiguration ?sig1 .
   ?instrument :hasSignalConfiguration ?sig2 .
   ?instrument rdfs:label ?instrumentLabel .
   ?sig1:canProduce ?generate1.
   ?sig2:canProduce ?generate2.
   ?instrument :productWebPage ?link .
   ?instrument :typeOfHardware ?type .
   ?type rdfs:label ?typeLabel .
   ?sig1 :minAcCurrent ?minAcA1 .
   ?sig1 :maxAcCurrent ?maxAcA1 .
   ?sig2 :minAcVoltage ?minAcV2 .
   ?sig2 :maxAcVoltage ?maxAcV2 .
   FILTER
   (
   ( regex(str(?generate1),"ac.*current","i") &&
    ( ( ?minAcA1 <= 2.0 ) && ( ?maxAcA1 >= 2.0) ) )
   &&
   ( regex(str(?generate2),"ac.*voltage","i") &&
    ( ( ?minAcV2 <= 1.0 ) && ( ?maxAcV2 >= 5.0) ) )
   )
   }GROUP BY ?instrumentLabel ?typeLabel
    ORDER BY ?instrument
```

The basic structure of the query is similar to the earlier examples where we depend on filter selection. However, in this case we filter not only on object properties, but also on the ranges of relevant data properties.

Example 4: Queries that take advantage of inference

Each of the previous examples leverage inference. Our queries did not take the more traditional form of “what instruments are voltmeters”. Instead, we approached the questions from higher level abstractions such as “what can measure voltage” and “what can generate current”.

However, even direct queries about instrument classes rely on inference. The following question and answer is an example:

What instruments are multimeters?
Here are some possibilities. The 34401a digital multimeter, the 34405a digital multimeter and the u1273ax handheld digital multimeter.

Remember that these instruments were never explicitly defined as multimeters, but the class axioms defined for ammeter, ohmmeter and voltmeter as well as multimeter allows the ontology to assign them the class and function of multimeter purely from the signal channel definitions.

VII. Discussion

From our experience developing and testing the Nicky prototype, we can begin to answer the research questions posed in Section II. It is important to note that while we considered usability in our design, our interest to date is primarily focused on the more basic issue of feasibility. We were successful in producing a working and accurate system that returns true statements about instrument queries, and this is the context for most of our observations.

Q1: Can an ontology represent the complexity of test and measurement instruments and capabilities?

Given the subset of instruments we examined, we manually verified that query matches were accurate and complete. Given our ability to generate these accurate queries we conclude that it is possible to create an ontological structure that captures the complexity of measurement types and ranges necessary to sufficiently describe the instruments in question and satisfy our initial use case. This structure includes distinguishing between input and output channels, signal measurement vs. signal production, the ranges of electrical and radio frequency properties supported as well as what software is compatible with each instrument.

Despite leveraging the reasoner to make class assignments, we still found that creating and populating a test and measurement ontology was somewhat tedious and complicated in practice due to the numerous attributes and varying properties required to completely define an instrument. The basic concept of an ontology is foreign to most test and measurement experts and the difference relative to a traditional database is a difficult concept to grasp. For constructing a more complete ontology of instruments we feel custom authoring tools would be helpful. Such tools are critical since few sufficiently knowledgeable

domain experts are likely to have the desire or skills for direct ontology editing in a system like Protégé. Further, our technique of isolating the individual instance data from the class hierarchy removes the burden of both the tools and the editors from needing complete understanding of the ontology. However, this approach will only work with sufficient and accurate class and axiom definitions to accomplish the appropriate instance to class mappings. Such axiom mappings can be useful for cases where a moderate amount of work on the ontology contributes to a substantial simplification of later specification entry.

An added issue is that in most cases, entering the measurement ranges and characteristics of an instrument requires manually extracting detailed information from an instrument data sheet. These values are generally expressed in a table and these tables do not always conform to a consistent layout. Extracting data from such tables requires a fairly complex reading and comprehension task on the part of the ontology author to interpret the table and extract the appropriate information.

Q2: For our scenario, can reasoners provide useful benefit beyond a traditional relational database?

Employing reasoners was found to be useful in two different scenarios:

During ontology construction, we were able to describe instances of instrument classes by simply describing the capabilities of the instruments via signal configurations and allowed the reasoner to infer the class based on signal-related axioms. This inference relieves the ontology author from having to think deeply about what the correct class assignment should be. This cognitive task is particularly problematic for multi-function instruments that may be definable by several different classes (e.g. a multimeter that is simultaneously an ammeter, voltmeter and ohmmeter).

The reasoner can also discover non-traditional classes or functionality. Many measurement devices are multi-function and semantic reasoning has the potential to remind the test engineer of obscure or novel uses of such instruments. For example, a waveform generator which can produce sinusoidal waveforms can be used in some limited cases as an AC power supply. And we have already seen an example of a multimeter which can function as a thermometer.

