Infrastructure for Web Explanations

Deborah L. McGuinness and Paulo Pinheiro da Silva

Knowledge Systems Laboratory,
Stanford University, Stanford CA 94305
{dlm,pp}@ksl.stanford.edu

Abstract

The Semantic Web lacks support for explaining knowledge provenance. When web 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. The Semantic Web also lacks support for explaining reasoning paths used to derive answers. The Inference Web (IW) aims to take opaque query answers and make the answers more transparent by providing explanations. The explanations include information concerning where answers came from and how they were derived (or retrieved). In this paper we describe an infrastructure for IW explanations. The infrastructure includes: an extensible web-based registry containing details on information sources, reasoners, languages, and rewrite rules; a portable proof specification; and a proof and explanation browser. Source information in the IW registry is used to convey knowledge provenance. Representation and reasoning language axioms and rewrite rules in the IW registry are used to support proofs, proof combination, and semantic web agent interoperability. The IW browser is used to support navigation and presentations of proofs and their explanations. The Inference Web is in use by two Semantic Web agents using an embedded reasoning engine fully registered in the IW. Additional reasoning engine registration is underway in order to help provide input for evaluation of the adequacy, breadth, and scalability of our approach.

1 Introduction

Inference Web (IW) aims to enable applications to generate portable and distributed explanations for any of their answers. IW addresses needs that arise with systems performing reasoning and retrieval tasks in heterogeneous environments such as the web. Users (humans and computer agents) need to decide when to trust answers from varied sources. We believe that the key to trust is understanding. Explanations of knowledge provenance and derivation history can be used to provide that understanding [16]. In the simplest case, users would 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 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 may agree to trust the deductions if they know what reasoner was used to deduce answers and what data sources were used in the proof. 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. It also addresses the issues of proof tracing 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 portable proof specification.

In this paper, 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 portable proof specification, the registry [17] (containing information about inference engines, proof methods, ontologies, and languages and their axioms), the explanation mechanism, 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.

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 [18]), 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 [20] on generating explanations.

IW also builds on the learnings of explanation in description logics (e.g., [1, 2, 13, 14]) which attempt to provide a logical infrastructure for separating pieces of logical proofs and automatically generating follow-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 also attempts to integrate learnings from the theorem proving community on proof presentation(e.g., [4, 9]) and explanation (e.g., [12]), 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 paper also builds on experience designing query components for frame-like systems [3, 10, 13] 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., [15]) and ontology environments (e.g., [8]). 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]).

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

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

[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/Novel_Intelligence/
[6] http://www.w3.org/2000/10/swap/doc/cwm.html
[7] http://www.agfa.com/w3c/euler/

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

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. Requirements as a result of reasoning may include the following:

- The reasoner used
- Reasoning method (e.g., tableaux, model elimination, etc.)
- Inference rules supported by the reasoner
- Reasoner soundness and completeness properties
- Reasoner assumptions (e.g., closed world vs. open world, unique names assumption, etc.)
- Reasoner authors, version, etc.
- 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 ¬A)
- Support for alternative reasoning paths to a single conclusion

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, we view an explanation 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. Requirements for explanations may include:

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

3.4 Support for Distributed Proofs

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 an language for sharing proofs).

3.5 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 followup 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:

- A method for asking for explanations (or proofs)
- A way of breaking up proofs into manageable pieces
- A method for pruning proofs and explanations to help the user find relevant information
- A method for allowing proof and explanation navigation (including the ability to ask followup questions)
- A presentation solution compatible with web browsers
- A way of seeing alternative justifications for answers

4 Use Cases

Every combination of a query language with a query-answering environment is a potential new context for the Inference Web. We provide two motivating scenarios. 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 used previously is 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 DQL[8] for retrieving information such as "zinfandels from Napa Valley" or "wine recommended for serving with a spicy red meat meal" (as exemplified in the wine agent example in the OWL guide document [19]). 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.

[8]http://www.daml.org/2002/08/dql/

- 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 Inference Web registry for storing the meta information and the Inference Web registrar web application for handling the registry. This addresses the issues related to knowledge provenance.
- 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 the portable proof format. It can display multiple formats including KIF[9] and English. Additionally, it is capable of using rewrite rules (or tactics) to abstract proofs in order to provide more understandable explanations. This addresses the issues related to reasoning, explanations, and presentation.

