Available online at www.sciencedirect.com
SCIENCE DIRECT®
Web Semantics: Science, Services and Agents on the World Wide Web 1 (2004) 397–413
JOURNAL OF Web Semantics
www.elsevier.com/locate/websem

Explaining answers from the Semantic Web: the Inference Web approach

Deborah L. McGuinness, Paulo Pinheiro da Silva*

Knowledge Systems Laboratory, Stanford University, Stanford, CA 94305, USA

Received 11 March 2004; received in revised form 16 June 2004; accepted 25 June 2004

Abstract

The Semantic Web lacks support for explaining answers from web applications. When applications return answers, many users do not know what information sources were used, when they were updated, how reliable the source was, or what information was looked up versus derived. Many users also do not know how implicit answers were derived. The Inference Web (IW) aims to take opaque query answers and make the answers more transparent by providing infrastructure for presenting and managing explanations. The explanations include information concerning where answers came from (knowledge provenance) and how they were derived (or retrieved). In this article we describe an infrastructure for IW explanations. The infrastructure includes: IWBase — an extensible web-based registry containing details about information sources, reasoners, languages, and rewrite rules; PML — the Proof Markup Language specification and API used for encoding portable proofs; IW browser — a tool supporting navigation and presentations of proofs and their explanations; and a new explanation dialogue component. Source information in the IWBase is used to convey knowledge provenance. Representation and reasoning language axioms and rewrite rules in the IWBase are used to support proofs, proof combination, and Semantic Web agent interoperability. The Inference Web is in use by four Semantic Web agents, three of them using embedded reasoning engines fully registered in the IW. Inference Web also provides explanation infrastructure for a number of DARPA and ARDA projects.

Keywords: Web explanations; Inference Web; Knowledge provenance; PML

1. Introduction

Inference Web (IW) aims to enable applications to generate portable and distributed justifications for any answer they produce. IW addresses needs that arise with systems performing reasoning and retrieval tasks in heterogeneous environments such as the web. Users

* Corresponding author. Tel.: +1 650 723 1876; fax: +1 650 725 5850.
E-mail addresses: dlm@ksl.sanford.edu (D.L. McGuinness), pp@ksl.sanford.edu (P. Pinheiro da Silva).

1570-8268/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2004.06.002

(humans and computer agents) need to decide when to trust answers before they can use those answers with confidence. We believe that the key to trust is understanding. Explanations of knowledge provenance and derivation history can be used to provide that understanding [20]. In one simple case, users retrieve information from individual or multiple sources and they may need knowledge provenance (e.g., source identification, source recency, authoritativeness, etc.) before they decide to trust an answer. Users may also obtain information from systems that manipulate data and derive information that was implicit rather than explicit. Users may need to inspect information contained in the deductive proof trace that was used to derive implicit information before they trust the system answer. Many times proof traces are long and complex so users may need the proof transformed (or abstracted) into something more understandable that we call an explanation.

Some users will decide to trust the deductions if they know what reasoner was used to deduce answers and what data sources were used in the proof. Other users may need additional information including how an answer was deduced before they will decide to trust the answer. Users may also obtain information from hybrid and distributed systems and they may need help integrating answers and solutions. As web usage grows, a broader and more distributed array of information services becomes available for use and the needs for explanations that are portable, sharable, and reusable grows. Inference Web addresses the issues of knowledge provenance with its registry infrastructure called IWBase. It also addresses the issues concerned with inspecting proofs and explanations with its browser. It addresses the issues of explanations (proofs transformed by rewrite rules for understandability) with its language axioms and rewrite rules. IW addresses the needs for combination and sharing with its Proof Markup Language (PML) specification.

In this article, we include a list of explanation requirements gathered from past work, literature searches, and from surveying users. We present the Inference Web architecture and provide a description of the major IW components including the PML specification [27] and API, the IWBase registry [22,28] (containing information about inference engines, proof methods, ontologies, and languages and their axioms), the explanation dialogue component, the proof abstractor API, and the justification browser. We also provide some simple usage examples. We conclude with a discussion of our work in the context of explanation work and state our contributions with respect to trust and reuse. This article is an expanded and updated version of an earlier conference paper [21]. The primary updates include the integration with the Proof Markup Language, a description of the IWBase architecture, an alpha version of an explanation dialogue component, and a broadening of the work to add focus on explaining query plans, satisfiability results, and results from extraction engines.

2. Background and related work

Recognition of the importance of explanation components for reasoning systems has existed in a number of fields for many years. For example, from the early days in expert systems (e.g., MYCIN [29]), expert systems researchers identified the need for systems that understood their reasoning processes and could generate explanations in a language understandable to its users. Inference Web attempts to stand on the shoulders of past work in expert systems, such as MYCIN and the Explainable Expert System [32] on generating explanations.

IW also builds on the learnings of explanation in description logics (e.g., [1,2,16,18]) which attempt to provide a logical infrastructure for separating pieces of logical proofs and automatically generating follow-up questions based on the logical format. IW goes beyond this work in providing an infrastructure for explaining answers in a distributed, web-based environment possibly integrating many question answering agents using multiple reasoners. IW provides access to multiple justification paths that may lead to a single conclusion and those paths may integrate conclusions from different systems with distributed components. IW also attempts to integrate learnings from the theorem proving community on proof presentation (e.g., [4,8]) and explanation (e.g., [14]), moving from proof tracing presentation to abstractions and understandable explanations. IW attempts to learn from this and push the explanation component started in Huang's work and also add the emphasis on provenance and distributed environments.

The work in this article also builds on experience designing query components for frame-like systems

[3,16,10] to generate requirements. The foundational work in those areas typically focus on answers and only secondarily on information supporting the understanding of the answers. In our requirements gathering effort, we obtained requirements input from contractors in DARPA-sponsored programs concerning knowledge-based applications (the High Performance Knowledge Base program,[1] Rapid Knowledge Formation Program,[2] and the DARPA Agent Markup Language Program[3]) and more recently, the ARDA AQUAINT[4] and NIMD[5] programs and DARPA's IPTO Office programs. We also gathered requirements from work on the usability of knowledge representation systems (e.g., [19]) and ontology environments (e.g., [7,15]). We have also gathered needs from the World Wide Web Consortium efforts on CWM[6] and the related reasoner effort on Euler.[7] Finally, we gathered knowledge provenance requirements from the programs above and from previous work on data provenance from the database community (e.g., [5]) and more recently from work integrating information from extractors such as the work in Tap[8] [13] leading to our enhanced knowledge provenance infrastructure [28] and information integrators (e.g., ISI's Prometheus mediator[9] which uses information obtained from Fetch's[10] wrappers in appropriate domains). Additionally requirements have been more recently obtained from initial efforts to explain text analytics work (e.g., IBM's UIMA [9]) as well as initial efforts to explain semantic matches using satisfiability engines (e.g., [12]).

3. Requirements

If humans and agents need to make informed decisions about when and how to use answers from applications, there are many things to consider. Decisions will be based on the quality of the source information, the suitability and quality of the reasoning/retrieval engine, and the context of the situation. Particularly for use on the web, information needs to be available in a distributed environment and be interoperable across applications.