Q3: Can a conversational interface efficiently support answering technical instrument queries and query refinement?

Due to the potential complexity of query refinement, care should be taken in how voice and even text input is used in this type of query task. We were driven, early in our prototyping process, to develop the ability to have ongoing multi-step refinement using a series of questions. Segmenting the dialog in this manner is necessary for two reasons:

First, speaking a complete specification question into the system might take multiple sentences and numerous parameter specifications. A lengthy monologue places an increased burden on the accuracy of the speech-to-text processing as well as the human trying to formulate a complex but interpretable question. User satisfaction would also be severely impacted if after dictating a several-minutes-long question, the system failed to parse the speech input correctly and the user had to repeat the entire, lengthy dictation. Thus, we submit what we consider a reasonable proposition: that it is better to ask simpler, shorter questions and maintain some form of multi-question query refinement process. In addition to a simpler conversational dialog, it may allow the questioner to observe an unexpected intermediate result that could lead to a different endpoint. This tactic is consistent with Brennen [10] which cited many relevant aspects of dialogue as an interaction technique. She specifically calls out building a complex query that takes more than one exchange.

Second, the results of some queries may include a fairly large number of potential instruments. This fact led us to implement the form of "summary" responses where a count of matches is provided with the assumption that the user would ask further refinement questions to reduce the response to a reasonable number of devices. Particularly for speech output, this is critical as a typical user would not be inclined to listen to a lengthy monologue of various model numbers and instrument types.

As a general observation we note that for simple questions and responses voice works well and is an engaging mode of interaction. As the dialog becomes more complex, text input and output seems more efficient. For *very* complex queries requiring multiple signal definitions and potentially requiring multiple instruments, a more traditional form-based query-by-example is probably required. Further user study is required to fully validate these findings, but we offer our initial experience and observations in the spirit that they may prove useful for future research.

Similarly, for our very focused and limited vocabulary, we found that a domain-specific language (DSL) parser was relatively easy and quick to implement. For many similarly focused cases, a DSL is probably sufficient and preferable to the overhead of a full NLP system. However, if our system were developed to handle increasingly complex compound queries over a broader range of domain questions, implementing a true NLP is likely necessary.

Q4: Can a domain-specific virtual assistant be built with modest resources and open source software?

We found that it was possible to construct an efficient system that executed primarily on the user's local machine. The only non-local processes were the speech-to-text and text-to-speech which may not be required or even desired for some cases such as noisy environments or where privacy might be of concern. This opens intriguing possibilities for similar semantic reasoning to be included in features of client-side applications typical of test and measurement software. Further, since many modern instruments are actually driven by embedded computer systems, such ontology-based capabilities could be practically integrated into the instrument itself, particularly if speech processing is not required. Ontology-driven test and measurement has been considered for some time [19] and our results appear to indicate that the computing power and software support has finally begun to catch up to this long established vision.

VIII. Conclusion and Future Work

We have presented Nicky, a proof-of-concept question-answering virtual assistant for selecting test and measurement

instruments based on user-supplied criteria. We have been able to explore a number of aspects of building a virtual assistant for a complex technical domain. This has provided useful insight into ontology construction, issues with user dialog management and user interface development. We also have a high degree of confidence that with sufficient effort, a workable ontology can be created to fully describe the capabilities and functional relationships of a large inventory of instrument types. While we have verified feasibility, formal user studies are needed to verify and refine the usability of such conversational systems for technical tasks such as instrument selection and at what point the complexity of the exercise requires a different mode of interaction.

Future work should include an expanded ontology that includes a wider array of instruments and devices to further explore issues of complexity and scalability. To enable this, we believe work toward domain-specific authoring tools is necessary to simplify data entry for domain experts. Such tools are particularly needed in an industrial context where ontology experts with sufficient domain expertise will typically be in short supply. This more complete ontology could leverage content and design principles from the Linked Open Data community [32] as well as potentially contribute to the growing body of publicly available ontologies.

While we have focused on a narrow use case of instrument selection, a semantic inventory of instruments offers a number of intriguing possibilities for future work ranging from on-line store search, instrument asset management to intelligent instrument control and automation. In addition, other technical domains with similarly structured relationships and selection criteria might also be amenable to a similar treatment. For example, selecting electronic components from a parts inventory could have similarities with respect to range criteria and complex multi-function and/or multi-application properties.

