# Combinable Proof Fragments for the Web

Paulo Pinheiro da Silva  Deborah L. McGuinness
Richard Fikes

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

**Abstract**

Web applications must deal with a rapidly evolving, distributed, heterogeneous environment of information and reasoning services. Many applications obtain information from multiple sources and thus need to determine how, why, and when to use answers. Currently few if any web-based reasoning services make explanations available and, if they do, the supporting proofs of these explanations are rarely portable across applications. Thus, proofs, if available, are difficult to understand and more difficult to combine. This paper describes the properties of Inference Web (IW) portable proof fragments. These proof fragments can be combined and thus can be used to specify distributed proofs for web-based reasoning tasks. Moreover, the proof fragment can be used for merging proofs generated by distinct applications. We introduce the portable proof fragment specification used to enable Inference Web applications by example. Our examples are chosen from real world configuration-driven examples.

## 1 Introduction

Explanations are expected, required results when performing automated reasoning tasks [1, 3, 14, 8]. In a centralized environment such as a computer, the dumping of some kind of proof is a standard feature of most inference engines these days. In a distributed and heterogeneous environment such as the web, however, explanations are unavailable since their supporting proofs need to be *distributed*, *sharable*, *portable* and *combinable* simultaneously. Proofs are distributed if proof fragments can be stored in distinct websites. Proofs are sharable if they can be represented in a notation with a semantic precise enough to make them machine understandable. Proofs are portable if they can be independent of both the inference engines and inference engine environments where they are generated. Proofs are combinable if they can be merged through their common elements such as formulas (it is desirable that original proofs can be identified in combined proofs).

Proofs for the web may need to be distributed since reasoners can be embedded in web services that are inherently distributed. Proofs for the web may need to be sharable since distinct services may be based on different reasoners (or single services may be based on many reasoners). Proofs for the web may need to be portable since inference engines may be unavailable at the time explanations are required. Proofs for the web may also need to be combinable if single explanations are expected for both single services or combined services. Thus, the lack of a specification of proofs that can be distributed, sharable, portable and combinable poses challenges for explaining web-based automated reasoning.

The Inference Web (IW) is an implemented, distributed explanation solution composed of specialized data and tools for generating, browsing, maintaining, annotating, and combining proofs used for explaining reasoning and retrieval tasks performed in the web [10]. This paper describes the properties of IW proof fragments that can be used by distinct agents to build and/or combine distributed, portable proofs. We present sharable proof fragments represented in W3C's DAML+OIL [4].

The rest of this paper is organized as follows. Section 2 presents an example pedagogical query. Section 3 shows how an IW proof can explain an answer for a deductive query. The section also explains why IW proofs are inherently distributed and portable. Section 4 describes related work and Section 5 presents some conclusions and future work.

## 2 A Proof Example

Any web application that queries a knowledge bases (or data bases) is a potential user of Inference Web proofs. In this paper we use the Wine Agent[1] application to introduce the IW proof.