3.1. Support for knowledge provenance information

Even when search engines or databases simply retrieve asserted or "told" information, users (and agents) may need to understand where the source information came from with varying degrees of detail. Similarly, even if users are willing to trust the background reasoner in a question answering environment, they may need to understand where the background reasoner obtained its ground facts. Information about the origins of asserted facts, sometimes called provenance, may be viewed as meta information about told information. Knowledge provenance requirements may include:

- Source name (e.g., CIA World Fact Book). If facts are encountered in multiple sources, any integrated solution needs to have a way of identifying from which source information was taken.
- Date and author(s) of original information and any updates.
- Authoritativeness of the source (is this knowledge store considered or certified as reliable by a third party?).
- Degree of belief (is the author certain about the information?).
- Degree of completeness (within a particular scope, is the source considered complete. For example, does this source have information about all of the employees of a particular organization up until a some date? If so, not finding information about a particular employee would mean that this person is not employed, counting employees would be an accurate response to number of employees, etc.).

The information above could be handled with meta information about content sources and about individual assertions. Additional types of information may be required if users need to understand the meaning of terms or implications of query answers.

- Term or phrase meaning (in natural language or a formal language).

[1] http://reliant.teknowledge.com/HPKB/.
[2] http://reliant.teknowledge.com/RKF/.
[3] http://www.daml.org/.
[4] http://www.ic-arda.org/InfoExploit/aquaint/.
[5] http://www.ic-arda.org/NoveLIntelligence/.
[6] http://www.w3.org/2000/10/swap/doc/cwm.html.
[7] http://www.agfa.com/w3c/euler/.
[8] http://tap.stanford.edu/.
[9] http://www.isi.edu/info-agents/Prometheus/.
[10] http://www.fetch.com/.

- Term inter-relationships (ontological relations including subclass, superclass, part-of, etc.).

As a system addresses meta information, many additional issues come into play such as security, access, efficiency, and usage. We have separated these into a separate list since they may appear to be a secondary in that they arise as a result of meeting the needs of the initial knowledge provenance demands. There is overlap on many of these requirements with those placed on sophisticated database applications. Also, the topics above are addressed in IW by providing explicit support for Dublin Core-like properties and is evolving as user needs and usage patterns reveal other informational needs. The topics below, in some cases, have preliminary support levels in our implementation and we have plans to increase the support in future work.

- Unique identifiers for provenance information.
- Effective methods for indexing, storing, and querying provenance information.
- Persistence of provenance information.
- Support for privacy levels in storage and access.
- Support for views based on a number of criteria such as privacy level, topic, thread, etc.
- Support for reuse of provenance information — tool support may be required for retrieving and reusing meta-information across multiple queries, e.g., the reuse of inference rule meta information generated by multiple engines.
- Support for reasoning about provenance information.

3.2. Support for reasoning information

Once systems do more than simple retrieval, additional requirements result. If information is manipulated as a result of integration, synthesis, abstraction, deduction, etc., then users may need access to a trace of the manipulations performed along with information about the manipulations as well as information about the provenance. We refer to this as reasoning traces or proof traces. Note that we consider any system that manipulates information to be a reasoner. For example, in additional to standard theorem provers, we consider extractors that take text as input and output markup and/or logical form to be reasoners. Similarly, we consider systems that take a query as input and in addition to answers are able to generate a query plan as output. Requirements as a result of reasoning may include the following:

- The reasoner used.
- Reasoning method (e.g., tableaux, model elimination, extraction type, etc.).
- Inference rules supported by the reasoner.
- Reasoner soundness and completeness properties.
- Reasoner assumptions (e.g., closed world versus open world, unique names assumption, etc.).
- Reasoner authors, version, etc.

The previous points all address meta information concerning the reasoner. The next set of requirements arise from using a reasoner and working with it in the context of an answer. These include:

- Detailed trace of inference rules applied (with appropriate variable bindings) to provide conclusion.
- Term coherence (is a particular definition incoherent?).
- Were assumptions used in a derivation? If so, have the assumptions changed?
- Source consistency (is there support in a system for both A and).
- Support for alternative reasoning paths to a single conclusion.
- Support for accessing alternative reasoning paths to the same conclusion.
- Support for accessing the implicit information that can be made explicit from any particular reasoning path.

3.3. Support for explanation generation

While knowledge provenance and proof traces may be enough for expert logicians when they attempt to understand why an answer was returned, usually they are inadequate for a typical user. For our purposes, one of our views of an explanation is as a transformation of a proof trace into an understandable justification for an answer. With this view in mind, we consider techniques for taking proofs and proof fragments and rewriting them into abstractions that produce the foundation for what is presented to users. In order to handle rewriting, details of the representation and reasoning language must be captured along with their intended semantics. Additionally, users may need to know both *what* manipulations were done (i.e., what rules of inference were

used) as well as *how* manipulations were done (i.e., what was the plan used to obtain information, were resource limitations in place, etc.) Support for both kinds of proof traces and their abstractions into explanations are needed in many applications. Requirements for explanations may include:

- Representation language identification.
- Representation language descriptions (e.g., DAML + OIL, OWL, RDF, etc.).
- Axioms capturing the semantics of the representation languages.
- Description of rewriting rules based on language axioms.

Much of the past work on explanation, whether from expert systems, theorem proving, or description logics, has focused on single systems or integrated systems that either use a single reasoner or use one integrated reasoning system. Systems being deployed on the web are moving to distributed environments where source information is quite varied and sometimes question answering systems include hybrid reasoning techniques. Additionally multi-agent systems may provide inference by many applications. Thus many additional requirements for proofs and their explanations may arise from a distributed architecture. Some requirements we are addressing are listed below:

- Reasoner result combinations (if a statement is proved by one system and another system uses that statement as a part of another proof, then the second system needs to have access to the proof trace from the first system).
- Portable proof interlingua (if two or more systems need to share proof fragments, they need a language to use as an interlingua for sharing proofs).
- Support for registering translators and comparators that can be used to translate statements from one language to another and can be used to identify similarities and differences between statements.
- Support for handling conflicting information.

3.4. Support for proof presentation

If humans are expected to view proofs and their explanations, presentation support needs to be provided. Human users will need some help in asking questions, obtaining manageable size answers, asking follow-up question, etc. Additionally, even agents need some control over proof requests. If agents request very large proofs, they may need assistance in breaking them into appropriate size portions and also in asking appropriate follow-up questions.

Requirements for proof presentation may include:

- Method(s) for asking for explanations (or proofs).
- Method(s) for breaking up proofs into manageable pieces.
- Method(s) for pruning proofs and explanations to help the user find relevant information.
- Method(s) for proof and explanation navigation (including the ability to ask follow-up questions).
- Presentation solution(s) compatible with web browsers.
- Method(s) for obtaining alternative justifications for answers.
- Different presentation formats (e.g., natural language, graphs, etc.) and associated translation techniques.
- Method(s) for obtaining justifications for conflicting answers.