ACKNOWLEDGMENTS

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

REFERENCES

[1] Green Jr, B.F., Wolf, A.K., Chomsky, C., and Laughery, K., "Baseball: an automatic question-answerer". Proc. Western Joint IRE-AIEE-ACM Computer Conference, May 9-11 1961 pp. 219-224

[2] Woods, W.A., and Kaplan, R., "Lunar rocks in natural English: Explorations in natural language question answering", Linguistic structures processing, 1977, 5, pp. 521-569

[3] Weizenbaum, J., "ELIZA-a computer program for the study of natural language communication between man and machine", Commun. ACM, 1966, 9, (1), pp. 36-45

[4] Siri, http://www.apple.com/ios/siri/ (accessed August 24, 2016)

[5] Cortana, http://microsoft.com/cortana (accessed August 24, 2016)

[6] Alexa, https://developer.amazon.com/alexa (accessed August 24, 2016)

[7] Google Now, http://www.google.com/landing/now/ (accessed August 24, 2016)

[8] Pérez, J., Arenas, M., and Gutierrez, C., "Semantics and Complexity of SPARQL". Proc. International Semantic Web Conference (ISWC 2006) 2006 pp. 30-43

[9] Lester, J., Branting, K., and Mott, B., "Conversational agents", The Practical Handbook of Internet Computing, 2004, pp. 220-240

[10] Brennan, S.E., "Conversation as direct manipulation: An iconoclastic view", in "The Art of Human-Computer Interface Design" (Addison-Wesley, 1990), pp. 393-404

[11] Shneiderman, B., "The future of interactive systems and the emergence of direct manipulation", Behaviour & Information Technology, 1982, 1, (3), pp. 237-256

[12] WolframAlpha, http://www.wolframalpha.com/about.html (accessed August 24, 2016)

[13] Ambite, J.L., Chaudhri, V.K., Fikes, R., Jenkins, J., Mishra, S., Muslea, M., Uribe, T., and Yang, G., "Design and implementation of the CALO query manager". Proc. Proceedings of the National Conference on Artificial Intelligence 2006 pp. 1751

[14] Hound, http://www.soundhound.com/hound (accessed August 24, 2016)

[15] Viv, http://viv.ai/ (accessed August 24, 2016)

[16] Dou, D., Wang, H., and Liu, H., "Semantic data mining: A survey of ontology-based approaches". Proc. Semantic Computing (ICSC), 2015 IEEE International Conference on, 7-9 Feb. 2015 2015 pp. 244-251

[17] Purver, M., Niekrasz, J., and Peters, S., "Ontology-based multi-party meeting understanding". Proc. Proceedings of CHI 2005 Workshop: CHI Virtuality 2005

[18] Keim, B., "Dr. watson will see you... someday", IEEE Spectrum, 2015, 52, (6), pp. 76-77

[19] Freiling, M., "Designing an inference engine: from ontology to control". Proc. Artificial Intelligence for Industrial Applications, 1988. IEEE AI '88., Proceedings of the International Workshop on, 25-27 May 1988 1988 pp. 20-26

[20] Semeraro, G., Andersen, H.H.K., Andersen, V., Lops, P., and Abbattista, F., "Evaluation and Validation of a Conversational Agent Embodied in a Bookstore", in "Universal Access Theoretical Perspectives, Practice, and Experience: 7th ERCIM International Workshop on User Interfaces for All, Paris, France, October 24–25, 2002, Revised Papers" (Springer Berlin Heidelberg, 2003), pp. 360-371

[21] Mumme, C., Pinkwart, N., and Loll, F., "Design and implementation of a virtual salesclerk". Proc. 9th International Conference on Intelligent Virtual Agents (IVA 2009), September 14-16 2009 pp. 379-385

[22] Shimazu, H., "ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops", Artificial Intelligence Review, 2002, 18, (3), pp. 223-244

[23] Raux, A., Bohus, D., Langner, B., Black, A.W., and Eskenazi, M., "Doing research on a deployed spoken dialogue system: one year of let's go! experience". Proc. Ninth International Conference on Spoken Language Processing (INTERSPEECH 2006), Pittsburgh, PA, USA, September 17-21 2006 pp. 65-68

[24] Black, A.W., and Eskenazi, M., "The spoken dialogue challenge". Proc. Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, London, United Kingdom 2009 pp. 337-340

[25] Luger, E., and Sellen, A., "Like Having a Really Bad PA: The Gulf between User Expectation and Experience of Conversational Agents". Proc. Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, Santa Clara, California, USA2016 pp. 5286-5297

[26] Antoniou, G., and Van Harmelen, F., "Web ontology language: OWL", in "Handbook on ontologies" (Springer, 2009), pp. 91-110

[27] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., and Katz, Y., "Pellet: A practical OWL-DL reasoner", Web Semantics: Science, Services and Agents on the World Wide Web, 2007, 5, (2), pp. 51-53

[28] ANTLR, 4.5.1, http://www.antlr.org/

[29] Parr, T., "The definitive ANTLR 4 reference" (Pragmatic Bookshelf, 2013. 2013)

[30] Apache Jena, http://jena.apache.org/ (accessed August 24, 2016)

[31] McBride, B., "Jena: a semantic Web toolkit", IEEE Internet Computing, 2002, 6, (6), pp. 55-59

[32] Heath, T., and Bizer, C., "Linked data: Evolving the web into a global data space", Synthesis lectures on the semantic web: theory and technology, 2011, 1, (1), pp. 1-136