5 Inference Web

Inference Web contains both data used for proof manipulation and tools for building, maintaining, presenting, and manipulating proofs. Figure 1 presents an abstract and partial view of the Inference Web framework[10]. There, Inference Web data includes proofs and explanations published anywhere on the web. Inference and search engines can generate proofs using the IW format. The explainer, an IW tool, can abstract proofs into explanations. Inference Web data also has a distributed repository of meta-data including sources, inference engines, inference rules and ontologies. In addition to the explainer, Inference Web tools include a registrar for interacting with the registry, a browser for displaying proofs, and planned future tools such as proof web-search engines, proof verifiers, proof combinators, and truth maintenance systems. In this paper, we limit our discussion to the portable proofs (and an associated parser), registry (and the associated registrar tools), explanations, and the browser.

5.1 Portable Proof

The Inference Web provides a proof specification written in the web markup language DAML+OIL[11] [7]. Proofs dumped in the portable proof format become a portion of the Inference Web data used for combining and presenting proofs and for generating explanations. Our portable proof specification includes two major components of IW proof trees: inference steps and node sets. Proof metadata as described in Section 5.2 are the other components of our proof specification.

Figure 2 presents a typical dump of an IW node set. Each node set is labeled by a well formed formula (WFF) written in KIF. (In this example, the node set

[9] http://logic.stanford.edu/kif/kif.html
[10] A more detailed view is available at http://ksl.stanford.edu/software/IW/details.shtml
[11] An OWL version is coming soon.

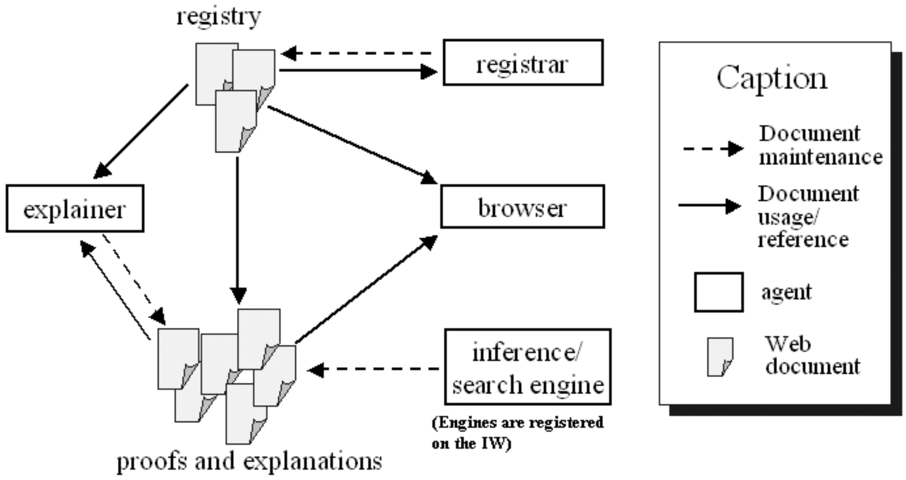

Figure 1: Inference Web framework overview.

is labeled with a WFF stating that the color of `W1` is `?x` or the value of the color property of `Wine1` 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. It is a set because there could be multiple deductive paths leading to the statement.

Figure 2 shows an instance of a node set, an inference step, and a reference to an inference rule. There is no source associated with this node set 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. In general, each node set can be associated with multiple, one, or no inference steps as described by the `iw:isConsequentOf` property of the node set in Figure 2. A proof can then be defined as a tree of inference steps explaining the process of deducing the consequent sentence. Thus, a proof can physically vary from a single file containing all its node sets to many files, each one containing a single node set. Also, 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 an IW proof is a forest of trees since the nodes of a proof tree are sets of inference steps. In contrast with typical proof trees that are composed of nodes rather than node sets, every theorem in an IW proof can have multiple justifications.