4. Use cases

Every query-answering environment is a potential new context for the Inference Web. We provide two motivating scenarios and use the second scenario for our examples throughout the article. Consider the situation where someone has analyzed a situation previously and wants to retrieve this analysis. In order to present the findings, the analyst may need to defend the conclusions by exposing the reasoning path used along with the source of the information. In order for the analyst to reuse the previous work, s/he will also need to decide if the source information and assumptions used previously are still valid (and possibly if the reasoning path is still valid).

Another simple motivating example arises when a user asks for information from a web application and then needs to decide whether to act on the information. For example, a user might use a search engine interface or a query language such as OWL-QL[11] for retrieving information such as “zinfandels from Napa Valley” or

[11] http://ksl.stanford.edu/projects/owl-ql/.

"wine recommended for serving with a spicy red meat meal" (as exemplified in the wine agent example in the OWL guide document [31]). A user might ask for an explanation of why the particular wines were recommended as well as why any particular property of the wine was recommended (like flavor, body, color, etc.). The user may also want information concerning whose recommendations these were (a wine store trying to move its inventory, a wine writer, etc.). In order for this scenario to be operationalized, we need to have the following:

- A way for applications (reasoners, retrieval engines, etc.) to dump justifications for their answers in a format that others can understand. This supports the distributed proofs requirements above. To solve this problem we introduce a portable and sharable proof specification called the Proof Markup Language.
- A place for receiving, storing, manipulating, annotating, comparing, and returning meta information used to enrich proofs and proof fragments. To address this requirement, we introduce the IWBase for storing the meta information and the Inference Web registrar web application for handling IWBase data. This provides the infrastructure to support the registration of provenance-related meta information.
- A way to present justifications to the user. Our solution to this has multiple components. First the IW browser is capable of navigating through proof dumps provided in PML format. It can display multiple formats including KIF[12] and a limited form of English. Additionally, it is capable of using rewrite rules (or tactics) to abstract proofs in order to provide more understandable explanations. Finally, we have an alpha version of an explanation dialogue component. This interface attempts to provide a useful summary explanation initially and then suggests appropriate follow-up questions chosen by context. The interface also supports an option for user-provided input that can be used to teach the system about user preferences and requested updates. Using this combination, we address issues related to reasoning, explanations, and presentation.

We will use the Wine Agent example to introduce the components of the Inference Web in the future sections. One typical task of the Wine Agent is to take a particular meal or a description of a meal as input and then to suggest a particular wine to serve with the meal or a description of the wine to serve with the meal. Typical questions that a user (or agent) might ask include: What color wine should be served? (white, red, etc.) What variety of wine should be served? (zinfandel, chardonnay, etc.) What type of food is being served? (a seafood dish, meat, etc.) Why is the system suggesting a particular wine or property?

Note that although these questions may seem superficial, the reasoning used to determine the suggestions or explanations is analogous to the reasoning used in configuration or matching tasks. In fact, the original wines demo was built to represent the reasoning that was being done in a complicated configurator implementation for telecommunications equipment but was cast in a more approachable domain for demonstrations [25].

5. Inference Web

The Inference Web framework contains the following:

- data used for representing proofs, explanations, and meta information about proofs and explanations;
- software tools and services used for building, maintaining, presenting, and manipulating proofs.

In terms of data, the Inference Web provides the Proof Markup Language — an OWL-based specification for documents representing both proofs and proof meta information. PML classes are OWL [24] classes (thus they are subclasses of `owl:Class`) and they are either *proof elements* (proof level concepts) or *provenance elements* (provenance level concepts).

Inference Web proofs and explanations are represented within PML documents built using proof elements and referring to provenance elements, as described in Section 5.1. PML documents become a portion of the Inference Web data used for combining and presenting proofs and for generating explanations. Fig. 1 presents an abstract and partial view of the Inference Web framework[13] showing proofs and explanations in the web. Inference Web data also

[12] http://logic.stanford.edu/kif/kif.html.

[13] A more detailed view is available at http://iw.stanford.edu/arch_details.html.

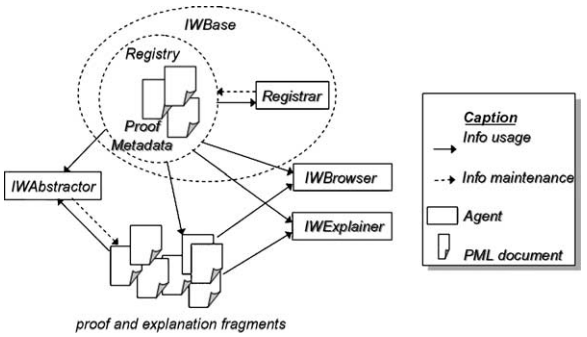

Fig. 1. Inference Web framework overview.

includes a distributed repository of PML documents representing proof-related meta information. The PML descriptions include provenance information about proof elements such as sources, inference engines and inference rules, as described in Section 5.2. IWBase is an infrastructure within the Inference Web framework for proof meta information, as described in Section 5.3.

In terms of software, Inference Web tools include: the registrar for handling IWBase entries, as described in Section 5.3; the proof abstractor API for transforming potentially long and incomprehensible PML proofs into shorter and more understandable PML explanations, as described in Section 5.4; the browser for displaying proofs, as described in Section 5.5; the explanation dialogue component for providing an explanation dialogue with users, as described in Section 5.6; and planned future tools such as proof web-search engines, proof verifiers, proof combinators, and truth maintenance systems. Fig. 1 presents how IW data is used by some of the IW tools mentioned above. For instance, it shows that the explainer has PML proofs as inputs and outputs. In this article, we limit our discussion to the PML specification (and an associated API), IWBase architecture (and the associated registrar tools and proof generation services), explanations, and the browser.

5.1. PML proof elements

Our PML specification includes two major components for building proof trees: `ModeSets` and `InferenceSteps`. Fig. 2 presents a typical dump of an IW node set. It may have been dumped after a user asked the wine agent for a wine recommendation and then the user was interested in determining the color

```
<rdf:RDF>
 <iw:NodeSet rdf:about="http://.../IW1_0.owl#IW1_0">
  <iw:hasConclusion>
      (wines:COLOR WINE9 ?x)
  </iw:hasConclusion>
  <iw:hasLanguage
        rdf:resource="http://.../registry/LG/KIF.owl#KIF"/>
  <iw:isConsequentOf rdf:parseType="Collection">
   <iw:InferenceStep>
    <iw:hasRule
         rdf:resource="http://.../registry/DPR/GMP.owl#GMP"/>
    <iw:hasInferenceEngine
       rdf:resource="http://.../registry/IE/JTP.owl#JTP"
       rdf:type="http://.../iw.owl#InferenceEngine"/>
    <iw:hasAntecedent rdf:parseType="Collection">
     <iw:NodeSet rdf:about="http://.../IW1_1.owl#IW1_1"/>
     <iw:NodeSet rdf:about="http://.../IW1_5.owl#IW1_5"/>
    </iw:hasAntecedent>
    <iw:hasVariableMapping rdf:parseType="Collection">
     <iw:VariableMapping iw:Variable="?x">
      <iw:Term>
           |http://.../wines.owl#|::|White|
      </iw:Term>
     </iw:VariableMapping>
    </iw:hasVariableMapping>
   </iw:InferenceStep>
  </iw:isConsequentOf>
 </iw:NodeSet>
</rdf:RDF>
```

Fig. 2. A PML node set.

of the recommended wine. A `NodeSet` represents a step in a proof whose conclusion is justified by any of a set of inference steps associated with the `NodeSet`. PML adopts the term “node set” since each instance of `NodeSet` can be viewed as a set of nodes gathered from one or more proof trees having the same conclusion. The `iw:hasConclusion` property of a node set represents the expression concluded by the proof step. Every node set has one conclusion, and a conclusion of a node set is represented in the language specified by the `iw:hasLanguage` property of the node set. In the example, the node set has a conclusion stating that the color of `WINE9` is ?x or the value of the color property of `WINE9` is the item of interest. The node set represents a statement and the last step in a deductive path that led a system to derive the statement.

