IWBase: Provenance Metadata Infrastructure for Explaining and Trusting Answers from the Web

Deborah L. McGuinness Paulo Pinheiro da Silva
Cynthia Chang

Knowledge Systems Laboratory, Stanford University
Stanford, CA 94305, USA.
e-mail: {dlm,pp,csc}@ksl.stanford.edu

Abstract

If users are expected to depend on answers from applications and services on the web, they need to have methods for asking for information that will increase their understanding and their trust in the answers. One way of increasing a user's understanding and trust is to provide a summarized or detailed description of how the answer was obtained. This process would typically depend on some kind of proof structure defending the answer but this alone (no matter how understandable and/or thorough the proof is) is not enough. Any presentation of why a user should depend on an answer also critically depends on metadata concerning elements used in the proof such as underlying data sources, their authoritativeness, reasoning methods used, any uncertainties, etc. In this paper, we introduce IWBase, a comprehensive OWL-based distributed and scalable infrastructure for effectively creating, storing, evolving, and using proof-related metadata for explanations. IWBase is currently in use by a number of DARPA and ARDA projects.

1 Introduction

Our research aims to improve user acceptance of answers by making them "actionable" and defensible. Answers will be actionable if users have enough information about the answer to improve their understanding to a point that they trust the answer and are ready to take action using the information. Answers are defensible if the user requesting the answer can justify why it is valid. Our approach is to provide explanatory information about the question answering process that may expose how the answer was generated. Thus, explanations can be directly presented to users or can be used to derive alternative explanations. Either way, explanations depend on provenance metadata and provenance metadata infrastructure to make answers actionable and defensible. Without

metadata, users may be unable to understand explanations because, for example, they may be unable to gather critical information about sources accessed, question answering systems, and inference methods. Without metadata, tools may be unable to derive alternative (potentially more understandable) explanations because the tools may not know which rules (rewriting rules) to use for abstracting explanations. Without metadata, users may not trust answers because because they may not have access to tools for checking the explanations and for computing trust values for answers. Without metadata infrastructure, it may also become too difficult for users, humans and agents, to include, maintain, evolve and query provenance metadata.

In this paper, we describe what is required to enable explanations that improve understanding and trust. These explanations may include information about the knowledge sources used as well as the processes involved for accessing and integrating knowledge. If answers are obtained simply by looking up stored information in a data source, then justifications need to include provenance information about the data source such as its recency, authoritativeness, etc. If answers are retrieved from multiple sources and combined in some manner, then information about all sources used as well as combination techniques needs to be accessible to interested users. If answers are obtained by information manipulation processes, then justifications need to be informed by descriptions of the manipulation processes such as their authors, versions, inference strategies, and assumptions. This kind of meta information is required in addition to the actual proof trace of the manipulations that were performed.

It may be worth noting that our original explanation focus was on the information manipulation area since our early target question answering applications used reliable and familiar data sources and used complicated (and potentially distributed) reasoning processes to reach conclusions. In those situations, users were familiar with and trusted the source data but needed help in understanding the deduction paths leading to conclusions. As our application focus expanded to more typical web applications with diverse data sources, some of which were likely to be unknown to users obtaining the final answers, it became clear that providing the option of accessing provenance information was required, even if just in a summary mode so that users could quickly check source usage. For example, some question answering applications rely on knowledge bases that were built and maintained using information extraction techniques, such as those in UIMA [5]. The original data may be in free text sources where some of the free text may be from known places like the AP news wire and other text sources may be unfamiliar (and of unknown quality and recency). Users obtaining answers based on these knowledge bases need to have the option of finding out which raw text sources were used, when they were updated, how they were transformed into structured knowledge bases, and then later how the information was manipulated.

This paper introduces our research thrust on provenance metadata and provenance metadata infrastructure that is required to present comprehensible and useful justifications of answers. The paper will describe the goals that emerged from our effort to provide interoperable, distributed explanations in

web settings. We will introduce the terminology used in our approach and then describe the distributed registries of provenance information required for explanations and the supporting infrastructure for maintaining the registries. We will introduce IWBase as a distributed repository of proof-related metadata that provides a solution to our provenance requirements. We will present our the IW architecture revealing the infrastructure (including registry and support services) that provides a solution that is maintainable, scalable, distributable, while supporting privacy and usability requirements. We will follow with an example of provenance meta data in explanations. We will conclude with a discussion of the relative merits of the approach, current and planned usage of the implementation, and related work.

2 Requirements and Principles

In this section, we will discuss the requirements that arise while trying to achieve the goal of improving user (and agent) trust in answers generated from distributed web-based applications. The requirements fall into three categories: those relating to requirements for the solution infrastructure, those relating to provenance of information, and those relating to applications that reason with information in potentially interoperable manners.

We recognize that web applications may be distributed, large, reference information from multiple sources of varying quality and recency, and may serve broad communities having diverse access needs. We have identified the following goals for our provenance-based solution.