An inference step is a single application of an inference rule, whether the rule is atomic or derived as discussed in Section 5.2. Inference rules (such as modus ponens) can be used to deduce a consequent (a well formed formula) from any number of antecedents (also well formed formulae). Inference steps contain pointers to proof nodes of its antecedents, the inference rule used, and any variable bindings used in the step. The antecedent sentence in an inference step

```
<?xml version='1.0'?> <rdf:RDF (...)>
<iw:NodeSet rdf:about='../sample/IW1.daml#IW1'>
    <iw:NodeSetContent>
      <iw:KIF>
         <iw:Statement>(wines:COLOR W1 ?x)</iw:Statement>
      </iw:KIF>
    </iw:NodeSetContent>
    <iw:isConsequentOf rdf:parseType='daml:collection'>
         (a NodeSet can be associated to a set of Inference steps)
     <iw:InferenceStep>
        <iw:hasInferenceRule rdf:parseType='daml:collection'>
          <iw:InferenceRule rdf:about='../registry/IR/GMP.daml'/>
        </iw:hasInferenceRule>
        <iw:hasInferenceEngine rdf:parseType='daml:collection'>
           <iw:InferenceEngine rdf:about='../registry/IE/JTP.daml'/>
         </iw:hasInferenceEngine>
         (...)
        <iw:has Antecedent rdf:parseType='daml:collection'>
            (inference step antecedents are IW files with their own URIs)
          <iw:NodeSet rdf:about='../sample/IW3.daml#IW3'/>
          <iw:NodeSet rdf:about='../sample/IW4.daml#IW3'/>
        </iw:hasAntecedent>
        <iw:hasVariableMapping
                rdf:type='http://www.daml.org/2001/03/daml+oil#List'/>
        (...)
      </iw:InferenceStep>
    </iw:isConsequentOf>
  </iw:NodeSet>
</rdf:RDF>
```

Figure 2: An Inference Web Proof.

may come from inference steps in other node sets, existing ontologies, extraction from documents, or they may be assumptions.

With respect to a query, a logical starting point for a proof in Inference Web is a proof fragment containing the last inference step used to derive a node set that contains the answer sentence for the query. Any inference step can be presented as a stand alone, meaningful proof fragment as it contains the inference rule used with links to its antecedents and variable bindings. 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, the Inference Web provides a web service that dumps 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 registry entries.

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 Figure 2 is available at http://www.ksl.stanford.edu/software/IW/spec/iw.daml.

5.2 Registry

The IW registry is a hierarchical interconnection of distributed repositories of information relevant to proofs and explanations. Entries in the registry contain the information linked to in the proofs. Every entry in the registry is a file written in DAML+OIL. Also, every entry is an instance of a registry concept. *InferenceEngine*, *Language* and *Source* are the core concepts in the registry. Other concepts in the registry are related to one of these core concepts.

In order to interact with the IW registry, the IW provides a web agent registrar that supports users in updating or browsing the registry. The registrar may grant update or access privileges on a concept basis and it may define and implement policies for accessing the registry. The current demonstration registrar is available at: http://onto.stanford.edu:8080/iwregistrar/.

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, Figure 2 shows that the `iw:hasInferenceEngine` property of `iw:InferenceStep` has a pointer to `JTP.daml`, which is the registry meta information about Stanford's JTP[12] model-elimination theorem prover. Inference engines may have the following properties associated with them: name, URL, author(s), date, version number, organization, etc. *InferenceRule* is one of the most important concepts associated with *InferenceEngine*. With respect to an inference engine, registered rules can be either atomic or derived from other registered rules.

A screen shot from an IW registrar interface browsing the entry for the generalized modus ponens (GMP) rule is presented in Figure 3. GMP is an atomic inference rule for JTP[13]. Each of the inference rules includes a name, description, optional example, and optional formal specification. An inference rule is formally specified by a set of sentences patterns for its premises, a sentence pattern for its conclusion, and optional side conditions. Patterns and conditions are specified using KIF and a set of name conventions for KIF arguments. For example, an argument `@Si` such as the `@S1` and `@S2` in Figure 3 says that it can be bound to a sentence while `@SSi` says that it can be bound to a set of sentences. 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 registry before the reasoner is actually used to generate IW 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] for example where the performance tableaux

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

[13] GMP or any rule may be atomic for one reasoner while it may be derived for another reasoner.

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. Inference Web's registry, when fully populated, will contain inference rule sets for many common reasoning systems. In this case, users may view inference rule sets to help them decide whether to use a particular inference engine.