In general, each node set can be associated with multiple or single inference steps as presented by the `iw:isConsequentOf` property of the node set in Fig. 2. A proof can then be defined as a tree of inference steps explaining the process of deducing the consequent sentence (a more formal definition of proofs within PML documents is described in [27]). In terms of number of files, a proof can physically vary from a single PML file containing all its node sets to many

PML files, each one containing a single node set. Also, PML files containing node sets can be distributed in the web. Considering the IW requirement that proofs need to be combinable, it is important to emphasize that a set of PML node sets inter-connected by their inference steps is a forest of proof trees since each node set can have multiple inference steps, each inference step representing an alternative justification for the node set conclusion.

An `InferenceStep` represents a justification for the conclusion of a node set. Inference steps are anonymous OWL classes defined within node sets. For this reason, it is assumed that applications handling PML proofs are able to identify the node set of an inference step. Also for this reason, inference steps have no URIs. For an IW proof, an `InferenceStep` is a single application of an inference rule, whether the rule is primitive or derived as discussed in Section 5.3. Inference rules (such as modus ponens) can be used to deduce a conclusion from any number of antecedents (that are the conclusions of other node sets). Inference steps contain URI references to node sets concluding its antecedents, the inference rule used, the supporting sources for the justification, and any variable bindings used in the step. There is no source associated with the node set in Fig. 2 since it is derived (although it could be derived and associated with a source). If it had been asserted, it would require an association to a source, which is typically an ontology that contains it. The antecedent sentence in an inference step may come from conclusions in other node sets, existing ontologies, extraction from documents, or they may be assumptions.

With respect to a query, logical starting points for a set of PML node sets are the node sets concluding the answer sentences for the query. Any node set can be presented as a stand alone, meaningful proof fragment as it contains at least one inference step, and each one of its inference steps has the inference rule used along with links to the inference step antecedents, sources and variable bindings.

The IW infrastructure can automatically generate follow-up questions for any proof fragment by asking how each antecedent sentence was derived. The individual proof fragments may be combined together to generate a complete proof, i.e., a set of inference steps culminating in inference steps containing only asserted (rather than derived) antecedents. When an antecedent sentence is asserted, there are no additional follow-up questions required and that ends the complete proof generation. The specification of IW concepts used in Fig. 2 is available at http://iw.stanford.edu/2004/03/iw.owl.

5.2. PML provenance elements

Provenance elements are used to provide information about the components used in a proof. Every IWBase entry is an instance of an PML provenance element. *InferenceEngine, Language*, and *Source* are the core provenance elements. Other PML provenance elements are related to one of these core elements.

The *InferenceEngine* is a core concept since every inference step should have a link to at least one entry of *InferenceEngine* that was responsible for instantiating the inference step itself. For instance, Fig. 2 shows that the `iw:hasInferenceEngine` property of `iw:InferenceStep` has a pointer to `JTP.owl`, which is the IWBase meta information about Stanford's JTP[14] model-elimination theorem prover. Inference engines currently may have the following properties associated with them: name, URL, author (s), date, version number, organization, etc. The property list may expand as usage demands dictate.

InferenceRule is one of the more important concepts associated with *InferenceEngine.* Inference rules basically tell a user or agent what kind of manipulations a particular inference engine may perform. With respect to an inference engine, registered rules can be either primitive or derived from other registered rules. Fig. 3 contains a screen shot from an IW browser interface presenting the entry for the modus ponens (MP) rule. Thus, MP may be a primitive rule for some inference engines.[15] Each of the inference rules may include a name, description, optional example, and optional formal specification. Fig. 3 shows that the MP inference rule can be formally specified by the string "`%p, (implies %p %q)| – %q; (Sent %p %q)`" that is written in Proof Protocol for Deductive Reasoning (PPDR) [26], which is built on top of the evolving SCL.[16] Thus, the meaning of the MP rule comes from the PPDR semantics and the MP specification in

[14] http://www.ksl.stanford.edu/software/jtp/.

[15] MP or any rule may be primitive for one reasoner while it may be derived for another reasoner.

[16] http://cl.tamu.edu/docs/scl/scl-latest.html.

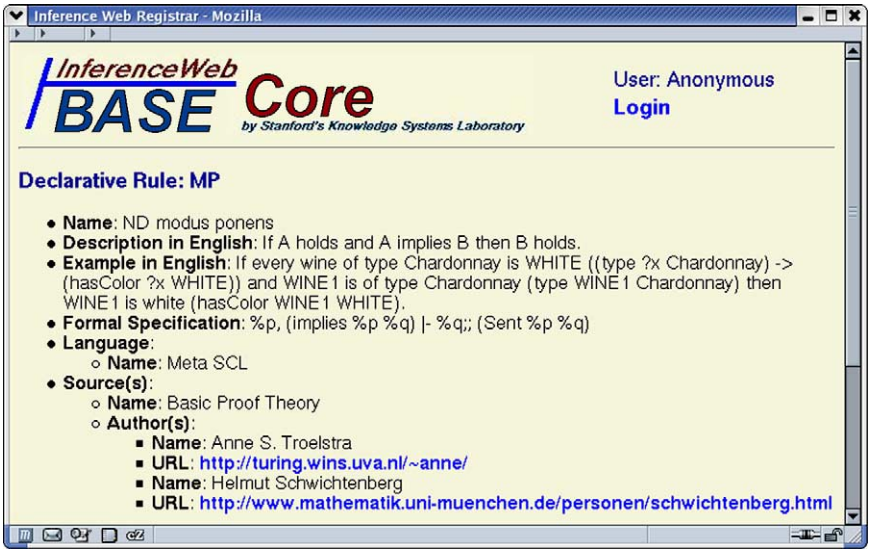

Fig. 3. Sample IWBase entry for an inference rule.

PPDR. Given a rule: it has "%p" and "(`implies %p %q`)" as premises; "%q" as conclusion; and "(`Sent %p %q`)" as side-condition. Moreover, premises and conclusion are sentence patterns since they use meta-variables (e.g., `%p and %q`). The (`Sent %p %q`) side condition says that the arguments `%p` and `%q` used in the premises and conclusion can be bound to a sentence.