- *Distributable*: The approach should be amenable to providing an architecture that can access information that is not all resident in one place in order to interact with potentially distributed web applications.
- *Scalable*: The approach should be able to scale with web applications that may grow in orders of magnitude as a result of rapidly expanding web-based data sources and user communities.
- *Maintainable*: The approach should include a scheme by which information may be updated by authorized users and agents so that its information is up to date. Additionally, the maintenance tools should provide enough support so that updates are simple to do and automatable where possible, since the value of the meta-data is directly tied to is recency.
- *Privacy*: The approach should be able to utilize information that may not necessarily be viewable by all users, thus it must incorporate a privacy scheme that supports appropriate updates, maintenance, and presentation of private information only to authorized users and agents.

Solutions that improve trust in answers by offering information about the data that was accessed during the answer generation need to provide information about the data. We have identified the following requirements for the

provenance as a result of gathering requirements from a number of government sponsored research programs including the High Performance Knowledge Base program[1], Rapid Knowledge Formation Program[2], the DARPA Agent Markup Language Program[3]), 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., [14]) and ontology environments (e.g., [4, 12]). We have also gathered needs from the World Wide Web Consortium efforts on CWM[6] and the related reasoner effort on Euler[7]. Additionally, we gathered knowledge provenance requirements from the programs above and from previous work on data provenance from the database community(e.g., [2]) and more recently from work integrating information from extractors such as the work in Tap[8] [8] leading to our enhanced knowledge provenance infrastructure [22] 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 obtained from efforts to explain text analytics work (e.g., IBM's UIMA [5]) as well as efforts to explain semantic matches using satisfiability engines (e.g., [7]) as work to explain emerging semantic web reasoners such as Pellet [19], where we are working on explaining contradictions.

More details on provenance metadata requirements are available in [15] and [22], with a summary here.

Knowledge provenance includes information about the origins of data. It includes

- Source name If facts are encountered in multiple sources, any integrated solution needs to have a way of identifying from which source information was taken and potentially which portion of the source was used.
- 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?, is this source known to be known and trusted by the user or agent or someone trusted by the agent?
- 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,

[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/
[8] http://tap.stanford.edu
[9] http://www.isi.edu/info-agents/Prometheus/
[10] http://www.fetch.com

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.)

- Term or phrase meaning (in natural language or a formal language)
- Term inter-relationships (ontological relations including subclass, superclass, part-of, etc.)

Additionally, there are requirements for meta information that arise from security, access, efficiency, and usage.

- 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, technical depth, 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

Finally, there are requirements on meta data that arise from the fact that meta data may be reasoned with and this reasoning may happen in a distributed manner, thus interoperability becomes a requirement. Because applications may use varying reasoning methods, provenance information is required about the information manipulation techniques.

- 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 vs. open world, unique names assumption, etc.)
- Reasoner authors, version, etc.

Explanation of the information manipulations goes beyond the scope of this paper but more information on this topic is addressed in papers on explanations of reasoning based on traces of reasoning steps and the Proof Markup Language [21]

3 Terminology

In this section, we will introduce the terminology used in our approach. The IWBase architecture described in Section 4 has two main concepts: *node element*s and the IWBase *node*. By defining node elements, we describe how meta-data is classified and represented. By defining IWBase node, we describe the evolving infrastructure for storing, maintaining and using proof-related metadata on the Web.

3.1 Node Elements

Node elements[11] are also Proof Markup Language (PML) elements. They are instances of PML classes, which are OWL [17] classes (thus they are subclasses of `owl:Class`). They are used to build OWL documents representing both proofs and proof provenance information. Thus, PML concepts can be considered to be either *proof level concepts* representing proof elements directly or *provenance level concepts* representing information about terms used in a proof.

Proof elements are PML elements at the proof level and they are used for building interchangeable, distributed proofs and explanations. *NodeSet* and *InferenceStep* are the main types of proof elements. A *NodeSet* represents a step in a proof whose conclusion is justified by any of a set of inference steps associated with the *NodeSet*. Each node set has one URI. PML adopts the term "node set" because each instance of *NodeSet* can be viewed as a set of nodes gathered from one or more proof trees having the same 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.

Provenance elements are PML elements describing the origin of proof elements. *ProvenanceElement* is a superclass of the PML concepts at the provenance level(such as source, language, and inference rule). A taxonomy of provenance elements is shown in Figure 1. The *ProvenanceElement* attributes are described below:

- The `URI` of a provenance element is the unique identifier of the provenance element. Every provenance element has one well-formed `URI`, which is an instance of the primitive type *anyURI*.
- The `URL` of a provenance element describes an URL used to browse an element's web document. For instance, if the provenance element is an organization named the New York Times, then the http://www.nytimes.com URL can be used to access a web document about the organization. In this case the URL points to the organization's web site. A provenance element can have zero or one URLs.

[11] Terms defined in this Section are typed in **bold**.