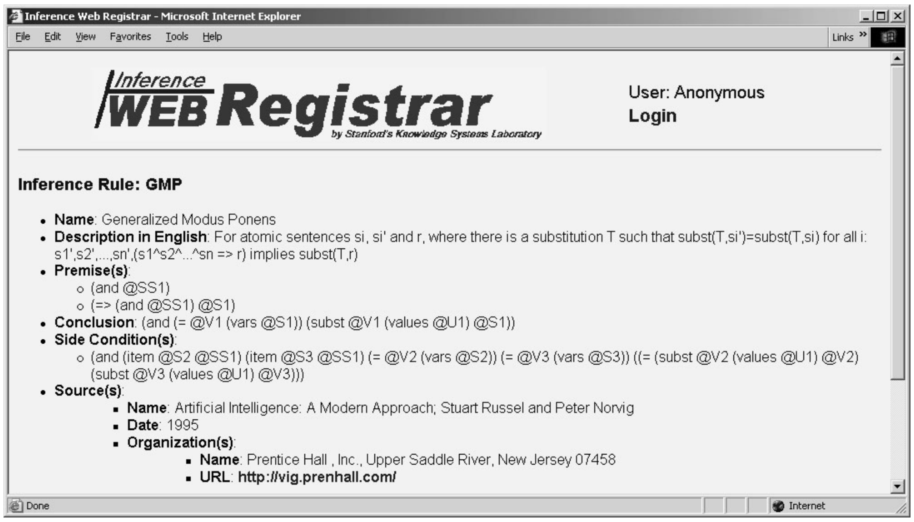

Figure 3: Sample registry entry for an inference rule.

Inference engines may use specialized language axioms to support a language such as DAML or RDF. *Language* is a core IW 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 *LanguageAxiomSet* 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 while another reasoner may use a slightly different set. 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 KIF.

Source is the other core concept of the registry. *Source* is specialized into five basic classes: *Person*, *Team*, *Publication*, *Ontology*, and *Organization*. At the moment, we are expanding the specification of (authoritative) sources as required. Thus, we are keeping a minimal description of these sources in the initial specification used in the IW. 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. Figure 4 contains a sample ontology registry entry for the ontology used in our wine examples.

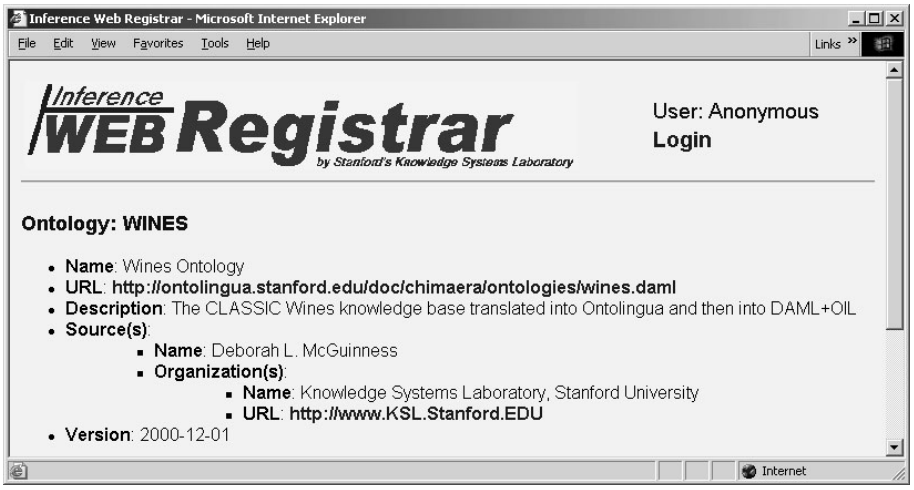

Figure 4: Sample registry entry for an ontology.

5.3 Explanations

Although essential for automated reasoning, inference rules such as those used by theorem provers and registered in the registry 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 [12]. 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 aggregating some semantics in order to make the rules more understandable. Entries of *DerivedRule* are the natural candidates for storing specialized sets of rewriting rules. In the IW, tactics are rules associated with axioms, and are used independent of whether a rule is atomic or derived.

Many intermediate results are "dropped" along with their supporting axioms, thereby abstracting the structure of proofs. The user may always ask follow-up questions and still obtain the detail, however the default explanation provides abstracted explanations. 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://www.ksl.stanford.edu/software/iw/Ex1/. The implementation of the IW explainer is work in progress. The explainer algorithm generate explanations in a systematic way using the derived rules related to a selected language axiom set.

5.4 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" and "KIF" formats[14]. We also expect that some applications may implement their own displays using the IW API. The 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 derived sentences and antecedent sentences.