Inference Web does not have a specific standard language for formalizing inference rule specifications. Instead, through PML, Inference Web provides a mechanism for registering rule specifications and the languages used to state the rule specifications. For instance, in the MP example above the rule specification is written in PPDR, which is an appropriate language for describing rules used in proofs where conclusions are written in KIF. Any valid instantiation of the premises and conclusion of the MP rule specification above are valid KIF sentences. PPDR is a convenient choice for rule specification since it was designed for this purpose, is built on the next generation of KIF, and is understood by Inference Web, thereby enabling Inference Web to provide proof abstraction services. It is not the only choice however that can be registered—rules may be specified in other languages as well. The downside to registration in languages other than SCL is that some IW tools such as proof checkers, however, may be unable to provide as many services for proofs applying rules formally specified in languages that the tools cannot understand.

Our experience specifying primitive rules in the Inference Web has demonstrated that a significant proportion of them can be formalized completely by a declarative specification language for rules such as PPDR. PML refers to these rules that can be formalized completely `DeclarativeRules`. Rules that cannot be fully specified formally are called `MethodRules` since rules of this category may need to rely on an additional method for deciding whether an inference step based on a method rule is a valid application of the rule. Method rules are specified in PML in order to accommodate meta-information about rules often called *procedural attachments*. For method rules, in addition to the formal specification, PML allows the registration of a method and a language used to write the method. In fact, the formal specification string of a method rule can still be used for matching premises with conclusions while the rule method can be applied later at proof-checking time to verify the correct use of the rule in a proof.

Many reasoners also use a set of derived rules that may be useful for optimization or other efficiency concerns. One individual reasoner may not be able to provide a proof of any particular derived rule but it

may point to another reasoner's proof of a rule. Thus, reasoner-specific rules can be explained in the IWBase before the reasoner is actually used to generate PML proofs. Inference Web thus provides a way to use one reasoner to explain another reasoner's inference rules (This was the strategy used in [2,1] for example where the performance tableaux reasoner was explained by a set of natural-deduction style inference rules in the explanation system). This strategy may be useful for explaining heavily optimized inference engines. It may also be useful for situations where it is known that one reasoning method (such as tableaux) is better for one type of explanation (such as counter example-based explanations of negative results), while another method is better for another type of explanation. IWBase already contains inference rule sets for many common reasoning systems. Users may view inference rule sets to help them decide whether to use a particular inference engine. Today IWBase contains rule sets for JTP, JTP's special purpose reasoners for DAML/OWL and temporal reasoning, SNARK, JSAT, ISI's Mediator, and ten of IBM's UIMA extractor engines. It also has partial rule sets for some other reasoners.

Inference engines may use specialized language axioms to support a language such as OWL or RDF. *Language* is a core IWBase concept. Axiom sets such as the one specified in [11] may be associated with a *Language*. The axiom set may be used as a source and specialized rewrites of those axioms may be used by a particular theorem prover to reason efficiently. Thus proofs may depend upon these language-specific axioms sets called *LanguageAxiomSets* in the IW. It is worth noting that an entry of *Language* may be associated with a number of entries of *LanguageAxiomSet* as different reasoners may find different sets of axioms to be more useful. For example, JTP uses a horn-style set of DAML axioms for its DAML reasoner while another reasoner may use a slightly different set for efficiency, stylistic, interoperability, or presentation reasons. Also, an entry of an *Axiom* can be included in multiple entries of *LanguageAxiomSet*. The content attribute of *Axiom* entries contains the axiom stated in the language specified by the language attribute of *Axiom*.

Source is the other core IWBase concept and it is a provenance element since it is used to identify the origin of a piece of information. *Source* is specialized into five basic classes: *Person*, *Team*, *Publication*, *Ontology*,[17] *Organization* and *Website*. At the moment, we are expanding the specification of (authoritative) sources as required. We have begun with a minimal description of these sources in the initial specification used in the IW and are expanding as needed based on empirical usage studies. Entries of *Ontology*, for example, describe stores of assertions that may be used in proofs. It can be important to be able to present information such as ontology source, date, version, URL (for browsing), etc. IW uses ontology in a broad sense [17] and includes both conceptual models as well as individual information and thus both knowledge bases and domain models are registered as ontologies in IWBase. Fig. 4 contains a sample ontology registry entry for the ontology used in our wine examples.

5.3. IWBase

IWBase is an inter-connected network of distributed repositories of proof and explanation meta information. Each repository of the network is an *IWBase node* residing in a web server. An IWBase node entry is a URI on the residing web server containing an OWL document of a provenance element. The content of each IWBase node URI is also mirrored in a database system. Therefore, PML proofs and explanations can have direct access to their meta information by resolving their URI references to IWBase entries.

IWBase node services, however, may need more sophisticated ways of querying the entries since they may not know exactly which entry to retrieve, for example, when an IWBase user needs to browse the entries in a node. In these cases, the services can take advantage of the underlying database system for querying node entries. For instance, in order to interact with IWBase, each node provides a collection of services collectively called a node *registrar* that supports users in updating or browsing the registry. The registrar may grant update or access privileges on a provenance element basis and the node administrator may define and implement policies for accessing the IWBase node. The generation of proof fragments is a straightforward task once inference engine data structures storing proof elements are identified as IW components. To facilitate the generation of proofs, IWBase provides

[17] LanguageAxiomSet is a subclass of Ontology.

Fig. 4. Sample IWBase entry for an ontology.

a set of SOAP-based web services that dump proofs from IW components and uploads IW components from proofs. This service is a language-independent facility used to dump proofs. Also, it is a valuable mechanism for recording the usage of IWBase entries.

In addition to the generic properties of IWBase nodes described above, the IWBase architecture specifies that each node is either a *core* node or a *domain* node. In fact, some provenance elements such as *inference engine*, *inference rule* and *language* are so generic that it may be appropriate to gather them in a single node, the core node, that is also publicly available for the other IWBase nodes. The current demonstration registrar for the core node is available at: http://inferenceweb.stanford.edu/iwregistrar/ and is one example core node. The core node architecture is convenient when there is one set of entries that describe the main meta information about the common engines, their rules, and representation and reasoning languages. Empirically, we have found our current uses of Inference Web benefit from such a core node. However, domain ontologies and their related meta information can vary widely from project to project thus these are appropriate to maintain in project-specific domain nodes. Just as the notion of upper ontologies is both popular and contentious, we anticipate some upper level ontologies to emerge that are popular enough and reused enough that Inference Web users will find that it may be beneficial to have these included in the core node registries. We expect these decisions to evolve with usage.

The IWBase architecture also specifies some services supporting the collaboration between the nodes. Basic services for making local copies of node entries are provided by a concurrent version system (CVS) repository where the OWL URIs are stored. Using the CVS services, users can check out personal copies of other node entries (whether they are entries from the core or a domain node) and store them locally. In our usage to date, we have found that domain node administrators may prefer to keep local copies of the core node for efficiency and/or privacy issues. More sophisticated services are provided for interacting with domain nodes. For instance, the administrator of a domain node (e.g., a node specialized with meta information about laptops) can specify that the node has visibility of another domain node. So, if a domain node for laptop computers may benefit from reusing some meta information already stored in a domain node about computers in general, it may use that node. These services between domain nodes also provide a solution for the problem of deciding where to store meta information about the so-called *upper level ontologies*. In fact, it is up to the users of another domain node to

decide whether they want to reuse meta information about other ontologies and thus they may decide what they would like to include.