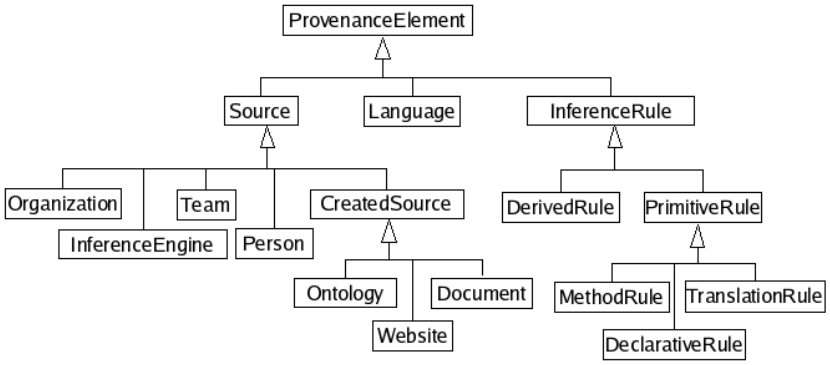

Figure 1: Top of the PML taxonomy of provenance elements.

- The `Name` of a provenance element describes a short name (or "nickname") for the element within the IWBase. Every provenance element has one name, and the name is an instance of the primitive type *string*.
- The `Submitter` of a provenance element represents the team of people[12] responsible for the registration of the provenance element in IWBase. Every provenance element has one submitter, and the submitter is an instance of the type *Team*, which is a subclass of *Source* (see Figure 1).
- The `DateTimeInitialSubmission` of a provenance element is the date when the provenance element was first registered in IWBase. Every provenance element has one `DateTimeInitialSubmission`, and that is an instance of the primitive type *dateTime*.
- The `DateTimeLastSubmission` of a provenance element is the last date when the provenance element was registered in IWBase. Every provenance element has one `DateTimeLastSubmission`, and that is an instance of the primitive type *dateTime*.
- The `EnglishDescription` of a provenance element is a description in English of the provenance element. A provenance element can have zero or one descriptions in English, and `EnglishDescription` is an instance of the primitive type *string*. The description in English is intended to be used by tools to present provenance elements to human agents. For example, the description in English of the *Modus Ponens* inference rule may be a better presentation and more informative for most users browsing a PML document than the presentation of the formal logical specification of the rule.

[12] Of course a team can consist of any number of people including zero known members to date or a single member.

Provenance elements are formally specified in OWL[13] and are briefly described below. Further descriptions of provenance elements including the description of some relationships among provenance elements are available in [21].

A *Source* represents an entity which is the source of the original data. A source can be either an *Organization*, an *InferenceEngine*, a *Team*, a *Person* or a *CreatedSource*. *CreatedSources* can be *Ontologies*, *Websites* or, generally, *Documents*. *InferenceEngine*is a particularly important source representing an engine that is able to produce a justification for a given conclusion. Note that the use of the term "inference engine" in this paper is not limited to engines with reasoning capabilities. For example, a search engine retrieving information may serve as an inference engine and it may provide a justification of its answer by a direct assertion inference step. Similarly extraction modules may be viewed as inference engines and, in fact, we have registered a number of extraction modules from the UIMA extraction toolkit[5] so that extracted answers may be explained in Inference Web using PML.

The *Language* provenance element is used to encode a language used to write conclusions of node sets. Any language can be registered in IWBase including formal logical languages, such as KIF, and natural languages, such as English, and representation languages such as DAML+OIL and OWL.

An *InferenceRule* is a specialization of *ProvenanceElement* used for describing rules applied to premises deriving node set conclusions. An *InferenceRule* can be either a *PrimitiveRule* or a *DerivedRule*. A *PrimitiveRule* is a type of an inference rule that is implemented by one or more inference engines. A given rule R_1 may only be called primitive when it becomes associated with one or more inference engines. Thus, assuming that R_1 is implemented by an inference engine E_1, the inference engine may declare R_1 to be a primitive rule. The notion that R_1 is a primitive rule for one specific engine is relevant since R_1 may also be derived from R_2, which is a primitive rule for another engine E_2. In this case, R_1 may be registered once as a primitive rule and zero or more times as a derived rule, depending on how many combinations of rules are used to derive R_1. A *DerivedRule* is an *InferenceRule* specified from a PML node set schema with the restriction that each node set schema must have one and only one inference step.

3.2 IWBase Node

IWBase Node critically depends on the IWBase registry, IWBase registrar and IWBase configuration file so we will define these first.

An **IWBase registry** is a repository of provenance elements. In order to support interoperability when sharing provenance metadata among Inference Web tools and between Inference Web tools and other Semantic Web tools in general, elements in the registry are stored as PML files. Thus, with the use of Semantic Web tools, one can retrieve, parse, and use proof-related metadata. For querying and maintaining large quantities of metadata, the parsing

[13]The PML ontology version 0.9.0 is available at http://iw.stanford.edu/2004/07/iw.owl

of PML files has shown to be too expensive. Therefore, to increase scalability, provenance elements are also stored in a database.

An **IWBase registrar** is a collection of applications used for maintaining a registry. From a human user point of view, the registrar is an interactive application where the user can add, update and browse the registry contents. From a software agent point of view, the registrar is a collection of services for querying and updating the registry. The registrar is responsible for keeping the synchronization between the registry repository of OWL files and the registry database.