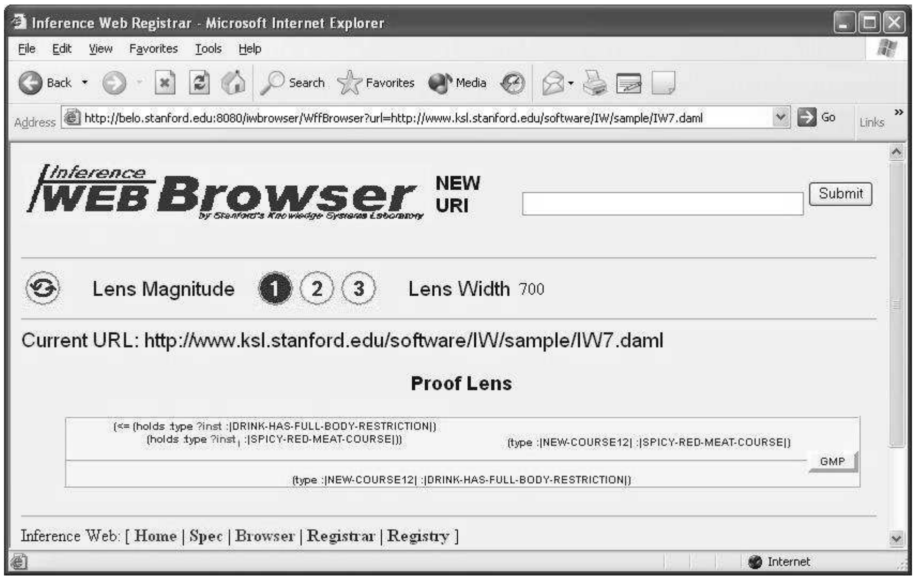
The wine agent is a web application that provides information about wines such as suggested wine and food pairings, descriptions of wines that match course descriptions, wine availability on selected wine web sites, etc. A user interacting with the wine agent is presented with an interface that allows them to choose either a particular food (such as oysters) or a description of a type of meal course (such as shellfish or pasta with spicy red sauce). The program then forms a syntactically correct query and submits it to a reasoner (in this case Stanford's JTP [6]. The wine agent then uses the wines ontology[2] with JTP to return a description of the wine that matches the meal as well as any individual wines that match that are in the knowledge base. It also provides the user with the option of finding the particular wines on a wine web site and provides the option of using the description of the wine as a query to the web site to provide other individual wine options. Since the reasoner used in this example can generate IW proofs, the wine agent uses these proofs in order to provide at least one explanation for any answer produced. IW proofs can also be

[1]http://ksl.stanford.edu/people/dlm/webont/wineAgent/

[2]http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml

used to support explanations for queries for which no answer was returned when working with reasoners capable of dumping proofs for why a query was over-constrained (incoherent) because of the existence of counter-examples or proofs of the negation of the query. We will limit this paper to the discussion of positive explanations however and leave "why-not" and counter-example explanations for a future paper.

As an example, lets say that the user choose a spicy red meat course and then is interested in knowing why the suggested wine description includes the notion of a full bodied wine. As presented in Figure 1, an explanation in the terms of the wine ontology is that DRINK-HAS-FULL-BODY-RESTRICTION as a result of the application of modus ponens on the user stated assumption that the NEW-COURSE is of type SPICY-RED-MEAT and the ontology statement that SPICY-RED-MEAT-COURSEs should be paired with wines that are members of the class DRINK-HAS-FULL-BODY-RESTRICTION.


Inference Web Registrar - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links
Inference WEB Browser
by Stanford's Knowledge Systems Laboratory
NEW URI
Submit
Lens Magnitude 1 2 3
Lens Width 700
Internet

http://belo.stanford.edu:8080/iwbrowser/WffBrowser?url=http://www.ksl.stanford.edu/software/IW/sample/IW7.daml

Current URL: http://www.ksl.stanford.edu/software/IW/sample/IW7.daml

Proof Lens

```
(<= (holds :type ?inst :|DRINK-HAS-FULL-BODY-RESTRICTION|)
    (holds :type ?inst :|SPICY-RED-MEAT-COURSE|))        (type :|NEW-COURSE12| :|SPICY-RED-MEAT-COURSE|)
------------------------------------------------------------------------------------------------ GMP
              (type :|NEW-COURSE12| :|DRINK-HAS-FULL-BODY-RESTRICTION|)
```

Inference Web: [ Home | Spec | Browser | Registrar | Registry ]

Figure 1: Browsing an IW proof.