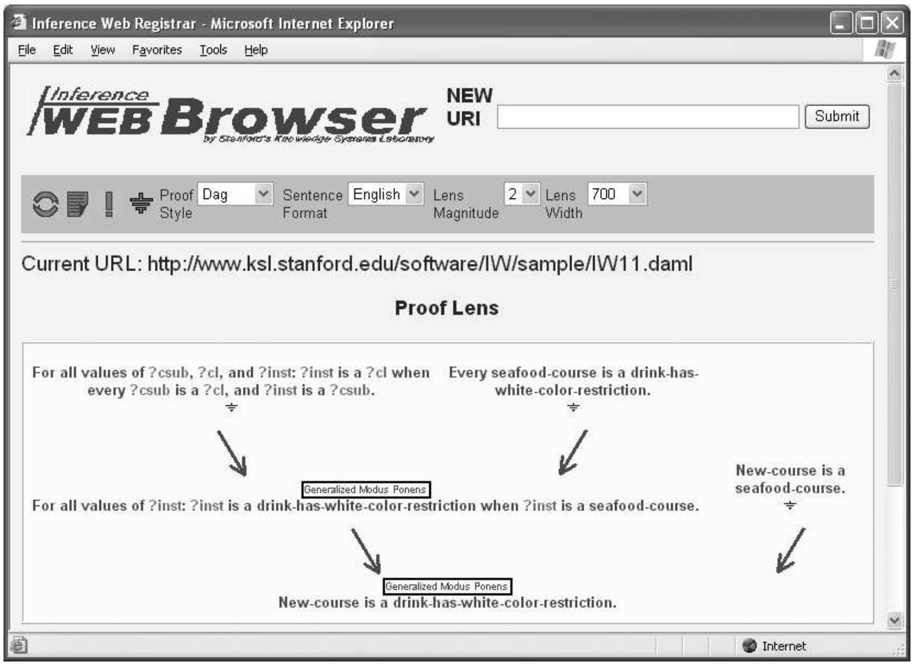

Figure 5: An Inference Web Browser screen.

Figure 5 presents two levels of inference step for one wine use case in Section 4. Prior to this view, the program has asked what wine to serve with a seafood course. In Figure 5, one can see 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 it including its antecedents and the antecedent's derivations.

We believe that one of the keys to presentation of justifications is breaking

[14]Current investigations are underway for an N3 format as well.

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 registry meta-information. The selection of the consequent presents details about the inference engine used to derive the actual theorem. The selection of an inference rule 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 Figure 5, selecting the consequent would present information about JTP - the inference engine used to derive it. Selecting GMP - the inference rule, would present information about JTP's Generalized Modus Ponens rule as presented in Figure 3.

6 Contributions and Future Work

The Wine Agent[15] and the DAML Query Language Front-End[16] are Semantic Web agents supported by the Inference Web. These agents are based on the Stanford's JTP theorem prover that produces portable proofs. The IW registry 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.

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. IW simply requires inference rule registration and portable proof format.

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

- *Support for knowledge provenance* is provided by: the portable proof specification that allows node sets to be associated with sources; and the registry that supports meta information for annotating sources.

- *Support for reasoning information* is provided by: the proof specification that supports a comprehensive representation of proof trees; and the registry 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.

[15]http://onto.stanford.edu:8080/wino/
[16]http://onto.stanford.edu:8080/dql/servlet/DQLFrontEnd

- *Support for explanation generation* is provided by the registry 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* are provided by the IW architecture. Portable proofs are specified in the emerging web standard DAML+OIL so as to leverage XML-, RDF-, and DAML-based information services. 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.

We are currently extending SRI's SNARK[17] theorem prover to produce portable proofs and simultaneously populating the IW registry with SNARK information. Also, we are in the process of discussing the registration of the W3C's CWM[18] theorem prover in the Inference Web.

Future work includes the registration of more inference engines. Explanations for some proofs can be automatically created as presented in http://www.ksl.stanford.edu/software/IW/Ex1. 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 [13]. We are also looking at additional support for proof browsing and pruning. We have also initiated conversations with the verification community in order to provide a portable proof and registry format that meets their needs as well as meeting the needs of the applications that require explanation. Initial discussions at least for utilizing registry inference rule information with "correct-by-construction" software environments such as Specware[19] appear promising.

7 Conclusion

Inference Web enables applications to generate portable explanations of their conclusions. We identified the support for knowledge provenance, reasoning information, explanation generation, distributed proofs, and proof presentation as requirements for explanations in the web. We described the major components of IW - the portable proof specification based on the emerging web language-DAML (soon to be updated to OWL) supporting proofs and their explanations, the registry, and the IW proof browser. We described how Inference Web features provide infrastructure for the identified requirements for web explanations.

