DAML Query

Deborah L. McGuinness and Richard Fikes

With contributions from Hayes, Horrocks, Hsu, Jenkins, McCool, Pinheiro da Silva, Joint Committee, …

Knowledge Systems Laboratory

Stanford University

http://www.ksl.stanford.edu

dlm@ksl.stanford.edu

Outline

- DAML Query
  - Semantic Web query language issues
  - OWL-QL (quick review and updates from DQL)
- DAML Query in action
  - Wine Agent example
- DAML Query plus Explanation
  - Inference Web
- Discussion

Semantic Web Query Issues

Context Setting: Queries on the semantic web may:

- be answered by reasoners as well as "look-up" systems (thus answers may be less transparent to clients)
- obtain information from unknown sources (thus users may need support for determining when to trust answers since they know little about provenance or information manipulation)
- need to find "answer kbs" without expecting clients to specify particular sources (thus servers may need to "know" which sources to query)
- need to interact with heterogeneous and dynamically appearing servers (thus servers may want to utilize an API that tells them how to interact with resources)
- be able to use semantics in order to make question answering systems appear more useful, efficient, and robust.

These and other issues motivate DQL and OWL-QL as well as our implementations and future work

OWL-QL Overview

- Query language for deductive query-answering.
- Editors: Fikes, Hayes, Horrocks.
- Based on DAML Query Language (DQL) from the EU/US Joint Committee on Markup Languages
- Source - knowledge represented in OWL on the Semantic Web
- Supports an inter-agent query-answering dialogue
  - Client – the querying agent
  - Server – the answering agent
- The server may derive answers to queries (as well as simply retrieve answers)
- Answers may take an unpredictable amount of time to compute
- There may be an unpredictable number of answers
- The knowledge may be in multiple knowledge bases
- The knowledge bases need not be specified by the client
- For further information –
  - Stanford OWL-QL Web site: ksl.stanford.edu/projects/owl-ql/
  - Paper: ksl.stanford.edu/KSL_Abstracts/KSL-03-14.html

Representation Language Independence

- A query contains a query pattern
  - A KB with some URIrefs designated as variables
  - Specifies a sentence schema
- Answers are determined from an "answer KB"
- An answer provides bindings for variables in the query pattern
  - Specifies a sentence that is entailed by the answer KB
- The KBs and sentences can be in any sentential representation language with a formal theory of logical entailment

  E.g., DQL has been used to support KIF queries and KBs
- So, converting DQL to OWL-QL was straightforward

Queries Can Be Rejected

- A server returns answers in bundles
- An answer bundle contains –
  - A process handle or
  - Termination tokens
- A server ends a dialogue by sending termination tokens
  - End – no further answers will be produced by the server
  - None – no further answers are entailed by the answer KB
  - Rejected – query is outside the server's scope of queries

Redundant Answers

- Clients want to know whether a server returns duplicate or redundant answers

  E.g., a variable that is a value of a maxCardinality restriction could have a binding of 5 or 6 or 7 or …

- Eliminating duplicate and redundant answers can be very expensive

  E.g., are "Golfer" and "Scientist" redundant bindings for V in {(type Joe V)}?

- OWL-QL specifies a set of conformance levels for servers
  - Non-repeating – No duplicate answers
  - Terse – No redundant answers
  - ☛ Serially terse – No answers redundant with previous answers
- Guaranteeing terseness is a harsh requirement
  - Produce all answers before returning any, or
  - Can't produce most specific answer because less specific answer already produced
- Expect most applications will use serially terse servers
- ☛ Extended definition of redundancy to include values of cardinality restrictions (one of coming)
  - Could not extend to types because of difficulty of deriving (not (subclassOf …))

Answering "How Many" Queries

- The number of answers produced by a server is not "how many"
  - The server may not guarantee it has found all of the answers
  - Bindings for a variable in multiple answers may all denote the same entity

  E.g., Client asks for X such that X is type Car and is owned by Joe.

  Server produces bindings Car1, Car2, and Car3 for X.

  There could be more answers to the query.

  Perhaps Car1=Car2 or Car1=Car3 or Car2=Car3.

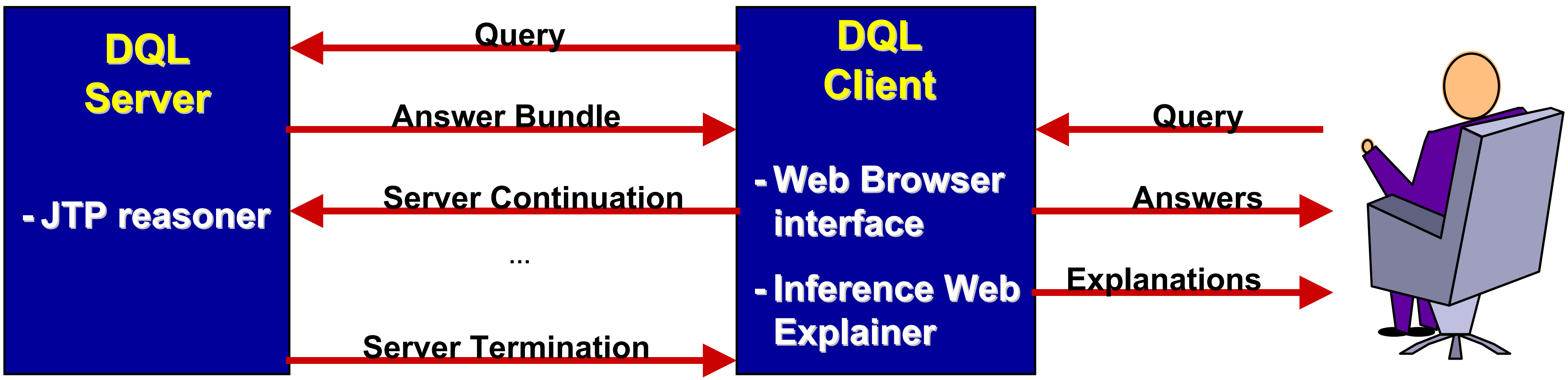
  Only can conclude that Joe owns at least one car.