An IW proof fragment supporting the explanation above is presented in Figure 2. In contrast with typical proofs that are normally monolithic documents containing cumbersome descriptions of proof steps and formulas, IW proofs are combinations of proof fragments as illustrated by the example in Figure 2. IW proofs are inherently distributed since each proof fragment can be accessed through its URI. As a consequence, the physical location of each proof fragment (and even whether the proof fragment is static or dynamic) is irrelevant for presenting explanations assuming web availability. The proof is machine understandable since it is based on terms of the *Inference Web Ontology*[3] that is written in DAML+OIL [4].

```
<?xml version='1.0'?>
<rdf:RDF
  xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
  xmlns:iw='http://www.ksl.stanford.edu/software/IW/spec/iw.daml#'
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:daml='http://www.daml.org/2001/03/daml+oil#'>
                 (...)
  <iw:APFS>
    <iw:hasFormula>
       <iw:KIF>(type NEW-COURSE FULL-BODY-RESTRICTION)</iw:KIF>
    </iw:hasFormula>
    <iw:isConsequentOf rdf:parseType='daml:collection'>
      <iw:InferenceStep>
        <iw:hasInferenceRule>
          <iw:InferenceRule
                rdf:about='(...)/IW/registry/IR/MP.daml'/>
        </iw:hasInferenceRule>
        <iw:hasInferenceEngine rdf:parseType='daml:collection'>
          <iw:InferenceEngine
                rdf:about='(...)/IW/registry/IE/JTP.daml'/>
        </iw:hasInferenceEngine>
        <iw:hasAntecedent rdf:parseType='daml:collection'>
          <iw:APFS rdf:about='../IW/sample/IW8.daml'/>
          <iw:APFS rdf:about='../IW/sample/IW9.daml'/>
        </iw:hasAntecedent>
      </iw:InferenceStep>
    </iw:isConsequentOf>
  </iw:APFS>
</rdf:RDF>
```

Figure 2: An IW proof fragment.

## 3 Inference Web Proof Specification

In order to provide a logical foundation for our approach, we present IW from a proof theoretic perspective.

### 3.1 Proof Fragments

We begin with some standard terminology used as building blocks for IW proofs and proof fragments. A *formula* in the IW is a first-order statement written in Knowledge Interchange Format (KIF) [7]. A formula labeling a position in a structure (e.g., tree, sequent, etc.) is a *formula occurrence*.

Using standard terminology taken directly from Troelstra and Schwichtenberg [16], "*tree*s are partially ordered sets $(X, \leq)$ with a lowest element and all sets $\{y : y \leq x\}$ for $x \in X$ linearly ordered. The elements of $X$ are called the *nodes* of the tree; *branches* are maximal linearly ordered subsets of X." A single node at the bottom of the tree is specified as the *root* of the tree. If a branch

[3] http://www.ksl.stanford.edu/software/IW/spec/iw.daml

is finite, it ends in a *leaf*. *Deduction trees* are finite trees with all nodes labeled by formulas.

A *n-premise rule* R is a set of formulas $\{S_0, \cdots, S_n, S\}$ of cardinality $n+1$. An instantiation of R is said to be an *inference step*. Within the inference step, $S$ is the *conclusion*, and $S_i$ are the *premises*. An *axiom* is a zero-premise rule. Instances of axioms appear in deduction trees as labels of leaf nodes. Axioms may be statements from an ontology or assumptions posed by a user or premises posed by a query language such as DQL [6].

An *atomic proof fragment* (APF) is a node of a deduction tree labeled by one inference step in addition to the labeling formula. Labeling formulas are formula occurrences. Conceptually one can think of an APF as a single application of a rule used to deduce a single formula. An *atomic proof fragment set* is a set of one or more APFs where all the APFs are labeled by a single formula. Conceptually one can think of an APFS as a set of applications of inference rules used to deduce the identical formula in a single step. APFSs are a critical building block of the Inference Web since they are the key to combination and multiple explanations Figure 2 presents an example of an APFS used on the query example in Section 2. There, the labeling formula is stored under the `<iw:hasFormula>` property of APFSs, as presented in line 10. Moreover, the IW `<iw:isConsequentOf>` property identifies the inference steps of the APFSs. For example, for the APFS in Figure 2, the text starting at line 13 and finishing at line 26 specifies an inference step (and this is the only inference step in the APFS).