An **IWBase configuration file** contains configuration and deployment information for IWBase node applications. Information is defined as parameters in name/value pairs and loaded at application startup time. Global parameters are shared among all applications deployed on the node. Each IWBase application deployed on the node has its own collection of parameters, which can override global parameters.

An **IWBase node** is a named collection of computing resources available in a machine. One machine may have multiple nodes. In concrete terms, a node is characterized by having the following resources:

- a registry;
- a registrar;
- access to read and write in a filesystem in order to store the registry, the registrar and any optional IWBase applications;
- access to a web server platform[14] supporting servlets/JSP and SOAP services; serving the node filesystem; and providing facilities for user authentication and authorization;
- permission to change the configuration of the web server;
- access to a database in a relational database management system (DBMS)[15] including the following: permission to update the schema of the database; permission to add and update tuples in the database;
- a collection of optional applications based on IWBase elements;
- a configuration file used to set parameters for the registry, some parameters for the web server and DBMS, and parameters for IWBase applications installed on the node.

One machine is allowed to have multiple nodes. Nodes is a common machine may share filesystems, web servers and DBMSs. However, to accommodate its registry, each node needs to have a non-shared directory in a shared filesystem for the OWL files and a non-shared database.

[14] IWBase currently has support for Tomcat and Axis.
[15] IWBase currently has support for MySQL.

3.3 IWBase Applications

An **IWBase application** is an application installed under an IWBase node web server, using the IWBase configuration file, and providing some functionality that depends on PML elements of the node. The registrar is an IWBase application.

Engines can use any strategy for creating PML documents including the use of **PML generation services (PGSs)**, the use of the PML API, or even the implementation of their own code to generate PML documents. IWBase node's PGSs, provide support for querying the registry through a combination of queries to database(s) and OWL files. In addition to the use of indexes for querying the registry, PGSs provide a mechanism for selecting and associating provenance elements with proof elements. Even if none of those services are critical for applications, users may still choose to use PGSs since they provide an uniform way of generating PML documents thus supporting consistent usage if the PML specification evolves.

We envision the addition of many IWBase applications to IWBase nodes. For instance, at the moment we are developing an **AnswerTrustComputation** service to compute trust values for answers. These services rely on a combination of trust relations, some of which are already implemented in IWBase. These services are supported by a trust network of users and node elements.

4 Architecture

We use the term **IWBase** to refer to an hypothetical "entire" collection of IWBase nodes, whether nodes are interconnected or not and whether nodes are available for use or not. In this section we discuss the different types of nodes and node inter-communications in IWBase. Before discussing node configurations, we may need to briefly discuss the roles of users on IWBase.

4.1 Users

IWBase users can play one or more of the following roles when interacting with an IWBase node.

A **node visitor** is some "web user", human or agent, using the node either to inspect the registry or to interact with the registrar with the purpose of browsing node elements. Node visitors do not need to be registered with the node and do not have ownership of node elements. In public nodes, node visitors may browse the entire content of the registry.

A **node member** is some user, human or agent, using the registrar for maintaining node elements. Node members are registered with the node administrator in order to have access to the registrar for adding and updating node elements. Node members are also registered as members of PML Teams (a PML provenance element).

A **node administrator** is a human user responsible for the following tasks: maintaining and tuning the IWBase node web server and DBMS; creating node

members; adding and removing IWBase applications; setting IWBase application parameters in the node configuration file.

4.2 Nodes and Node Associations

An IWBase node is either a **core node** or a **domain node**. Some PML elements such as *InferenceEngine*, *InferenceRule* and *Language* (see Figure 1) 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 registry. Alternatively, a user can have nodes owning entries for relevant ontologies associated with domain nodes where the user is a node member as described below in this section. We expect these decisions to evolve with usage.

Private and public nodes. IWBase relies on typical internet and web security apparatus, i.e., firewalls, for protecting access to node metadata. A public node implements no restrictions for web users or agents to access the node's contents. If any kind of restriction is enforced, the node is private. Other than the core node that is always public, nodes can be either public or private.

The IWBase architecture also specifies some services supporting the collaboration between the nodes described as follows.

Domain node–domain node associations. Inter-domain communications are accomplished by configuring **association links** between nodes. An association link from node B to node A means that services provided by node B are based on meta-information coming from both B and A. To illustrate the usefulness of association links, assume a link from *vehicles* to *cars*. Thus, automated services for browsing the registry in *vehicles* would list elements registered both in *vehicles* and *cars*. In this case, *vehicles* could leverage from a relevant list of sources about cars available in *cars*, for example. Moreover, *vehicles* could set associations links to other domain nodes such as *bicycles* increasing the amount of meta-information available for using on proofs. Association links aim to increase elements' re-usability, eliminate data duplication among domain nodes, and to prevent inconsistency.