- "How many" queries need to be formulated as a query about the value of a cardinality restriction

  E.g., Ask what is the value of a cardinality restriction on property ownsCar for Joe?, where ownsCar is a subproperty of owns that has an allValuesFrom restriction of Car for Joe

- OWL-QL does allow a query to include an <u>answer number request</u>
  - Many database servers record information about the number of entries in their data tables and can rapidly respond to requests for this information

Answer Generation System



- DAML Query Language (DQL – OWL-QL)
  - Agent to agent protocol for deductive query answering
- JTP hybrid reasoning system
  - Includes temporal reasoner, DAML/OWL reasoner, …
- Inference Web
  - Provide proofs and explanations

Wine Example

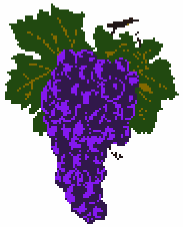
- Choose a food – either a particular one such as crab or a general one.
- Application then generates a query in DQL to JTP which provides answers along with portable proofs so that user can ask for explanations.
- Connects to web sites for dynamic queries for real time information
- Info: http://www.ksl.stanford.edu/people/dlm/webont/wineAgent/
- Work with McGuinness, Hsu, Jenkins, McCool, Pinheiro da Silva

Wine Agent (version 1.0) - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://onto.stanford.edu:8080/wino/index.jsp Links



Wine Agent 1.0

How does it work?

Please select a type of course:

SEAFOOD	RED MEAT	PASTA	DESSERT
Fish:	• regular red meat	• pasta w/ regular red sauce	• sweets
• bland fish	• spicy red meat	• pasta w/ spicy red sauce	• nuts and cheese
• flavorful fish		• pasta w/ light cream sauce	
Shellfish:	WHITE MEAT	• pasta w/ heavy cream sauce	FRUIT
• oysters	• light-meat fowl		• sweet fruit
• other shellfish	• dark-meat fowl	TOMATO-BASED FOOD	• unsweet fruit

---

Or, select a specific item from the sample menu:

Starters: Dozen clams - Dozen oysters - Dozen mussels - Personal cheese pizza

Poultry: Rotisserie chicken - Roast duck - Roast goose - Roast turkey

Meat: Grilled T-Bone steak - 10 oz. Prime rib - Garlicky roast beef tenderloin - Grilled veal chops - Grilled pork chops - Lamb curry

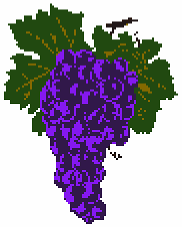
Pasta: Spaghetti with tomato sauce - Fetuccine Alfredo - Fra Diavolo - Linguine with white clam sauce

Internet

Wine Agent (version 1.0) - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://onto.stanford.edu:8080/wino/index.jsp?aqstring=INDVFOOD-CRAB



Wine Agent 1.0

How does it work?

Course Type: NON-OYSTER-SHELLFISH

"Pairs well with *dry white* varieties. *Full*-bodied wines match especially well." why?

The local knowledge base particularly recommends the following:

- CHATEAU DE MEURSAULT MEURSAULT
- MOUNTADAM CHARDONNAY
- FORMAN CHARDONNAY
- CORBANS PRIVATE BIN SAUVIGNON BLANC
- FOXEN CHENIN BLANC
- CORTON MONTRACHET WHITE BURGUNDY
- KALIN CELLARS SEMILLON

The recommended wines can be found below, along with some comparable selections:

Web Inventory Search

Internet

Why?

- Provides information concerning answers
  - Meta information concerning sources, question answering system
  - Reasoning path to answer

Inference Web

Framework for *explaining* question answering tasks by storing, exchanging, combining, annotating, filtering, segmenting, comparing, and rendering proofs and proof fragments.

- DAML/OWL *specification of proofs* is an interlingua for proof interchange
- *Proof browser* for displaying IW proofs and their explanations (possibly from multiple inference engines)
- *Registration* for inference engines/rules/languages
- *Proof explainer* for abstracting proofs into more understandable formats
- *Proof generation service* to facilitate the creation of IW proofs by inference engines
- *Prototype implementation* with Stanford's JTP reasoner and SRI's SNARK reasoner
- Integrated with DQL and JTP in a few web agents for demonstrations
- Discussions with Boeing, Cycorp, Fetch, ISI, Northwestern, SRI, UT, UW, W3C, …

info: www.ksl.stanford.edu/software/iw

Discussion

- Architecture used in:
  - KSL Wine Agent
  - AQUA – Question Answering Effort for AQUAINT
  - Laptop buying demonstration scenario for PAL
- Provides foundation for working Query Manager design document for cooperative query answering for CALO
  - accepting queries in OWL or KIF
  - Uses JTP's hybrid reasoning architecture
  - Inference Web for explanation
  - OAA for interoperation and special purpose question answerer
  - ISI's query planner
- OWL-QL info available from: ksl.stanford.edu/projects/owl-ql/, ksl.stanford.edu/projects/dql/
- Inference Web info: www.ksl.stanford.edu/software/iw/ & ISWC conf paper

EXTRAS

Wine Example

If C1 is a Seafood Course and W1 is a drink of C1, what color is W1?

P: (type C1 Seafood-Course) (drink C1 W1)

Q: (has-color W1 ?x) must-bind ?x

A: White

answer KB **(the KB that this query is being asked against) contains:**

```
<rdfs:Class rdf:ID="SEAFOOD-COURSE">
   <owl:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#DRINK"/>
         <owl:toClass>
         <owl:Restriction>
                  <owl:onProperty rdf:resource="#COLOR"/>
                  <owl:hasValue rdf:resource="#WHITE"/>
   </owl:Restriction> </owl:toClass> </owl:Restriction> </owl:subClassOf> </rdfs:Class>
```

Wine Premise /Query

- <owl-ql:premise>

  <rdf:RDF>

  <rdf:Description rdf:about="#C1">

  <rdf:type rdf:resource="#Seafood-Course"/>

  <drink rdf:resource="#W1"/>

  </rdf:Description> </rdf:RDF> </owl-ql:premise>

- <owl-ql:queryPattern>

  <rdf:RDF>

  <rdf:Description rdf:about="#W1">

  <has-color rdf:resource="http://www.w3.org/2003/10/owl-ql-variables#x"/>

  </rdf:Description> </rdf:RDF> </owl-ql:queryPattern>

Query Answer

- After answer pattern specified…
- <owl-ql:binding-set>

```
    <var:x rdf:resource="#White"/>
    </owl-ql:binding-set>
  <owl-ql:answerPatternInstance>
    <rdf:RDF>
    <rdf:Description rdf:about="#W1">
    <has-color rdf:resource="#White"/>
    </rdf:Description>
  </rdf:RDF>
  <owl-ql:answerPatternInstance>
```

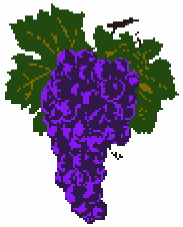
Example from:http://ksl.stanford.edu/projects/owl-ql/syntax.shtml

Wine Agent (version 1.0) - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://onto.stanford.edu:8080/wino/index.jsp?aqstring=SEAFOOD Links



Wine Agent 1.0

How does it work?

Course Type: SEAFOOD

"Pairs well with *white* varieties." why?