The current IWBase provides support for provenance information at the level of knowledge bases and ontologies. However, we are in the process of extending the IWBase infrastructure in order to provide support for provenance information whenever it is possible to identify some document or document element to which we can associate provenance information as described in [28].

5.4. Proof abstractor API

Although essential for automated reasoning, inference rules such as those used by theorem provers and registered in the IWBase as *InferenceRule* entries are often inappropriate for "explaining" reasoning tasks. Moreover, syntactic manipulations of proofs based on atomic inference rules may also be insufficient for abstracting machine-generated proofs into some more understandable proofs [14]. Proofs, however, can be abstracted when they are rewritten using rules derived from axioms and other rules. Axioms in rewriting rules are the elements responsible for recognizing patterns and providing rewritten abstracted versions of the rules. Entries of *DerivedRule* are the natural candidates for storing specialized sets of rewriting rules. In IW, tactics are rewrite rules associated with axioms, and are used independent of whether a rule is atomic or derived.

The proof abstractor algorithm generates explanations in a systematic way using IWBase derived rules. Many intermediate results are "dropped" along with their supporting axioms, thereby abstracting the structure of proofs, when applying the algorithm. The general result is to hide the core reasoner rules and expose abstractions of the higher-level derived rules. An example of an IW explanation is described in the Inference Web web page at: http://iw.stanford.edu/documents-abstractions.html. The implementation of the proof abstractor API is work in progress. We have used the current rewrite rule set to abstract presentations of answers obtained from JTP in analysis applications.

5.5. Browser

Inference Web includes a browser that can display both proofs and their explanations in a number of proof styles and sentence formats. Initially, we include the "English", "Proof" and "Dag" styles and the restricted "English", "KIF" and "Raw" preferred sentence formats.[18] We also expect that some applications may implement their own displays using the IW API and one of our projects uses this model. The IW browser implements a lens metaphor responsible for rendering a fixed number of levels of inference steps depending on the lens magnitude setting. The prototype browser allows a user to see up to five levels of inference steps simultaneously along with their conclusions and antecedent sentences.

Fig. 5 shows a screen shot of the browser presenting two levels of inference step for one proof of the wine use case in Section 4. Prior to this view, the program has asked what wine to serve with a seafood course. Fig. 5 shows a proof fragment concluding that `New-course`, which is the selected meal course, requires a drink that has a white color since it is a seafood course. The sentences are formatted in English and the lens magnitude is two, thus the browser displays the inference steps used to derive the proof fragment conclusion including its antecedents and the antecedent's derivations. Concerning preferred sentence formats, the browser supports some restricted translations between sentences that can be requested by the user. For example, the "Raw" format indicates that the user wants to see node set conclusions as originally stated. However, if the user selects "KIF", then if node set conclusions are not already in KIF, and the browser has a translator from the original language into KIF, then the browser translates and presents the sentences in KIF. Otherwise, it presents the sentence in its original language. The same method is used for translating other formats into English, for example. We currently have a KIF to limited English translator for such needs.

We believe that one of the keys to presentation of justifications is breaking proofs into separable pieces. Since we present fragments, automatic follow-up question support is a critical function of the IW browser. Every element in the viewing lens can trigger a browser action. The selection of an antecedent re-focuses the lens on an antecedent's inference step. For other lens elements, associated actions present

[18] Current investigations are underway for N3 as an additional format.

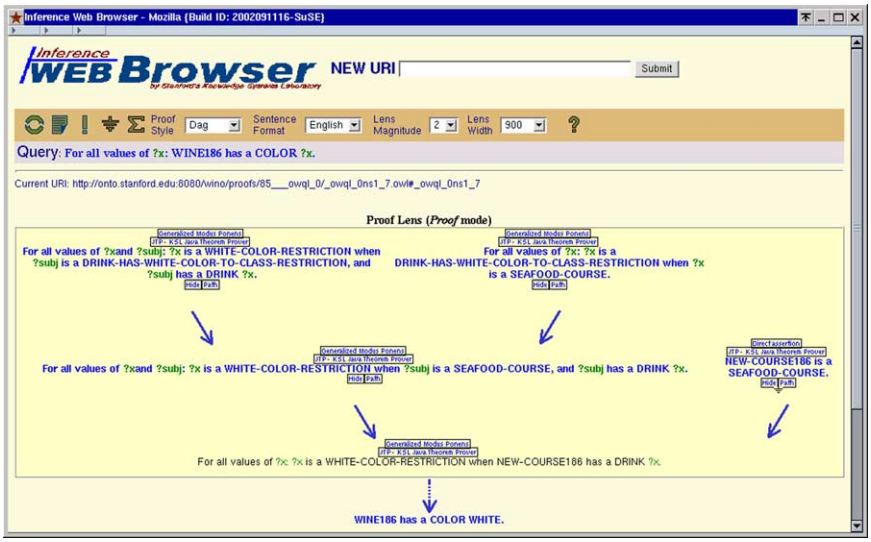

Fig. 5. An Inference Web browser screen.

IWBase meta information. The selection of an inference engine box presents details about the inference engine used to derive the actual theorem. The selection of an inference rule box presents a description of the rule. The selection of the source icon beside sentences associated with source documents presents details about sources where the axiom is defined. In Fig. 5, selecting a "Generalized Modus Ponens" box — the inference rule, would present information about JTP's Generalized Modus Ponens rule as in Fig. 3.

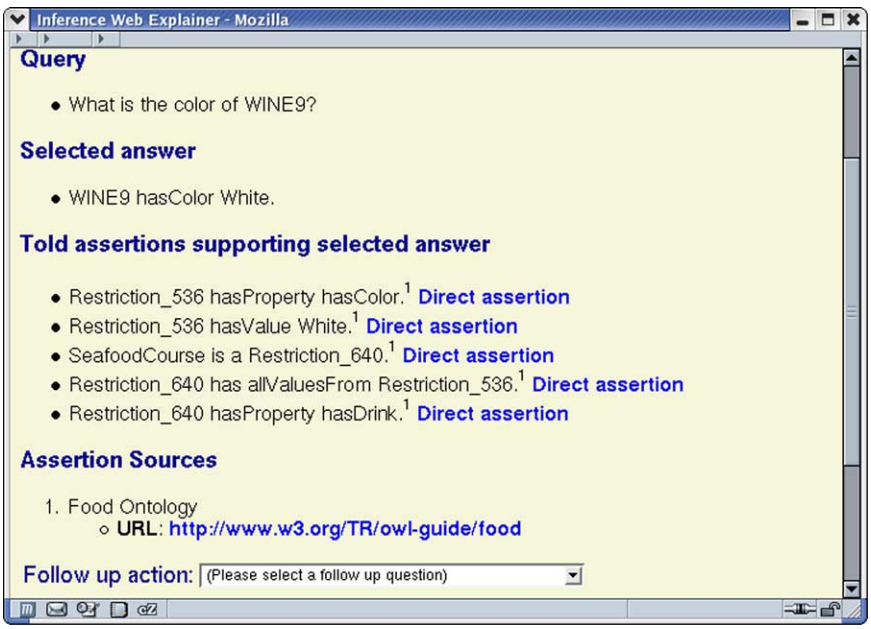

Fig. 6. An Inference Web explainer screen.