Mirrored core registry. Information in the core node registry is required for running services for a domain node such as proof generation services. In

some situations, however, node services may have no access to the core node, i.e., when a domain node has no access to the internet. In other situations, although a domain node may have access to the internet, the cost of getting information through the web can be too expensive and having a local copy of the core registry reduces the internet traffic. If any of these situations occurs, domain node administrators have the option of copying the content of the core registry to a local filesystem and to set the domain node configuration file to point to the local copy of the core registry, the called *mirrored core registry*. The currency of the mirrored core registry depends on how often the domain node administrator updates the copy. Core node elements are only registered and maintained by the core node registrar and elements in the mirrored core registry are not supposed to be updated.

Core node–domain node associations. Every domain node must have an association with either the core node or a mirrored core registry. The association is required since some classes of provenance elements supporting services in the domain node, such as *Language* and *Inference Engine*, are available on the core registry only. Using the configuration file, a domain node administrator can choose which core registry the node should use.

5 Metadata in Action: An Example

The need for provenance elements is compelling when any kind of meta-cognition activity is performed on proofs describing answer derivation (or answer composition, answer extraction, etc.). In this section we will illustrate how IWBase elements support the following tasks: understanding proofs, generating explanations from proofs, and checking the correctness of answer derivation processes (by checking proofs). The example in this section is used to show IWBase support for checking proofs and for generating explanations from proofs. Some details on how these tasks are performed are not exposed in the paper.

5.1 Understanding Proofs

Raw proof traces are often used by reasoner developers to debug their engines. More sophisticated uses of proofs involve the exchange of proofs among reasoners in a hybrid reasoning environment [23] or the use of proofs as an input for generating human understandable explanations [16]. In these examples and in many others, providing information aimed at providing help understanding proofs means accessing information often unavailable from reasoners. The IWBrowser rendering of a proof in Figure 2 has several examples of uses of the IWBase elements. There, each statement is annotated with the name of the engine responsible for adding the statement to the proof, i.e., `JTP - Java Theorem Prover` [6]. Further, if a statement is derived, the engine represents the software responsible for the statement derivation in addition to representing the source responsible for creating of the node set. For derived statements, the proof is annotated with the inference rule applied by the engine to the statement's

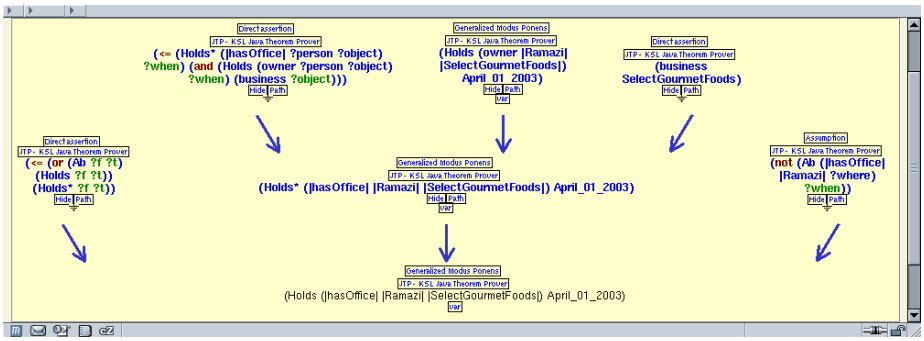

Figure 2: A typical proof.

antecedents allowing the derivation of the consequent statement. For example, in Figure 2, the `generalized modus ponens` rule was used to conclude[16] that Ramazi had an office at SelectGourmetFood on April 1, 2003, which is represented in the proof as"`(Holds (|hasOffice| |Ramazi| |SelectGourmetFoods|) April_01_2003)`".

Proof annotations mentioned above are just few examples of IWBase elements associated with the proof in Figure 2. As presented in the IWBrowser, node elements already increase the understanding of proofs for reasoner developers and even for other users not involved on the answer generation process, e.g., a user trying to understand how the answer was produced. In Figure 2, each PML element in the proof is rendered as a hot link leading to further information about proof elements. For example, Figure 4 presented later in this section shows the outcome of selecting the hot link for the `generalized modus ponens` rule. There, we can see information useful for increasing user understanding of the meaning of the rule such as a description in English of the rule and a reference to a textbook.

5.2 Generating Explanations from Proofs

Ordinary users on the web are not expected to browse proofs in order to understand answers. Instead, users should have access to a set of explanations, each providing an alternative way for users to understand the answer. If a user does not understand an answer, a user may explore one of these explanations in a summary mode or in a detailed mode.

Abstracting proof traces into explanations is one way of explaining answers that depend on IWBase elements. For example, Figure 3 shows an explanation generated from the proof in Figure 2. The explanation consists of a simple application of an explanation tactic for rewriting the proof in an abstracted way and from there to render the proof in plain English. The explanation

[16]This used three antecedents representing a provisional conclusion about Ramazi's office and the lack of atypicality with respect to Ramazi's office.

tactic is the key element for explaining the proof and it is represented and stored as a *derived rule* in an IWBase domain node. The matching of the tactic against the proof is performed by the PML Abstractor service. IWBrowser and IWExplainer are examples of IWBase applications using the PML Abstractor service.