An inference step is an instance of the rule referred to as the `<iw:Inference-Rule>` element attached to its `<iw:hasInferenceRule>` property. The conclusion of an inference step is the formula labeling the APFS owning the inference step. The conclusion of the inference step in Figure 2 is the formula in line 10. The premises of an inference step are the formulas labeling APFS associated to the inference step through its `<iw:hasAntecedent>` property. The premises of the inference step in Figure 2 are the formulas labeling the `IW8.daml` and `IW9.daml` APFSs in lines 23 and 24, respectively. If the labeling formula of one APFS is a premise of an inference step of another APFS then these two APFSs are said to be *consecutive* APFSs.

Using the Inference Web terminology, a formal system with *local rules*, or a *LR-system*, is specified by a finite set of rules [16].

## 3.2 Rules and the IW Registry

A description of rules in terms of their formulas is required for identifying the LR-system of a deduction tree. One could potentially specify the set of rules of a new LR-system **T** by using a well known set of inference rules identified say as a Gentzen System LK. Thus, standard literature could be used to access the rules and their associated formulas for very well known systems. We could specify another LR-system **W** by the set of rules supported by a particular inference engine (e.g., JTP [6]). However, this may not suffice for identifying the set of formulas associated with each rule of **W** since a declarative specification of the inference rules, including their formulas, can not be retrieved. For example, line 16 of Figure 2 includes the name of an inference rule but does not include its description. Moreover, while inference engines can create detailed proofs, they rarely can inspect themselves in order to access the set of formulas associated with the rules they implement.

The *IW Registry*[4] is a repository of proof metadata enriching IW proofs, as explained in [10]. Each entry in the Registry has its own URI. Entries are machine understandable since they are based on terms publicly defined in the Inference Web and DAML+OIL ontologies. Thus, using the registry developers of inference engines can provide declarative descriptions of rules that are available on the web. For example, Figure 3 presents the entry for the modus ponens rule used in the APFS described in Section 3.1 (see line 16 of Figure 2). In Figure 3, the set of formulas {`(<= ?A ?B)`,`(holds ?B)`,`(holds ?A)`} of MP are defined KIF formulas in the rule specification. The conclusion `(holds ?A)` is attached to the `<iw:Conclusion>` property of rules in line 12. The premises `(<= ?A ?B)` and `(holds ?B)` are attached to the `<iw:Premises>` property of rules starting in line 14.

```
1  <rdf:RDF
2    xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
3    xmlns:iw='http://www.ksl.stanford.edu/software/IW/spec/iw.daml#'
4    xmlns:daml='http://www.daml.org/2001/03/daml+oil#'
5    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
6         (...)
7    <iw:InferenceRule
8         iw:URL=''>
9      <iw:Name>Modus Ponens</iw:Name>
10           (...)
11     <iw:Conclusion>
12       <iw:KIF>(holds ?A)</iw:KIF>
13     </iw:Conclusion>
14     <iw:Premises rdf:parseType='daml:collection'>
15       <iw:KIF>(<= ?A ?B)</iw:KIF>
16       <iw:KIF>(holds ?B)</iw:KIF>
17     </iw:Premises>
18     <iw:InferenceRuleSource rdf:parseType='daml:collection'>
19        <iw:Source
20           rdf:about='(...)/IW/registry/SRC/RUSSELLNORVIG1995.daml'/>
21     </iw:InferenceRuleSource>
22   </iw:InferenceRule>
23 </rdf:RDF>
```

Figure 3: IW Registry entry for the MP rule.

Following the terminology of Troelstra and Schwichtenberg [16], if **T** is a LR-system, the rules specifying the system are called *primitive rules* of the system. A rule R is said to be a *derivable rule* in **T**, if for each instance $S_0, \cdots, S_n, S$ there is a deduction of $S$ from the $S_i$ by means of the rules of **T**. That is to say, in this deduction the $S_i$ are treated as additional axioms. The Inference Web supports this definition and use of primitive and derivable rules. Derivable rules are defined from APFSs previously generated using primitive rules defined

[4]http://www.ksl.stanford.edu/software/IW/registry

in the IW Registry and other derivable rules. APFSs of derivable rules are incorporated in the IW Registry.