5.6. The explanation dialogue component

Inference Web includes a new explanation dialogue component that was motivated by usage observations. Fig. 6 shows an IW explainer snapshot explaining why `WINE9` has color white. The goal is to present a simple format that is a typical abstraction of useful information supporting a conclusion. The current instantiation provides a presentation of the question and answer, the ground facts on which the answer depended, and an abstraction of the meta information about those facts. There is also a follow-up action option that allows users to browse the proof or explanation, obtain the assumptions that were used, get more meta information about the sources, provide input to the system, etc. Additionally all information presented on any of the screens is "hot" and thus if someone clicked on any explanation element, they could obtain information about that element including its description and meta information. This interface is expected to be the interface with which the average human user of inference web interacts. We are currently in the mode of gathering feedback and requests from the user community for additional feature support.

6. Contributions and future work

The Wine Agent[19] and the DAML Query Language Front-End[20] are two example Semantic Web agents supported by the Inference Web. These agents are based on the Stanford's JTP theorem prover that produces PML proofs. The IWBase is populated with JTP information: one *InferenceEngine* entry for the reasoner itself, nine entries for its primitive inference rules, one entry for its set of DAML axioms, and 56 entries for the axioms. Using this registration of JTP and the fact that JTP dumps PML proofs, Inference Web can be used to present proofs and explanations of any of JTP's answers.

Beyond just explaining a single system, Inference Web attempts to incorporate best in class explanations and provide a way of combining and presenting proofs that are available. It does not take one stance on the form of the explanation since it allows deductive engines to dump single or multiple explanations of any deduction in the deductive language of their choice. It provides the user with flexibility in viewing fragments of single or multiple explanations in multiple formats. The new explanation dialogue component initially presents a summary of the question, answer, and the foundation for the answer along with minimal meta information that helps users evaluate at a glance, how to interpret an answer. That component then provides a follow-up question list along with a feedback option for learning.

IW attempts to minimize the burden for interoperability. IW simply requires inference rule registration and PML format. It does not limit itself to only explaining deductive engines. It provides a proof theoretic foundation on which to build and present its explanations, but any question answering system may be registered in the Inference Web and thus explained. More recently, we have begun integrating with query planners and extractors and focus more on explaining the process by which an answer was determined rather than the exact inference rules used to obtain a particular answer. This lets the Inference Web provide explanations for tasks that need to know what was done and also provide explanations for how something was done. For example, in joint work with IBM, Inference Web can be used to explain the markup that was generated from text, it can point to the text it used, the extractors that were used, and the meta information about the text such as its recency and authoritativeness ranking.

Revisiting the Inference Web requirements in Section 3, we can identify the following contributions:

- *Support for knowledge provenance* is provided by: the PML specification that allows node sets to be associated with sources; and the IWBase that supports meta information for annotating sources and provides database storage and access to the meta information.
- *Support for reasoning information* is provided by: the proof specification that supports a comprehensive representation of proof trees; and the IWBase that supports meta information for annotating inference engines along with their primitive inference rules. Also, the proof specification provides support for alternative justifications by allowing multiple inference steps per node set and the proof browser supports navigation of the information.

[19] http://www.ksl.stanford.edu/people/dlm/webont/wineAgent/.
[20] http://www.onto.stanford.edu:8080/dql/servlet/DQLFrontEnd.

- *Support for explanation generation* is provided by the IWBase that supports both formal and informal information about languages, axioms, axiom sets, derived and rewrite rules. Rewrite rules provide the key to abstracting complicated proofs into more understandable explanations. The proof support for alternative justifications allows derivations to be performed by performance reasoners with explanations being generated by alternative reasoners aimed at human consumption.
- *Support for distributed proofs* is provided by the IW architecture. Proofs are specified in PML (using the emerging web standard OWL so as to leverage XML-, RDF-, and OWL-based information services) and are interoperable. Proof fragments as well as entire proofs may be combined and interchanged.
- *Support for proof presentation* is provided by a lightweight proof browsing using the lens-based IW browser. The browser can present either pruned justifications or guided viewing of a complete reasoning path. Support is also provided by the explanation dialogue component to provide summaries and follow-up question support.

We have registered a few theorem provers and updated them so that they produce PML proofs. Inference Web can then be used to browse proofs and explanations of any answer produced by those reasoners. Inference web was originally aimed at explaining answers from theorem provers that encode a set of declaratively specified inference rules. More recently, we have looked at other kinds of reasoning engines such as JSAT and can now browse proofs generated from this satisfiability reasoner [30,23]. In joint work with Ambite, Knoblock and Muslea, we have also registered the ISI Mediator so that query plans can be presented by the Inference Web. In joint work with Ferrucci, Murdock, and Welty from IBM, we have enabled IBM to register a number of their text analytics engines to enable explanations of markup and KB generation from text. With these more recent efforts, we have broadened our notion of what kinds of inference rules and reasoners can be registered and thus broadened the kinds of question answering systems that can be explained.

Future work includes the registration of more question answering systems—whether they are theorem provers, planners, extractors, or of other types. We have encoded some rewrite rules that enable explanations to be generated from the more detailed proofs but this is an area of future work focus as well. Some simple examples of abstracting proofs can be seen on the Inference Web web site (e.g., http://iw.stanford.edu/documents-abstractions.html). Currently, we are developing tools for generating tactics that are required for explaining other proofs. We also intend to provide specialized support for why-not questions expanding upon [6] and [16]. We have also begun an effort to provide specialized support for explaining contradictory information. We anticipate special support for integration of proofs that include conflicting statements so that we can enable users to view conflicting evidence more easily. We are also looking at additional support for proof browsing and pruning. The initial explainer dialogue component provides version 1 in this effort but we envision this work to expand rapidly with broader user communities. We have also initiated conversations with the verification community in order to provide a PML format that meets their needs as well as meeting the needs of the applications that require explanation. Initial discussions at least for utilizing IWBase inference rule information with “correct-by-construction” software environments such as Specware[21] appear promising.

7. Conclusion

Inference Web enables applications and services to generate portable explanations of their conclusions. We identified the support for knowledge provenance, reasoning information, explanation generation, distributed proofs, and proof/explanation presentation as requirements for explanations in the web. We described the major components of IW, the PML specification based on the emerging web language, OWL supporting proofs and their explanations, the IWBase, the proof abstractor API, the IW browser, and the explanation dialogue component. We described how Inference Web features provide infrastructure for the identified requirements for web explanations. We facilitated use in a distributed environment by providing IW tools for registering and manipulating proofs, proof fragments, inference engines, ontologies, and source information.

[21] http://www.kestrel.edu/HTML/prototypes/specware.html.

We also facilitated interoperability by specifying the PML format and providing tools for manipulating proofs and fragments. We have implemented the IW approach for four Semantic Web agents (three of them based on JTP and one based on JSAT) and are in discussions with additional reasoner authors to include more reasoning engines. We have presented the work at government sponsored program meetings (RKF, DAML, PAL, AQUAINT, and NIMD) to gather input from other reasoner authors/users and have obtained feedback and interest. Current registration work includes IBM's UIMA, ISI's Mediator, SRI's SNARK, and W3C's CWM.