Someone who owns a business typically has an office at that business. Because Ramazi owned SelectGourmetFood on April 1, 2003 and SelectGourmetFood is a business, it follows that Ramazi had an office at SelectGourmetFood on April 1, 2003.

Figure 3: An explanation derived from the proof in Figure 2.

Summarizing the usage of information sources and assumptions during answer derivation is another alternative for explaining proofs that is based on IWBase elements. Our premise is that users will understand and trust answers more if they know more about the information sources for answers. Thus, the summarization of sources and assumptions associated with a set of PML node sets along with the possibility of further investigating the IWBase registration of metadata is a simple and useful way of increasing user understanding and trust of answers. Users also inherit the ability to search for sources associated with proofs as a consequence of using PML, which supports full tracking of provenance information [22].

5.3 Checking Proofs

Our experience with instrumenting engines to dump proofs in PML has demonstrated that the identification of rules implemented inside engines is one of the challenges in generating useful proofs. Rules are typically implemented as procedural methods rather than being derived from declarative specifications. Further, methods implementing inference rules often have a number of optimization strategies complicating the task of figuring out which rules are implemented in an engine by inspecting the engine's code. Thus, the characterization of methods as inference rules is not always trivial task.

IWBase facilitates the work of figuring out which rules are implemented in an engine since the core node has a number of registered primitive rules and some of them are formally specified in Inference Meta Language (InferenceML)[17] [20]. By checking steps of a proof against rule specifications in the core node, a tool may be able to identify rules implemented in the engine producing the proof.

For example, the browsing of the generalized modus ponens rule used in the proof in Figure 2 is shown in Figure 4. There, the `rule specification` attribute is a sentence in InferenceML. This specification enables tools using the PML Checker API to verify if the the proof steps using the rule represent a correct application of the rules. This verification is possible due to the following

[17]InferenceML was formally known as Proof Protocol for Deductive Reasoning (PPDR).

facts: (1) the proof step is explicitly associated with a PML primitive rule; and (2) the node element for the rule has a formal specification in InferenceML.

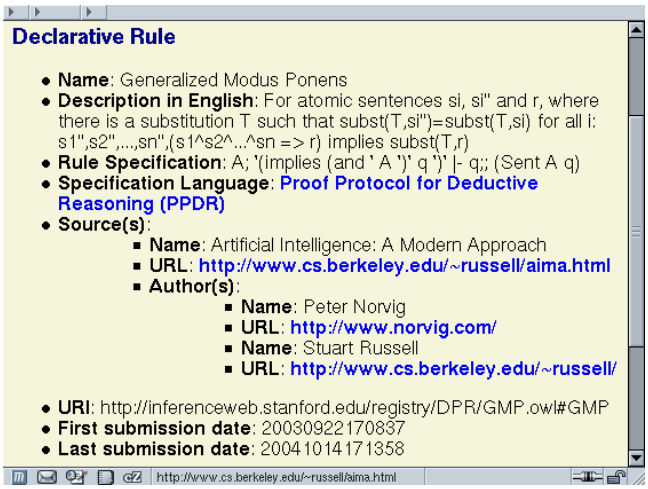

Figure 4: The IWBase element for the *Generalized Modus Ponens* inference rule.

There are, of course, many benefits to proof checking. For instance, one can confirm the correctness of engine rule applications by running the PML checker on PML proofs; and one can increase his/her level of trust in an answer if steps in proofs for answers can be checked.

6 Discussion and Status

In this section, we will return to our goals and summarize our contributions with respect to those goals. Our approach is distributed in that it supports a number of IWBase nodes. The IWBase core node contains information common to applications using proof-related metadata. The registry for the core node could be either local or remote. IWBase may have multiple domain nodes that contain information specific to any particular application. There is no limit to the number of domain nodes and they can be physically located on any machine. IWBase administration is also distributed since node administrators may make the decisions about who can make updates by registering other members and administrators.

The scalability of our approach is supported from a few perspectives. We have utilized a database backend for IWBase so that we may leverage the scalability and efficiency of DBMSs for querying and updating databases. We also do not require registration of all meta data in our repositories; as long as meta data is accessible, that is all that is required. For example, in our joint effort with IBM to explain text analytics, we found the sheer volume of information to be

enough that a special purpose repository made sense that was tuned to the needs of text extraction. This database is accessed when PML proofs are generated so that provenance information is available for explanations on demand but is not pro-actively registered in IWBase for all possible content. Additionally, when we designed the Proof Markup Language, we chose to represent it in OWL and stayed within the description logic (DL) portion of the language. Thus, when we rely on reasoning with the OWL representations, we can claim the same efficiency of reasoning that OWL-DL can claim. OWL-DL was designed specifically to be a language for which efficient reasoners may be implemented.

The IWBase infrastructure includes tools to help minimize maintenance effort. This is intentional since we understand that the quality of the explanations will be related to the quality and recency of the provenance metadata. We include a registration service that supports automatic registration of sources. We also have included many reasoners and inferences in the core node in the hopes that registration many times will be identifying which items to reuse rather than providing original specifications of new inferences. We also provide checking services to identify if PML dumps are consistent in their usage with the inference rule registrations that they claim to contain.