The local knowledge base particularly recommends the following:

- CONGRESS SPRINGS SEMILLON
- CHATEAU DE MEURSAULT MEURSAULT
- SELAKS SAUVIGNON BLANC
- MOUNTADAM RIESLING
- MOUNTADAM CHARDONNAY
- CORBANS SAUVIGNON BLANC
- FORMAN CHARDONNAY
- CORBANS PRIVATE BIN SAUVIGNON BLANC
- BANCROFT CHARDONNAY
- FOXEN CHENIN BLANC
- MOUNT EDEN VINEYARD EDNA VALLEY CHARDONNAY
- STONLEIGH SAUVIGNON BLANC
- PULIGNY MONTRACHET WHITE BURGUNDY

Done Internet

Iterative Optimization

- **Query patterns have the same expressivity as OWL**

  E.g., cannot directly ask for most specific subclass of a given class

  - **Rationale is not to burden a server beyond reasoning in OWL**

- **Can indirectly find optimum values of variables as follows:**
  - **To optimize the value of a must-bind variable V in a query Q with respect to a transitive property P and a server S:**
    - **Send Q to S asking for at most one answer.**
    - **If S provides an answer to Q with a binding of Bi for V, then**
      - Send S a query Q' consisting of Q with the additional premise "(P Bi V)" and asking for at most one answer.
      - If S does not provide an answer to Q', then Bi is the optimal binding that S can provide for V.
      - If S provides an answer to Q' with a binding of Bj for V, then
        - Continue this iterative querying until S does not provide an answer.
    - **The last binding produced for V is the optimal binding that S can provide for V.**

Motivation - TRUST

If users (humans and agents) are to use and integrate web application answers, they must trust them.

System transparency supports understanding and trust.

Even simple “lookup” systems should be able to provide information about their sources.

As question answering systems become more complex, they may incorporate multiple hybrid information sources, multiple information manipulation techniques, integration of reasoners, conflict resolution strategies, prioritization, assumptions, etc., all of which may need explanation.

Thus, systems should be able to explain their actions, sources, and beliefs.

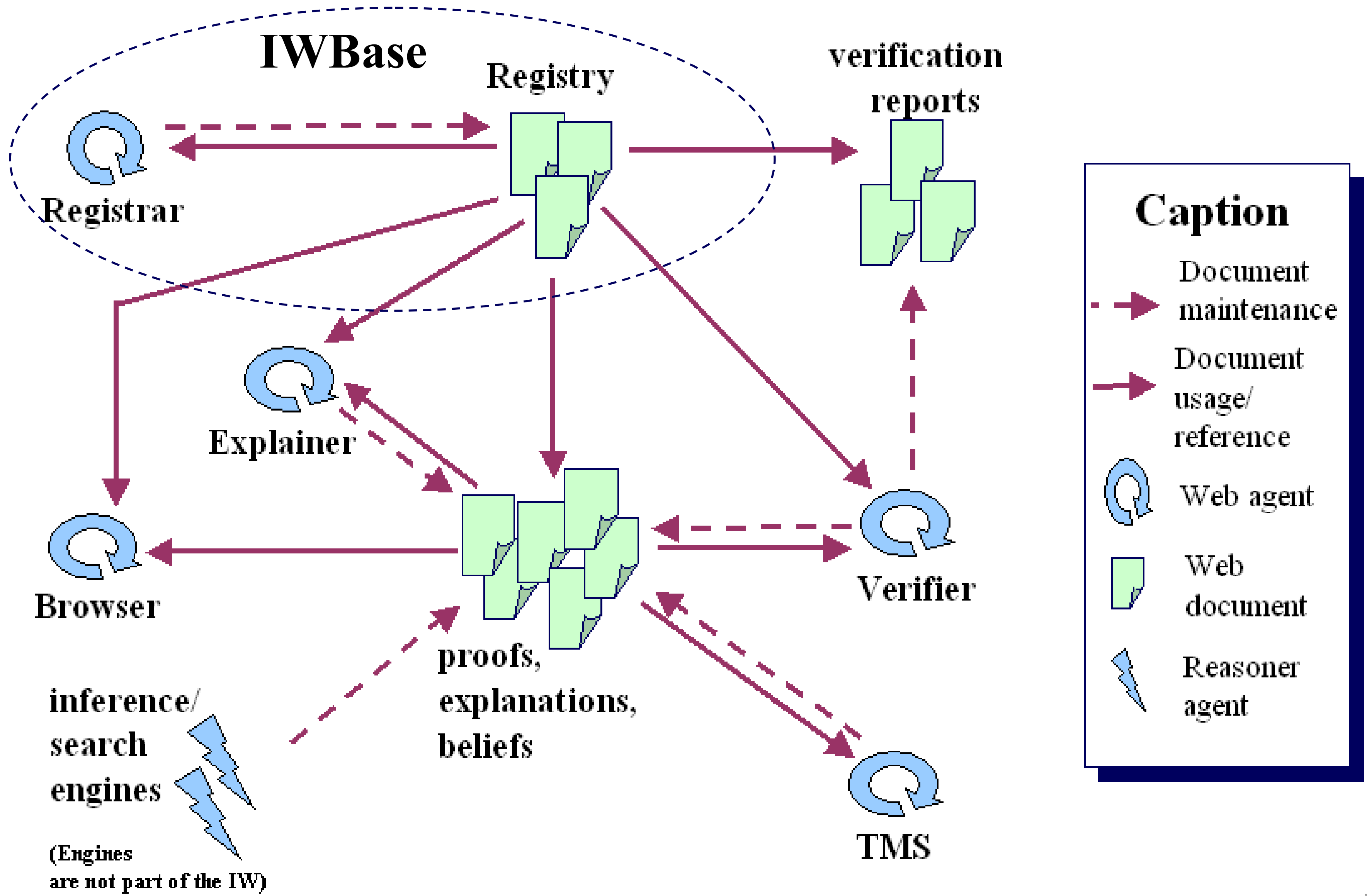
Inference Web

Framework for *explaining* question answering tasks by storing, exchanging, combining, annotating, filtering, segmenting, comparing, and rendering proofs and proof fragments.