## 3.3 Inference Engine Rules

In the IW architecture, the Registry is responsible for storing documentation of rules as well as declarative definitions of sets of rules implemented by inference engines. Thus, the trivial identification of the inference engine generating an inference step $\gamma$ suffices for specifying the LR-system associated with $\gamma$. For example, Figure 4 presents the Registry entry for JTP. There, the `<iw:InferenceEngineRule>` property of inference engines starting in line 14 and finishing in line 20 identifies the set of rules implemented by JTP. For instance, in line 18 we can see a reference to the modus ponens rule discussed in Section 3.2. The APFS in Figure 2 is from the LR-system specified by JTP since its only inference step is associated with JTP (see line 20 in Figure 2).

```
 1 <rdf:RDF
 2   xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
 3   xmlns:iw='http://www.ksl.stanford.edu/software/IW/spec/iw.daml#'
 4   xmlns:daml='http://www.daml.org/2001/03/daml+oil#'
 5   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 6              (...)
 7   <iw:InferenceEngine>
 8     <iw:Name>Java Theorem Prover</iw:Name>
 9     <iw:URL>http://www.ksl.stanford.edu/software/JTP/</iw:URL>
10                    (...)
11     <iw:InferenceEngineSource rdf:parseType='daml:collection'>
12       <iw:Source rdf:about='(...)/IW/registry/SRC/JTP-TEAM.daml'/>
13     </iw:InferenceEngineSource>
14     <iw:InferenceEngineRule rdf:parseType='daml:collection'>
15       <iw:InferenceRule rdf:about='(...)/IW/registry/IR/Demod.daml'/>
16       <iw:InferenceRule rdf:about='(...)/IW/registry/IR/Enum.daml'/>
17       <iw:InferenceRule rdf:about='(...)/IW/registry/IR/Func.daml'/>
18       <iw:InferenceRule rdf:about='(...)/IW/registry/IR/MP.daml'/>
19                    (...)
20     </iw:InferenceEngineRule>
21   </iw:InferenceEngine>
22 </rdf:RDF>
```

Figure 4: IW Registry entry for JTP.

The registration of the rules of inference engines has demonstrated to be a useful resource for dealing with proofs with multiple LR-systems (e.g., proofs produced by heterogeneous web agents). For instance, during the generation of proofs, inference engines can use the Registry as a background ontology of rule descriptions. Thus, they can simply dump a rule name and users can obtain comprehensive rule definitions from the registry. Additionally inference engines can use the Registry to identify undocumented rules. Proof verifiers can also use the registry to retrieve rule formula specifications. Web tools can produce rich explanations by linking their proofs to entries in the Registry.

## 4 Related and Future Work

Our work on the inference web extends and expands on work explaining description logics initiated by McGuinness [8, 9] and expanded to include explanation of tableaux reasoners [2, 1]. It utilizes the notions of a proof theoretic approach for providing an extensible proof fragment-based foundation for delivering customizable explanations. This work focuses on the new demands of explanation imposed by a web-based deployment of heterogeneous applications potentially using many reasoners and many data sources. It also focuses on the needs gathered from a number of government-sponsored intelligent systems programs such as the DARPA High Performance Knowledge Base program[5], the DARPA Agent Markup Language Program[6], and the DARPA Rapid Knowledge Formation Program[7]. We have gathered requirements from contractors in all programs and also presented Inference Web portable proof specifications for comments. Our initial choice of reasoners for incorporation was driven by the representation and reasoning needs of these programs and more recently also from the needs of the ARDA Novel Intelligence for Massive Data[8] and ARDA Advanced Question and Answering for Intelligence Programs[9].