The IWBase solution addresses privacy using internet and web security apparatus as described in Section 4.2. Nodes can be private or public and access to update the IWBase requires registration. While this solution has been adequate to date, we do anticipate augmenting our privacy solution in order to address more of the concerns of the intelligence community.

In terms of the provenance requirements we identified in Section 2, our current solution includes explicit information for all but degree of completeness.

The primary contributions of our work with respect to provenance metadata infrastructure for explanations revolves around specifications, infrastructure, and tools. We have gathered requirements from a broad diversity of users from many perspectives - backgrounds, usage, sophistication, etc. We have collected the requirements in one place and have provided a description of a solution architecture that meets these requirements. The solution is not only described in papers but is also implemented in a freely available, open-source implementation distributed through the Inference Web website[18] and soon through SemWebCentral[19]. This implementation can serve as infrastructure for other's explanation and trust efforts, or if required, it could serve as an operational specification and starting point for someone to generated their own customized solution. Our approach is also modular and stresses interoperability so one could, for example, use our IWBase core node, some existing domain nodes, some new internal domain nodes, our proof generation services and an internal interface/browser interface.

IWBase, as part of the Inference Web, is gathering users and is currently integrated with a large text analytics tool suite - UIMA. It has been integrated with the JTP hybrid reasoning engine and is being used by a number of projects

[18] http://iw.stanford.edu
[19] http://www.semwebcentral.org/

that require explanations of hybrid reasoners - in particular in the CALO project of the DARPA PAL program. As a result, it is explaining results from applications such as ISI's Ariadne system [13], where information was scraped from, how query management is done in terms of breaking up questions into smaller pieces which together can answer a larger question, and it is also being integrated with the University of Texas's KM system [3] and the SRI's SPARK system [18]. It is also being used by the KANI project within ARDA's NIMD program which means it needs to range from explaining text analytics to temporal reasoning to context reasoning with hypotheticals.

Work is proceeding to both broaden and deepen IWBase capabilities. Since explanations are improved when metadata information is more current and more complete, we continue to improve the registration services. Currently IWBase has services providing automatic support for maintaining node elements. There are still open issues concerning automated inclusion of attributes to sources. For instance, an agent needs to select and add authors to a registered document but it may have difficulty since it is required to select the appropriate metadata corresponding to the authors.

We are also working to support more flexible security strategies for accessing domain nodes behind firewalls. For instance, some organizations may require their domain nodes to be behind a firewall but they still want selected external users to access parts of their registries.

7 Related Work

When we first specified our metadata registry requirements, we did a literature search and did not find any existing solution that we believed could meet our requirements (Section 2). Our initial approach was to extend Stanford's WebBase [11] to register proof-related metadata. We learned from our own work and that of Heery [10] that accessing metadata can be quite complex. IWBase requires elements to be grouped and indexed by their types. This led us away from an approach such as WebBase that indexes through a uniform attribute. We decided to design an solution for registering metadata that was based on a combination of a web repository and of a database system.

An agent's ability to share metadata with other agents is a major reason for IWBase to store proof-related metadata in the registry as OWL files. In fact, the lack of a standard format for metadata was a concern in previous efforts to build metadata registries [1].

The Dublin Core Metadata Initiative (DCMI)[20] is probably the best-known collection of bibliographic metadata. In the PML provenance element taxonomy, *Document* is the element representing Dublin Core metadata. *Document* is the only non-extensible PML element and it is expected to conform with DCMI. Furthermore, DCMI does have a registry[21] for storing and maintaining DCMI metadata and the registry shares a number of functional requirements as

[20] http://dublincore.org/
[21] http://dublincore.org/dcregistry/

described in [9] with IWBase. For instance, the IWBase and DCMI Registry are expected to store reliable trusted information, to provide authoritative definitions, and to manage the evolution of their elements. One difference between IWBase and DCMI Registry is that IWBase includes instances of metadata types.

There are many registries of domain-specific metadata available on the web. For example, the DoD Metadata Registry and Clearinghouse[22] is a domain-specific registry of software related metadata for use by the U.S. Department of Defense. Other government agencies throughout the world have their own registries. In the future, we expect applications that use government registries such as this and/or maintainers of this registry to use information from IWBase.

8 Summary

This paper presents IWBase as a distributed, scalable, maintainable provenance metadata infrastructure with a privacy strategy. Provenance metadata in IWBase may help users to understand and trust answers from applications and services on the web. We gathered a set of requirements for provenance metadata for web applications and designed and implemented a solution that addressed these requirements. Specific contributions from this effort include:

- A distributed registry of provenance metadata with a taxonomy of provenance elements specified in an interoperable proof interlingua (PML) based on the Ontology Web Language.
- A registrar that provides support for the registration and maintenance of provenance metadata. For humans, the registrar provides its functionalities through an interactive interface. For computer agents, it provides automated services.
- Services for generating proofs in PML, checking those proofs against inference specifications, abstracting those proofs into understandable explanations, and computing trust values for answers supported by those proofs and explanations. We are able to provide these services because of the infrastructure for provenance metadata.