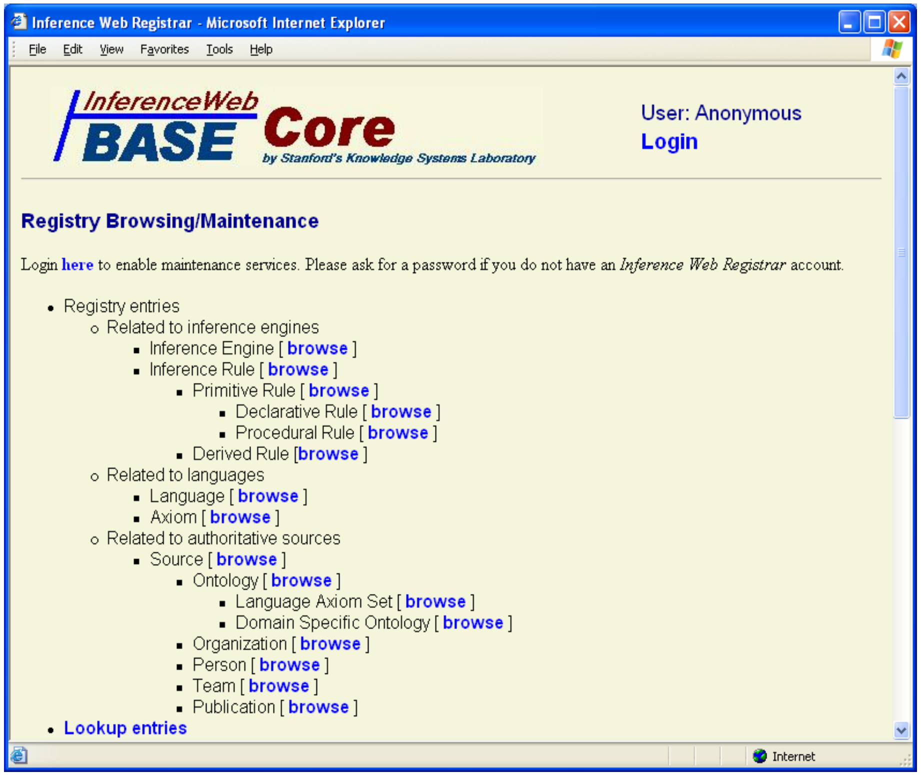
- DAML/OWL *specification of proofs* is an interlingua for proof interchange
- *Proof browser* for displaying IW proofs and their explanations (possibly from multiple inference engines)
- *Registration* for inference engines/rules/languages
- *Proof explainer* for abstracting proofs into more understandable formats
- *Proof generation service* to facilitate the creation of IW proofs by inference engines
- *Prototype implementation* with Stanford's JTP reasoner and SRI's SNARK reasoner
- Integrated with DQL and JTP in a few web agents for demonstrations
- Discussions with Boeing, Cycorp, Fetch, ISI, Northwestern, SRI, UT, UW, W3C, …

