Representing Proofs in the Semantic Web
– Draft –

Christoph Wernhard
Persist AG
wernhard@persistag.com

June 7, 2001

1 Introduction

One of the seven layers in the Semantic Web architecture suggested by Tim Berners-Lee is the proof layer [1, 2]. His idea is that a client can submit to a server a proof why it should have access rights to the requested information. The server then validates the proof. Its response again contains a proof that he returns in deed what was asked for.

In general proofs can capture epistemic and intensional information. Meta-information about a resource, such as authorship, can be viewed as a proof of its content. If the resource incorporates information from another resource, the proof of the referred resource becomes a subproof of the first resource.

There are other uses of proofs in the Semantic Web: With a graphical or natural language based proof presentation it is possible for humans to retrace the derivation of answers. Proofs can be used as templates for solving problems similar to the proven one.

So it would be desirable to have a "standard" format for exchanging proofs in the Semantic Web. In the following we make first steps towards this goal. We outline a RDF [3] representation of proofs within the Semantic Web. The way proofs are represented follows the spirit of RDF. Both simple notions of proof and proofs created by actual automated reasoning systems such as Otter [4] are considered in a unified way.

2 Basic RDF Proof Representation

We orient ourselves at the so-called *ILF Standard Proof Format* (abbreviated here as *ILF-SPF*) [5, 6] [1], which follows a similar representation paradigm

[1] ILF-SPF was developed in the mid-nineties within the *Deduction* activity of Deutsche Forschungsgemeinschaft at Humboldt University, Berlin. Its objectives are the conversion

as RDF. ILF-SPF actually uses Subject-Property-Object triples in a Prolog serialization for proof representation.

2.1 The `proof` class

An object of the `proof` class represents the proof of a particular sentence. By the `reference` property proof objects can be combined to complex proofs.

The `proof` class is domain of the following properties:

- `content` — A single-valued property that must be supplied. The content, i.e. the sentence proved. The value can be a literal, representing a sentence opaque to RDF, or, if we have an RDF representation of sentences, a resource identifying a sentence.

- `rule` — A single-valued property that must be supplied. The value can be a literal or a resource. It identifies the rule (of some calculus) by which the content is proved.

  The proof representation format makes no presuppositions about the calculus used. The value of `rule` might for example be just some literal of a fictive calculus or a resource referring to some specific calculus published in the Semantic Web.

- `reference` — A multi-valued property. The values are proofs that are used to justify this proof according to the `rule`.

  The number of references depends on the `rule` property.

  Issue: It might be useful that the references can have special parametric roles with respect to the rule. This can not be expressed simply by a multi-valued reference property.

3 Structuring Properties

The three basic properties outlined above seem sufficient to characterize the semantics of proofs. In this section we look at further properties found in ILF-SPF, that seem to express information about concrete structuring of proofs. Perhaps such properties could form a second layer of a RDF proof format, that is useful e.g. for proof presentation. It should then be possible that facts with these properties can always be stripped off a proof without changing its meaning. On the other hand structuring properties can always

of proofs output by different automated reasoning systems such as Otter (a resolution prover), Setheo (a model elimination prover) and Discount (an equational prover) to human readable mathematical text-book style proofs, to combine proofs for subproblems by different systems and to integrate automated systems with an interactive one. On the ILF-SPF proof representation transformations such as mapping between calculi, proof restructurings and the elimination of trivial subproofs can be performed.

be added, provided they meet certain integrity constraints with respect to the basic proof.

- `predecessor` — A single-valued optional property - the preceding proof in a total ordering of proofs.

  In ILF-SPF proof is a sequence of so-called proof lines. The last line corresponds to the overall goal proven.

  Issue: If the proof is considered as a graph, only a partial ordering determined by the *reference* properties instead of the total ordering is used and the predecessor property is not needed.

  Issue: In ILF-SPF *predecessor* is multi-valued (hence called *predecessors*). We don't know if this is actually used by ILF applications.

- `subproof` — A single-valued optional property. In ILF-SPF the value is a *proof object* which is contrasted to *proof line* objects. (What we call *proof* here corresponds to a *proof line* object).

  Issue: Like the total ordering by `predecessor` this seems a further structuring property. Can this be expressed e.g. by having a proof object (i.e. one of the references) as value?

  Issue: The structuring into totally ordered proof lines and subproofs must be in accordance with the `reference` property: References must be in the scope of *usable* lines. A referenced line is usable from the subsequent lines of the same proof and from lines in the subproofs (transitively) of these lines.

- `status` — Must be supplied. Value is *proved* or *assumption*.

  Issue: Is *assumption* needed? Is it a structuring means like `predecessor` and `subproof`?

  Issue: Should *axiom* be a status for terminal nodes or should they have just *proved*? Perhaps the fact that it has no references suffices to identify an axiom. Also the `rule` property might be used to identify axioms.

  Issue: ILF-SPF also has a status value *unproved* to represent incomplete proofs.

4 Further Issues on Properties of Proofs

Properties of Complex Proofs How are properties expressed, that affect complex proofs (i.e. proof objects related through the `reference` property) as a whole? Examples for such properties could be a calculus or system by which a complex proof was obtained or certain constraints which are ensured to be satisfied by the complex proof. Perhaps it suffices if they

can be asserted explicitly for a top level proof object? Their scope could then be defined as following the `reference` property until they are overridden by another explicit statement. Or an extra object representing the complex proof as a whole can be used.

Distinguishing Axioms While the terminal nodes of the proof graph can be seen as axioms, most theorem provers do some preprocessing on the input axioms before actually proving (such as clausal form transformation of first order axioms). Special rules might be used for initial proof steps that lead e.g. from an original axiom to one of its clauses.

Naming Sentences It might be useful to name certain sentences, for example axioms. An additional property of proof objects, `name`, could be used for that purpose.

5 General Issues

Representing Plans A proof is very similar to a plan in the sense of AI planning systems, i.e. a partial ordering of actions. The proof representation should perhaps also allow to represent plans.

Mapping to Lambda Calculus Based Proof Representations There seems a lot of theoretical work on such proof representations. Some interactive theorem proving systems use a notion of proof based on the lambda calculus.

Work Out Proof Transformations The work done for ILF can be helpful there.

Write Translators for Provers such as Otter and Model-Elimination Provers The work done for ILF can be helpful there.

References

[1] T. Berners-Lee. Semantic Web — XML 2000. Presentation Slides. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

[2] T. Berners-Lee. Semantic Web Road map, 1988. http://www.w3.org/DesignIssues/Semantic.html

[3] O. Lassila, R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999. http://www.w3.org/TR/REC-rdf-syntax/

[4] W. McCune. OTTER 3.0 Reference Manual and Guide, Argonne National Laboratory/IL, USA, 1994. http://www.mcs.anl.gov/home/mccune/ar/otter/index.html

[5] B. I. Dahn et al.: ILF Server Manual. http://www-irm.mathematik.hu-berlin.de/ ilf-serv/

[6] B. I. Dahn and A. Wolf. Journal for Information Processing and Cybernetics (EIK), (5–6): pp. 261–276, 1994. http://www-irm.mathematik.hu-berlin.de/ ilf/papers/block.ps