Acknowledgments

Many people have provided valuable input to our work. Thanks in particular go to current and past colleagues at KSL including Richard Fikes, Cynthia Chang, Jessica Jenkins, Gleb Frank, Eric Hsu, Bill MacCartney, Rob McCool, Sheila Mcllraith, Priyendra Deshwal, Kaan Baloglu and Yulin Li for input on JTP, our specification or applications. Thanks also go to Pat Hayes for his reviews of our work and joint work on PPDR. Thanks to many people from our government sponsored research programs for providing requirements and comments. Thanks to current collaborators on recent integrations, in particular Jose-Luis Ambite, Dave Ferrucci, Fausto Giunchiglia, Craig Knoblock, Bill Murdock, Pavel Shvaiko, Mark Stickel, and Chris Welty. All errors, of course are our responsibility.This work is supported by the following grants DARPA F30602-00-2-0579, N66001-00-C-8027, NBCHD030010, and ARDA H278000*000 and H768000*000/4400049114.

References

[1] A. Borgida, E. Franconi, I. Horrocks, Explaining ACL subsumption, in: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI2000), IOS Press, 2000, pp. 209–213.

[2] A. Borgida, E. Franconi, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, Explaining ALC subsumption, in: Proceedings of the International Workshop on Description Logics (DL'99), Linköping, Sweden, 1999, pp. 33–36.

[3] A. Borgida, D.L. McGuinness, Asking queries about frames, in: Proceedings of Fifth International Conference on the Principles of Knowledge Representation and Reasoning, Cambridge, MA, Morgan Kaufmann, 1996.

[4] R. Boyer, M. Kaufmann, J. Moore, The Boyer-Moore theorem prover and its interactive enhancements, Computers Math. Appl. 29 (2) (1995) 27–62.

[5] P. Buneman, S. Khanna, W.-C. Tan, Why and where: a characterization of data provenance, in: Proceedings of 8th International Conference on Database Theory, 2001, pp. 316–330.

[6] H. Chalupsky, T. Russ, Why not: debugging failed queries in large knowledge bases, in: Proceedings of the 14th Innovative Applications of Artificial Intelligence Conference (IAAI-02), 2002, pp. 870–877.

[7] A. Das, W. Wu, D.L. McGuinness, Industrial strength ontology management, in: I. Cruz, S. Decker, J. Euzenat, D.L. McGuinness (Eds.), The Emerging Semantic Web, IOS Press, 2002.

[8] A. Felty, D. Miller, Proof Explanation and Revision, Technical Report MSCIS8817, University of Pennsylvania, 1987.

[9] D. Ferrucci, A. Lally, UIMA: an architectural approach to unstructured information processing in the corporate research environment, J. Nat. Language Eng. (2004), in press.

[10] R. Fikes, P. Hayes, I. Horrocks, DAML Query Language (DQL) Abstract Specification, Technical report, W3C, 2002.

[11] R. Fikes, D.L. McGuinness, An Axiomatic Semantics for RDF, RDF-S, and DAML + OIL (March 2001), Technical Report Note 18, W3C, December 2001.

[12] F. Giunchiglia, P. Shvaiko, Semantic matching, Technical Report, vol. 71, CEUR, WS, 2003.

[13] R.V. Guha, R. McCool, E. Miller, Semantic search, in: Proceedings of the 12th International World Wide Web Conference (WWW2003), Budapest, Hungary, ACM Press, 2003, pp. 700–709.

[14] X. Huang, Reconstructing proofs at the assertion level, in: Proceedings of CADE-94, LNAI-814, Springer, 1994, pp. 738–752.

[15] E.F. Kendall, M.E. Dutra, D.L. McGuinness, Towards a commercial ontology development environment, in: International Semantic Web Conference Late Breaking Topics, Sadinia, Italy, 9–12 June 2002.

[16] D.L. McGuinness, Explaining Reasoning in Description Logics, PhD thesis, Rutgers University, 1996.

[17] D.L. McGuinness, Ontologies come of age, in: D. Fensel, J.A. Hendler, H. Lieberman, W. Wahlster (Eds.), Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2003, pp. 171–194.

[18] D.L. McGuinness, A. Borgida, Explaining subsumption in description logics, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada, Morgan Kaufmann, 1995, pp. 816–821.

[19] D.L. McGuinness, P. Patel-Schneider, From description logic provers to knowledge representation systems, in: F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003, pp. 265–281.

[20] D.L. McGuinness, P.P. da Silva, Trusting answers on the web, in: M.T. Maybury (Ed.), New Directions in Question Answering, AAAI/MIT Press, in press.

[21] D.L. McGuinness, P.P. da Silva, Infrastructure for web explanations, in: D. Fensel, K. Sycara, J. Mylopoulos (Eds.), Proceedings of 2nd International Semantic Web Conference (ISWC2003), LNCS-2870, Sanibel, FL, USA, Springer, 2003, pp. 113–129.
[22] D.L. McGuinness, P.P. da Silva, Registry-based support for information integration, in: Proceedings of IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico, 2003, pp. 117–122.
[23] D.L. McGuinness, P. Shvaiko, F. Giunchiglia, P.P. da Silva, Towards explaining semantic matching, in: Volker Haarslev, Ralf Moller (Eds.), Proceedings of the 2004 International Workshop on Description Logics, vol. 104, CEUR-WS, 2004.
[24] D.L. McGuinness, F. van Harmelen, OWL Web Ontology Language Overview, Technical report, World Wide Web Consortium (W3C), 10 February 2004 (Recommendation).
[25] D.L. McGuinness, J. Wright, An industrial strength description logic-based configurator platform, IEEE Intell. Syst. 13 (4) (1998) 69–77, July/August.
[26] P.P. da Silva, P. Hayes, D.L. McGuinness, R. Fikes, PPDR: A Proof Protocol for Deductive Reasoning, Technical Report KSL-04-04, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA, March 2004.
[27] P.P. da Silva, D.L. McGuinness, R. Fikes, A Proof Markup Language for Semantic Web Services, Technical Report KSL-04-01, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA, January 2004.
[28] P.P. da Silva, D.L. McGuinness, R. McCool, Knowledge provenance infrastructure, IEEE Data Eng. Bull. 25 (2) (2003) 179–227.
[29] E.H. Shortliffe, Computer-Based Medical Consultations: MYCIN, Elsevier, North Holland, New York, USA, 1976.
[30] P. Shvaiko, F. Giunchiglia, P.P. da Silva, D.L. McGuinness, Web Explanations for Semantic Heterogeneity Discovery, Technical Report KSL-04-02, Knowledge Systems Laboratory, Stanford University, Stanford, CA, January 2004.
[31] M. Smith, D.L. McGuinness, R. Volz, C. Welty, Web Ontology Language (OWL) Guide Version 1.0, Technical Report Working Draft, World Wide Web Committee (W3C), 2003.
[32] W. Swartout, C. Paris, J. Moore, Explanations in knowledge systems: design for explainable expert systems, IEEE Intell. Syst. (June) (1991).