IWBase is being used to support explanations of question answering applications that range from text analytics and aggregation to hybrid theorem proving in academic, industrial, and government settings.

9 Acknowledgments

The authors would like to thank the following programs that funded portions of this work: DARPA's DAML and PAL programs and ARDA's NIMD program.

[22] http://diides.ncr.disa.mil/mdregHomePage/mdregHome.portal

Additionally, we gratefully acknowledge valuable comments from Jose-Luis Ambite, Priyendra Deshwal, J William Murdock and Ilya Zaihrayeu.

References

[1] C. Blanchi and J. Petrone. Distributed Interoperable Metadata Registry. *D-Lib Magazine*, 7(12), December 2001.

[2] P. Buneman, S. Khanna, and W.-C. Tan. Why and Where: A Characterization of Data Provenance. In *Proceedings of 8th International Conference on Database Theory*, pages 316–330, January 2001.

[3] P. Clark and B. Porter. *The Knowledge Machine Users Manual*. University of Texas, Austin, Austin, TX, 2004.

[4] A. Das, W. Wu, and D. L. McGuinness. Industrial Strength Ontology Management. In I. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, *The Emerging Semantic Web*. IOS Press, 2002.

[5] D. Ferrucci and A. Lally. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Journal of Natural Language Engineering*, June 2004. To appear.

[6] R. Fikes, J. Jenkins, and G. Frank. JTP: A System Architecture and Component Library for Hybrid Reasoning. Technical Report KSL-03-01, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA, 2003.

[7] F. Giunchiglia and P. Shvaiko. Semantic matching. Technical Report Vol. 71, CEUR - WS, 2003.

[8] R. V. Guha, R. McCool, and E. Miller. Semantic search. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, pages 700–709, Budapest, Hungary, May 2003. ACM Press.

[9] R. Heery. Draft DCMI Open Metadata Registry Functional Requirements, October 2001. http://dublincore.org/groups/registry/fun_req_ph1-20011031.shtml as accessed on 6 Nov 2004.

[10] R. Heery and H. Wagner. A Metadata Registry for the Semantic Web. *D-Lib Magazine*, 8(5), May 2002.

[11] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase : A repository of Web pages. *Computer Networks*, 33(1-6):277–293, May 2000.

[12] E. F. Kendall, M. E. Dutra, and D. L. McGuinness. Towards A Commercial Ontology Development Environment. In *International Semantic Web Conference Late Breaking Topics*, Sadinia, Italy, June 9-12 2002.

[13] C. A. Knoblock, S. Minton, J.-L. Ambite, N. Ashish, I. Muslea, A. G. Philpot, and S. Tejada. The ariadne approach to web-based information integration. *International Journal of Cooperative Information Systems (IJCIS), Special Issue on Intelligent Information Agents: Theory and Applications*, 10(1/2):145–169, 2001.

[14] D. L. McGuinness and P. Patel-Schneider. From Description Logic Provers to Knowledge Representation Systems. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 265–281. Cambridge University Press, 2003.

[15] D. L. McGuinness and P. Pinheiro da Silva. Explaining Answers from the Semantic Web. *Journal of Web Semantics*, 1(4):397–413, October 2004.

[16] D. L. McGuinness and P. Pinheiro da Silva. Trusting Answers on the Web. In M. T. Maybury, editor, *New Directions in Question Answering*, chapter 21. AAAI/MIT Press, October 2004.

[17] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), December 9 2003. Proposed Recommendation.

[18] D. Morley and K. Myers. The SPARK Agent Framework. In *Proc. of the Third Int. Joint Conf. on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, 2004.

[19] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL Ontologies. Submitted for publication.

[20] P. Pinheiro da Silva, P. Hayes, D. L. McGuinness, and R. Fikes. PPDR: A Proof Protocol for Deductive Reasoning. Technical Report KSL-04-04, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA, March 2004.

[21] P. Pinheiro da Silva, D. L. McGuinness, and R. Fikes. A Proof Markup Language for Semantic Web Services. *Information Systems*, 2004. (to appear).

[22] P. Pinheiro da Silva, D. L. McGuinness, and R. McCool. Knowledge Provenance Infrastructure. *IEEE Data Engineering Bulletin*, 25(2):179–227, December 2003.

[23] J. H. Siekmann, C. Benzmüller, V. Brezhnev, L. Cheikhrouhou, A. Fiedler, A. Franke, H. Horacek, M. Kohlhase, A. Meier, E. Melis, M. Moschner, I. Normann, M. Pollet, V. Sorge, C. Ullrich, C.-P. Wirth, and J. Zimmer. Proof Development with OMEGA. In *Proceedings of 18th International Conference on Automated Deduction*, volume 2392 of *LNCS*, pages 144–149, Copenhagen, Denmark, July 2002. Springer.