info: www.ksl.stanford.edu/software/iw

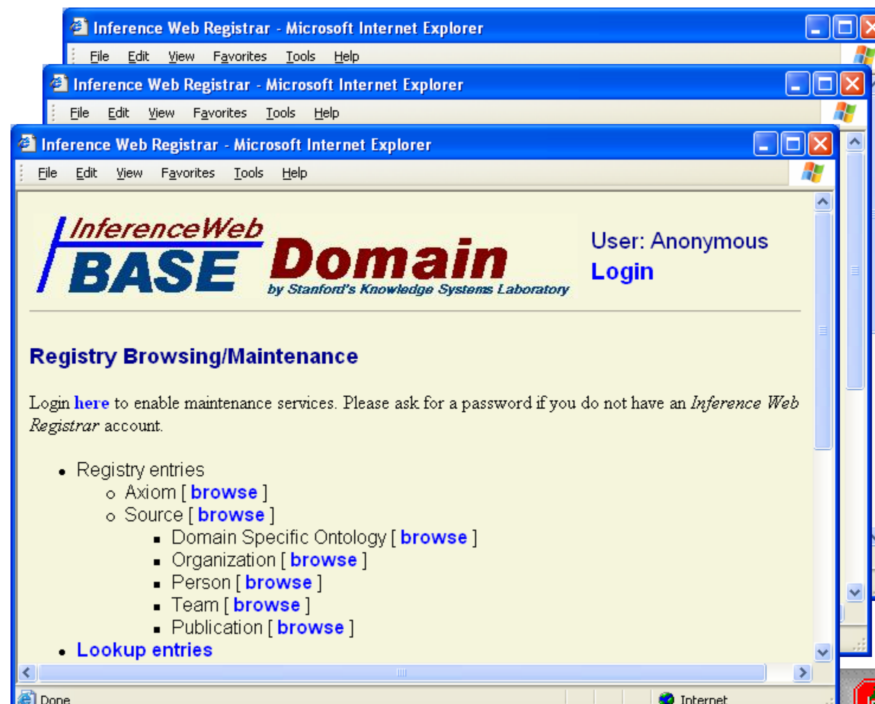
Inference Web Architecture



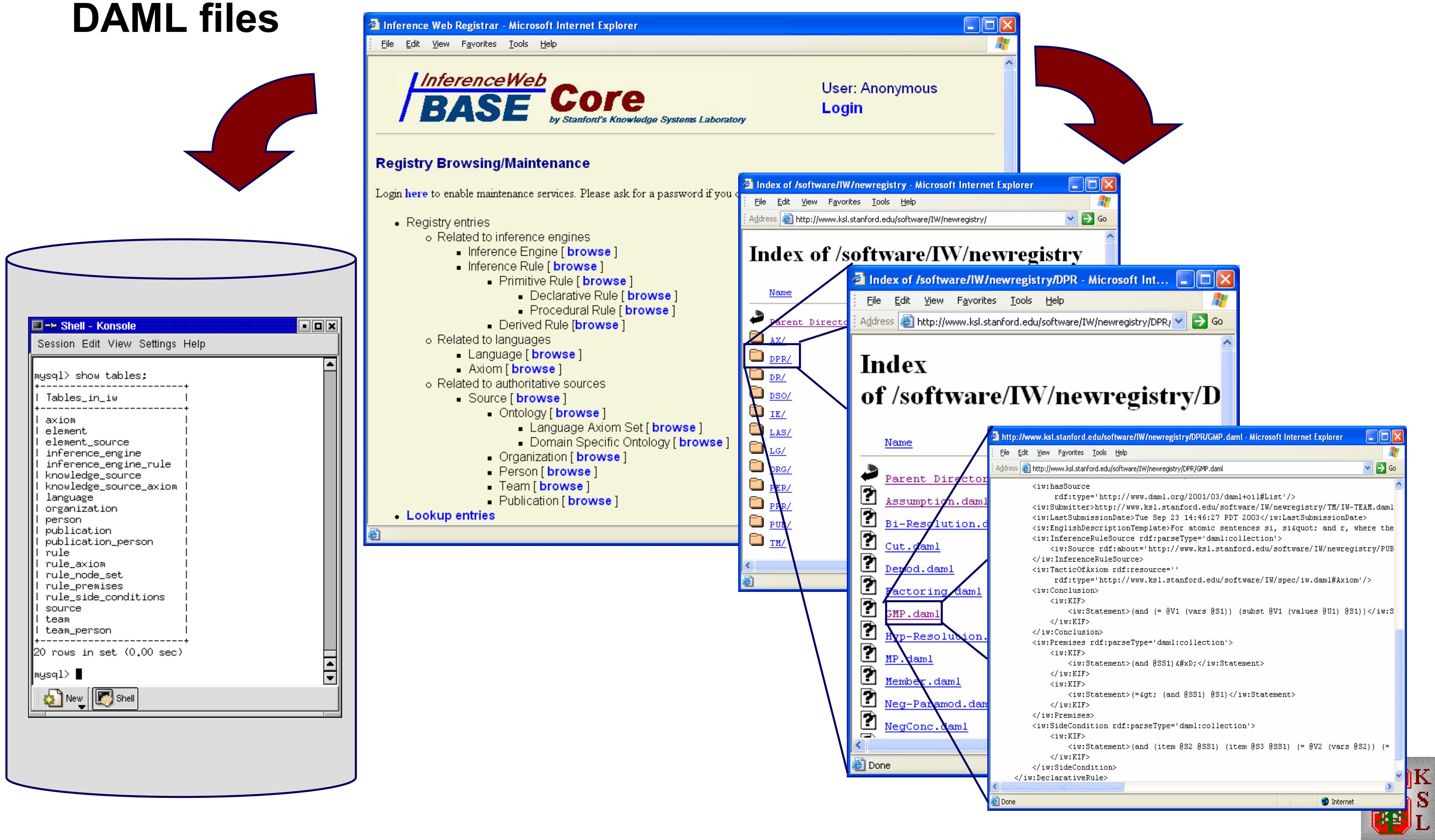
■ **Composed of a Core node …**



… and multiple Domain-specific nodes



IWBase entries are stored both in a database and in a repository of DAML files



InferenceWeb BASE

Registration of Inference Rules

Inference Web Registrar - Microsoft Internet Explorer

Address: http://belo.stanford.edu:8080/iwregistrar/Lookup?uri=http%3A%2F%2Fwww.ksl.stanford.edu%2Fsoftware%2FIW%2Fnewregistry%2FDPR%2FBi-Resolution.daml%23Bi-Resolut

InferenceWeb BASE Core by Stanford's Knowledge Systems Laboratory

User: Anonymous

Login

Declarative Rule: Bi-Resolution

- **Name**: Binary Resolution
- **Description in English**: "Binary resolution always focuses on two clauses and one literal in each. To admit a conclusion, the literals must be opposite in sign and alike in predicate, and there must exist a unifier (substitution of terms for variables) to otherwise make them identical. If a conclusion results, it is obtained by applying the unifier to the two clauses excluding the two literals in focus, and taking the union of the transformed literals."
- **Example in English**: Suppose we have as axioms the following clauses:

```
(or (mortal ?x) (not (human ?x)))
(human Socrates)
```

By substituting "Socrates" for "?x", the second literal of the first clause is made identical with the negation of the sole literal of the second clause. The opposed literals are resolved, leaving as a conclusion the resolvent

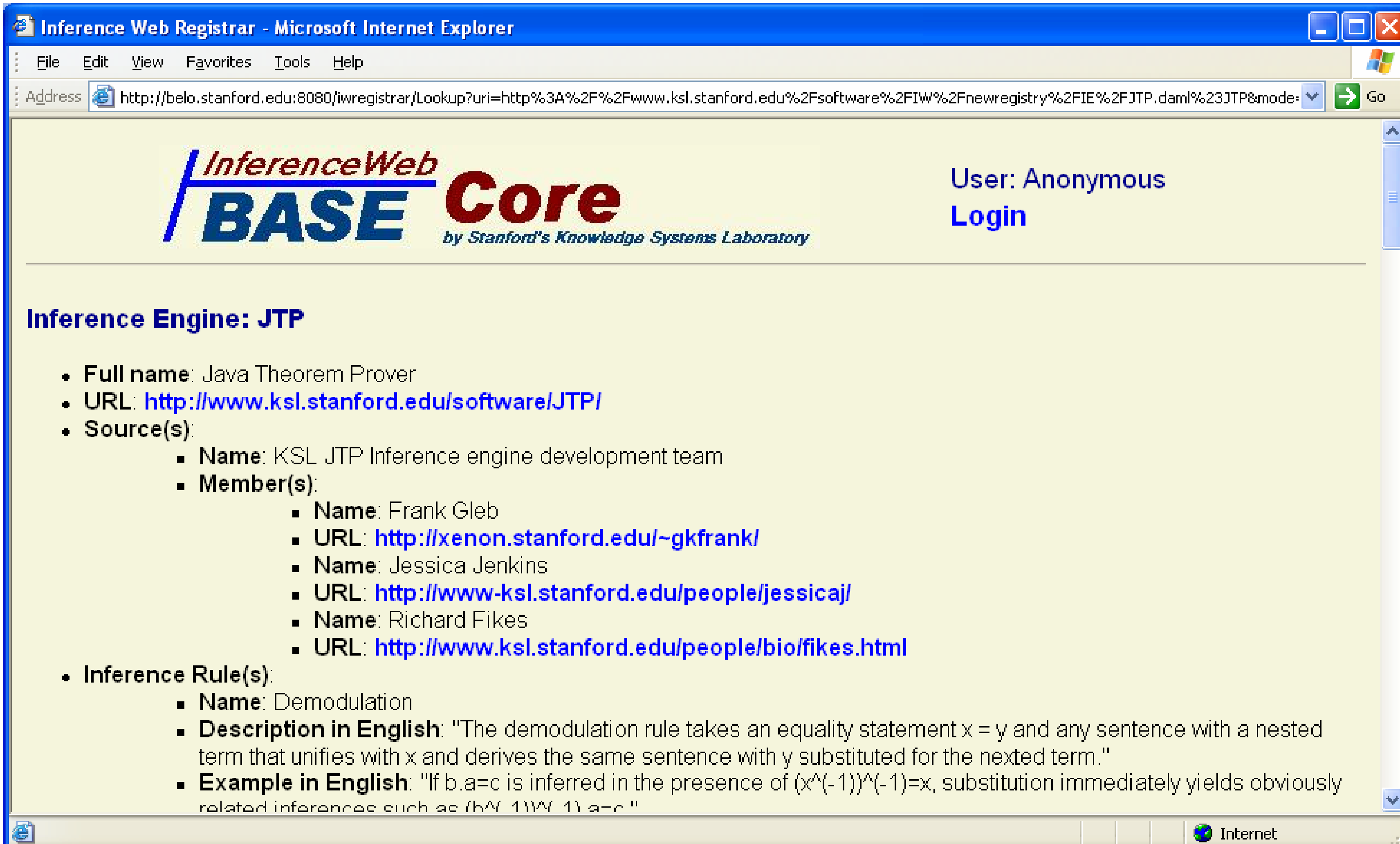
```
(mortal Socrates)
```

- **Premise(s)**:
  - (or @L1 @C1)
  - (or (not @L1) @C2)
- **Conclusion**: (or (or (subst @V1 (values @U1) @C1)) (subst @V2 (values @U1) @C2))
- **Side Condition(s)**:
  - (= (= @V1 (vars @C1)) (= @V2 (vars @C1)) (subst @V1 (values @U1) @C1) (subst @V2 (values @U1) @c2))
- **Source(s)**:
  - **Name**: A Summary of Inference Rules Used by Argonne Automated Deduction Software
  - **URL**: http://www-unix.mcs.anl.gov/AR/inf_rules.html
  - **Name**: Otter development team
  - **Member(s)**:

Internet

InferenceWeb BASE

Registration of Inference Engines



Description of Inference Engine's Capabilities

Inference Web Registrar - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://belo.stanford.edu:8080/iwregistrar/IRsOfIEs

Main Menu > Rules Per Inference Engine

Inference Engine	Primitive Rule
Java Theorem Prover	Demodulation Direct assertion Function Rule Generalized Modus Ponens Membership Rule Reformulation Subsumption Rule Time Point for Temporal Reasoning Time Point Mapping
Otter	Binary Resolution Factoring Hyperresolution Negative Paramodulation Paramodulation Unit Deletion UR-Resolution
SNARK - SRI New Automated Reasoning Kit	Assumption Binary Resolution Direct assertion Hyperresolution Negated Conclusion Paramodulation
The Knowledge Machine	Direct assertion

Internet

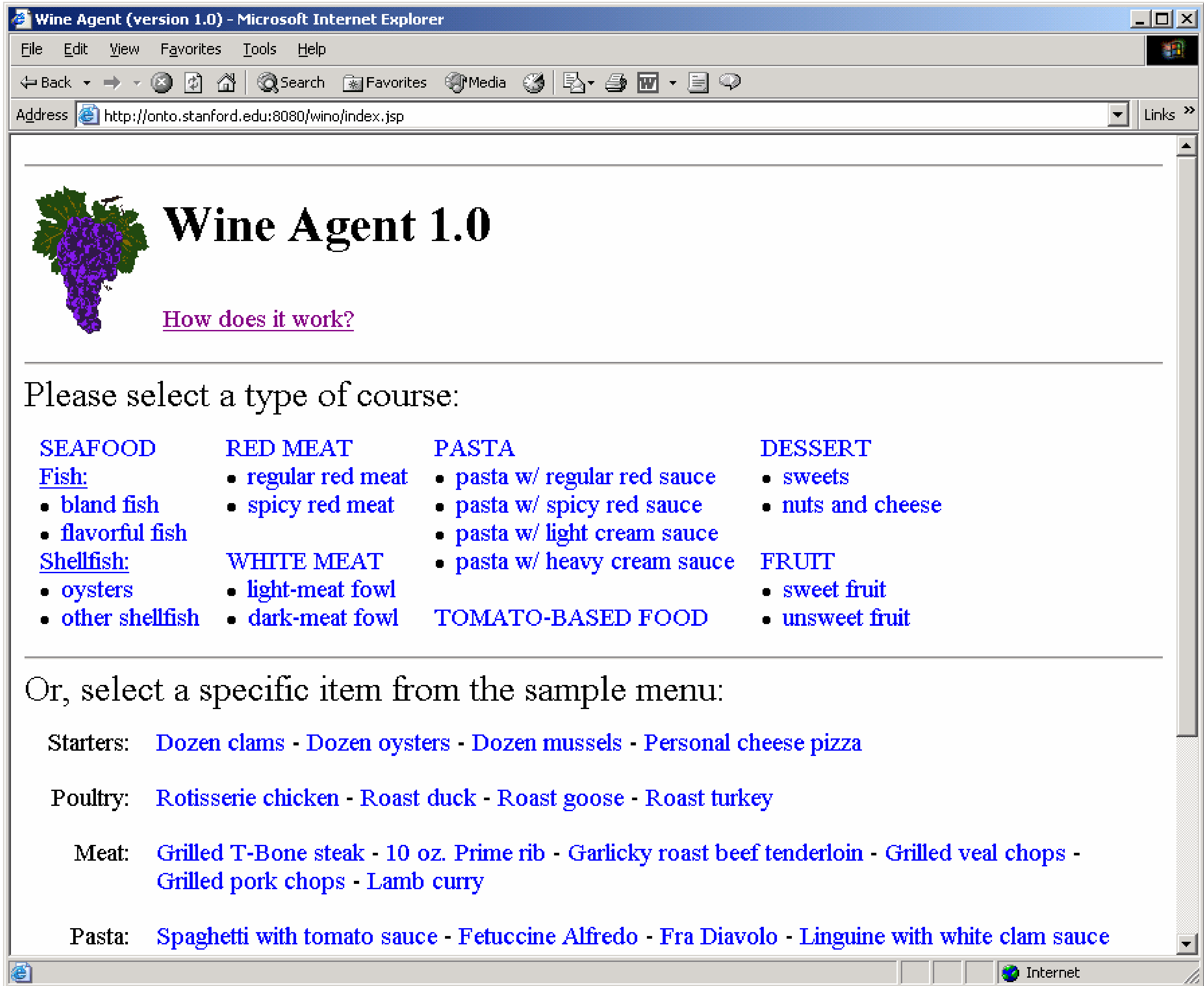
Example

Suppose you are using the KSL Wine Agent - http://www.ksl.stanford.edu/people/dlm/webont/wineAgent/

which gives recommendations about what kinds of wines to drink with particular meals (and helps find those wines for purchase on the web).