The Inference Web also looks to research in other areas to gather both requirements and leverage. We draw some motivation from explanation in automated theorem provers such as [5] where they attempt to provide justifications that are more readable than simple proof traces. We also look to the expert systems explanation community for requirements and strategies including the seminal work in systems such as MYCIN [15] to work moving to explanations of design rationale [12]. Lessons learned early in this community that we leverage were that acceptance of reasoning systems is critically dependent upon explanations and deduction traces are inadequate. Our work differs from this and other explanation systems from these communities in its emphasis on distribution, combination, and fragmentation with follow-up question generation.

We believe that our portable proof specification is adequate for the reasoning needs identified in the programs above thus we expect only minor extensions to the proof specification. We do expect additions to the ontology of entries in the IW Registry in order to incorporate more expressive representation of information provenance. We also expect significant expansion of the inference engines and inference rule portions of the registry. Other extensions include a proof checker and specialized support for contradiction explanation and follow-up questions. In the future, we may also consider the use of a meta-language such as the one in Isabelle [13] for building formulas in logics other than first order logic.

[5]http://reliant.teknowledge.com/HPKB/
[6]http://www.daml.org
[7]http://reliant.tecknowledge.com/RKF/
[8]http://www.ic-arda.org/Novel_Intelligence/
[9]http://www.ic-arda.org/InfoExploit/aquaint/

## 5 Conclusions

Our work is motivated by the evolving demands of distributed web service applications. We claim that proofs need to be distributed, sharable, portable, and combinable in order to support explanations for intelligent agents. Since heterogeneous agents require access to explanations if they are to understand each other's answers, we further claim that an approach like Inference Web is required in order to achieve collaboration and trust between distributed web agents.

In this paper, we have presented Inference Web proofs. These are used to provide a uniform solution for distributing sharable proofs. We introduced the notion of an atomic proof fragment as a way of breaking proofs into manageable portions and combining proofs. We also introduced *atomic proof fragment set*s (APFSs) that exploit a URI-based design in order to provide a distributed and portable proof. APFSs are the key for constructing combined proofs since they may contain multiple inference steps potentially generated by heterogeneous applications deriving the same formula. We also described the IW Registry *inference rule entries* and *inference engine entries* used to provide declarative rule descriptions used by multiple applications.

Our implemented example is based on the Wines knowledge base that was designed to contain reasoning and query patterns isomorphic to those found in the PROSE/QUESTAR family of configurators [11]. This knowledge base provides a small and understandable domain while being modeled off of industrial help desk configurator applications. We believe Inference Web provides an extensible solution to explaining web retrieval and deduction.

## References

[1] Alex Borgida, Enrico Franconi, and Ian Horrocks. Explaining *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 *ALC* Subsumption. In *Proc. of the International Workshop on Description Logics (DL'99)*, pages 33–36, Linköping, Sweden, July 1999.

[3] 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.

[4] 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.

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

[6] Richard Fikes, Jessica Jenkins, and Gleb 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] Michael R. Genesereth and Richard Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA, 1992.

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

[9] 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.

[10] Deborah L. McGuinness and Paulo Pinheiro da Silva. Inference Web: Portable ans Sharable Explanations for Question Answering. In *Proc. of the AAAI Spring Symposium Workshop on New Directions for Question Answering*, Stanford, CA, USA, March 2003. AAAI Press. to appear.

[11] Deborah L. McGuinness and Jon Wright. An Industrial Strength Description Logic-based Configurator Platform. *IEEE Intelligent Systems*, 13(4):69–77, July/August 1998.

[12] Johanna D. Moore and William R. Swartout. A Reactive Approach to Explanation. In *Proc. of the 11th International Joint Conference on Artificial Intelligence*, pages 1504–1510, Detroit, MI, USA, August 1989. Morgan Kaufmann.

[13] Lawrence C. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[14] Lawrence C. Paulson. *Handbook of Logic in Computer Science*, volume II, pages 415–475. Oxford, 1992.

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

[16] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Cambridge University Press, Cambridge, UK, second edition, 2000.