[17] http://www.ai.sri.com/~stickel/snark.html
[18] http://www.w3.org/2000/10/swap/doc/cwm.html
[19] http://www.kestrel.edu/HTML/prototypes/specware.html

We facilitated use in a distributed environment by providing IW tools for registering and manipulating proofs, proof fragments, inference engines, ontologies, and source information. We also facilitated interoperability by specifying the portable proof format and providing tools for manipulating proofs and fragments. We have implemented the IW approach for two Semantic Web agents based on JTP 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, AQUAINT, and NIMD) to gather input from other reasoner authors/users and have obtained feedback and interest. Current registration work includes SRI's SNARK and W3C's CWM.

Acknowledgments Many people have provided valuable input to our work. Thanks in particular go to colleagues at KSL including Richard Fikes, Jessica Jenkins, Gleb Frank, Eric Hsu, Bill MacCartney, Rob McCool, Sheila McIlraith, and Yulin Li for input on JTP, our specification or applications. Also thanks go to a number of colleagues in some government programs who provided input including Hans Chalupsky, Peter Clark, Ken Forbus, Ken Murray, and Steve Reed. 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] Alex Borgida, Enrico Franconi, and Ian Horrocks. Explaining $\mathcal{ACL}$ Subsumption. In *Proc. of the 14th European Conf. on Artificial Intelligence (ECAI2000)*, pages 209–213. IOS Press, 2000.

[2] Alex Borgida, Enrico Franconi, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. Explaining $\mathcal{ALC}$ Subsumption. In *Proc. of the International Workshop on Description Logics (DL'99)*, pages 33–36, Linköping, Sweden, July 1999.

[3] Alex Borgida and Deborah L. McGuinness. Asking Queries about Frames. In *Proceedings of Fifth International Conference on the Principles of Knowledge Representation and Reasoning*, Cambridge, Massachusetts, November 1996. Morgan Kaufmann.

[4] Robert Boyer, Matt Kaufmann, and J. Moore. The Boyer-Moore Theorem Prover and Its Interactive Enhancements. *Computers and Mathematics with Applications*, 29(2):27–62, 1995.

[5] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and Where: A Characterization of Data Provenance. In *Proceedings of 8th International Conference on Database Theory*, pages 316–330, January 2001.

[6] Hans Chalupsky and Tom Russ. WhyNot: Debugging Failed Queries in Large Knowledge Bases. In *Proc. of the 14th Innovative Applications of Artificial Intelligence Conference (IAAI-02)*, pages 870–877, 2002.

[7] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL (March 2001) Reference Description. Technical Report Note 18, World Wide Web Committee (W3C), December 2001.

[8] Aseem Das, Wei Wu, and Deborah L. McGuinness. Industrial Strength Ontology Management. In Isabel Cruz, Stefan Decker, Jerome Euzenat, and Deborah L. McGuinness, editors, *The Emerging Semantic Web*. IOS Press, 2002.

[9] Amy Felty and Dale Miller. Proof Explanation and Revision. Technical Report MSCIS8817, University of Pennsylvania, 1987.

[10] Richard Fikes, Pat Hayes, and Ian Horrocks. DAML Query Language (DQL) Abstract Specification. Technical report, W3C, 2002.

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

[12] Xiaorong Huang. Reconstructing Proofs at the Assertion Level. In *Proceedings of CADE-94*, LNAI-814, pages 738–752. Springer, 1994.

[13] Deborah L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Rutgers University, 1996.

[14] Deborah L. McGuinness and Alex Borgida. Explaining Subsumption in Description Logics. In *Proc. of the 14th International Joint Conference on Artificial Intelligence*, pages 816–821, Montreal, Canada, August 1995. Morgan Kaufmann.

[15] Deborah L. McGuinness and Peter Patel-Schneider. From Description Logic Provers to Knowledge Representation Systems. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 265–281. Cambridge University Press, 2003.

[16] Deborah L. McGuinness and Paulo Pinheiro da Silva. Trusting Answers on the Web. In Mark T. Maybury, editor, *New Directions in Question Answering*. AAAI/MIT Press. To appear.

[17] Deborah L. McGuinness and Paulo Pinheiro da Silva. Registry-Based Support for Information Integration. In *Proceedings of IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, pages 117–122, Acapulco, Mexico, August 2003.

[18] Edward Hance Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier/North Holland, New York, NY, USA, 1976.

[19] Michael Smith, Deborah L. McGuinness, Raphael Volz, and Chris Welty. Web Ontology Language (OWL) Guide Version 1.0. Technical Report Working Draft, World Wide Web Committee (W3C), 2003.

[20] W. Swartout, C. Paris, and J. Moore. Explanations in Knowledge Systems: Design for Explainable Expert Systems. *IEEE Intelligent Systems*, June 1991.