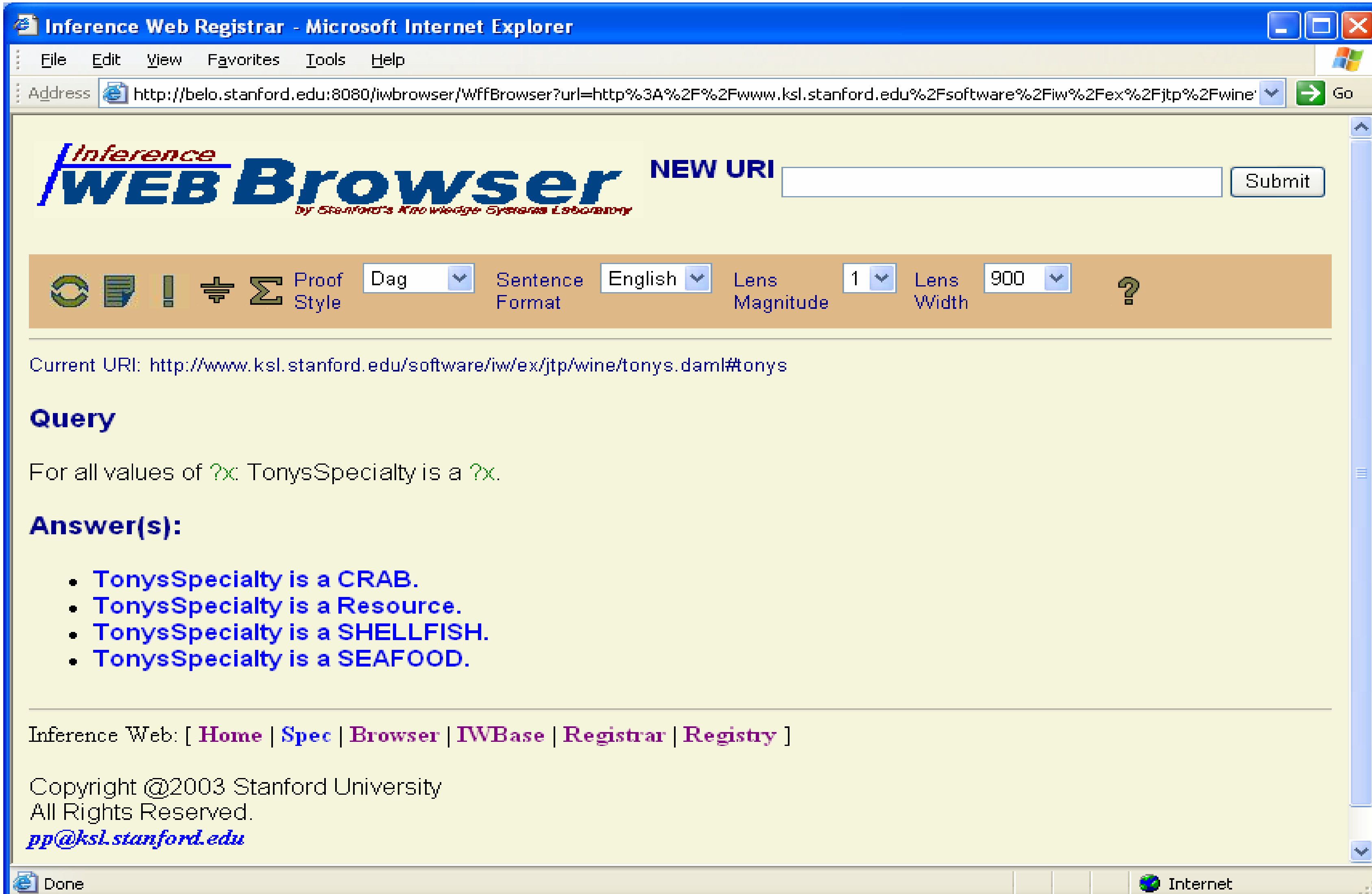
Suppose you choose a meal and are interested in the types of food the meal is classified under and you are interested in finding out about why the system recommended a particular wine

Wine Agent

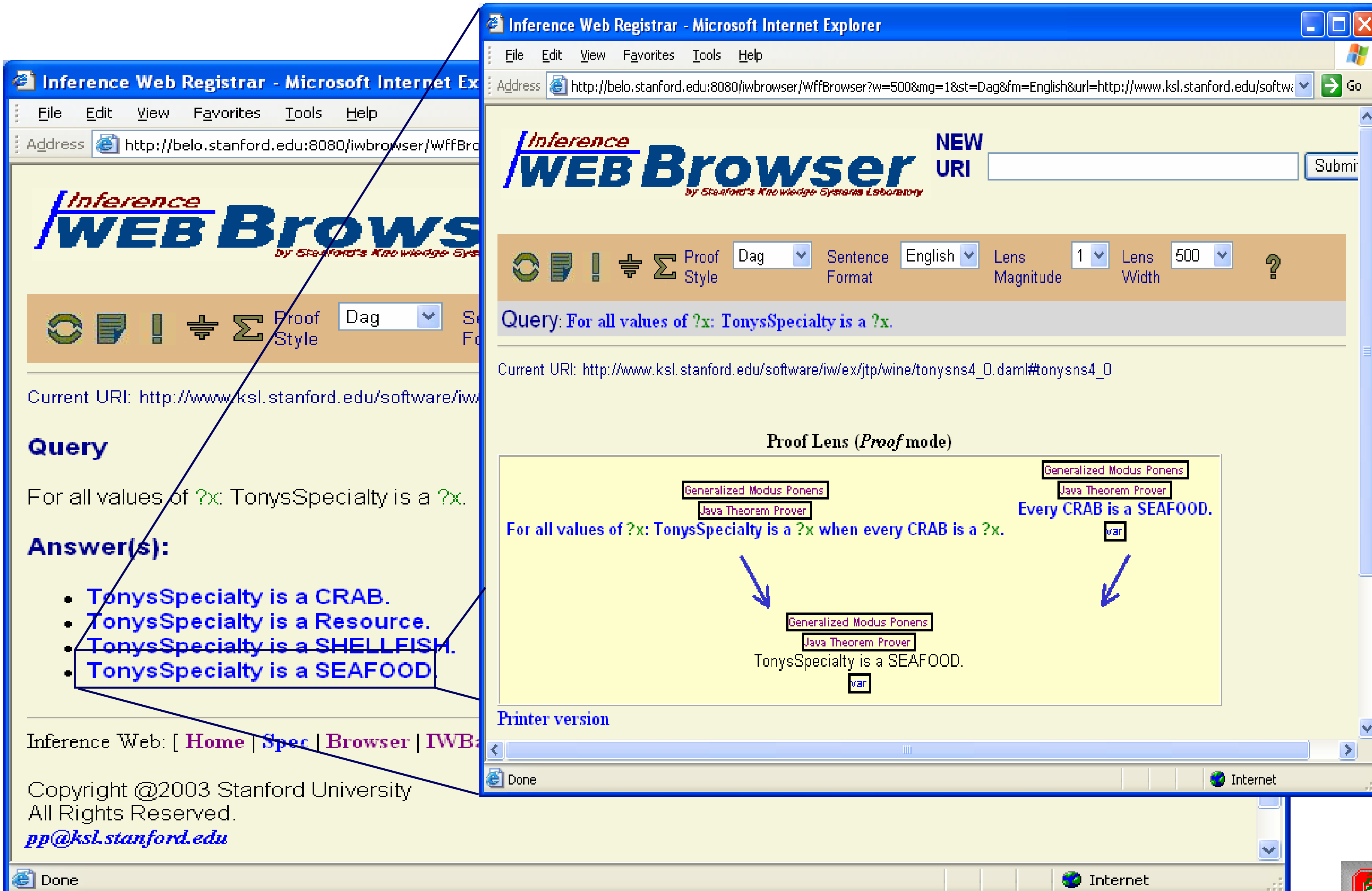

Wine Agent (version 1.0) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://onto.stanford.edu:8080/wino/index.jsp
Links
Wine Agent 1.0
How does it work?
Please select a type of course:
SEAFOOD
Fish:
bland fish
flavorful fish
Shellfish:
oysters
other shellfish
RED MEAT
regular red meat
spicy red meat
WHITE MEAT
light-meat fowl
dark-meat fowl
PASTA
pasta w/ regular red sauce
pasta w/ spicy red sauce
pasta w/ light cream sauce
pasta w/ heavy cream sauce
TOMATO-BASED FOOD
DESSERT
sweets
nuts and cheese
FRUIT
sweet fruit
unsweet fruit
Or, select a specific item from the sample menu:
Starters: Dozen clams - Dozen oysters - Dozen mussels - Personal cheese pizza
Poultry: Rotisserie chicken - Roast duck - Roast goose - Roast turkey
Meat: Grilled T-Bone steak - 10 oz. Prime rib - Garlicky roast beef tenderloin - Grilled veal chops - Grilled pork chops - Lamb curry
Pasta: Spaghetti with tomato sauce - Fetuccine Alfredo - Fra Diavolo - Linguine with white clam sauce
Internet

Browsing a Query

What kind of thing is Tony's Speciality ?

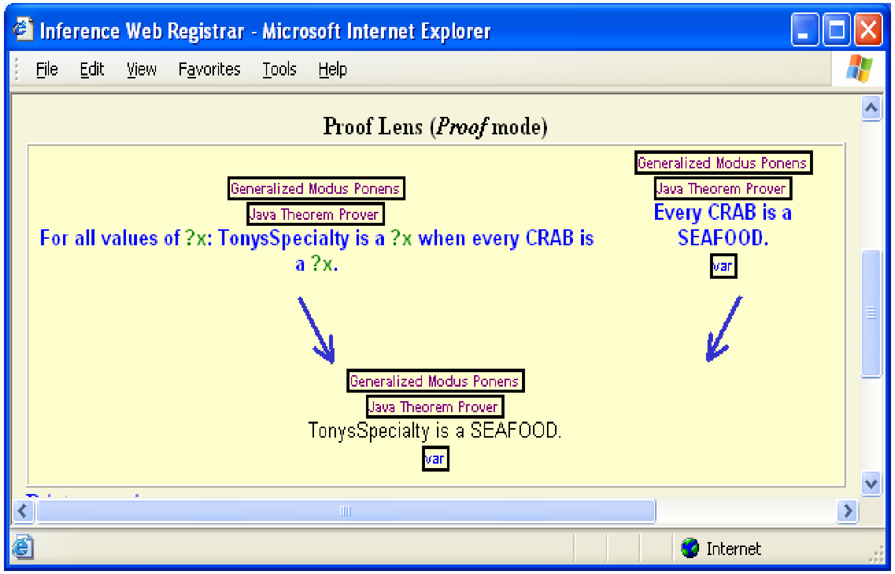


Browsing an Answer Proof

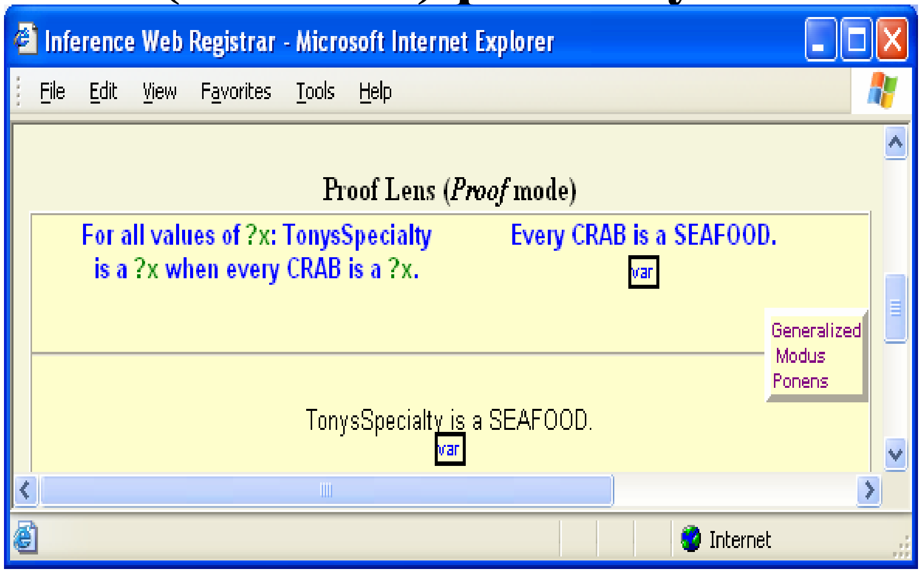


Multiple Browsing Styles

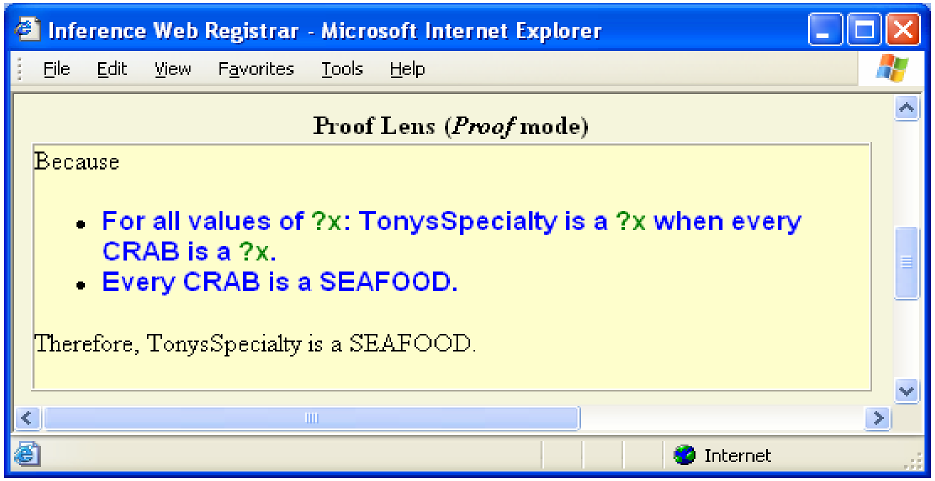
dag style



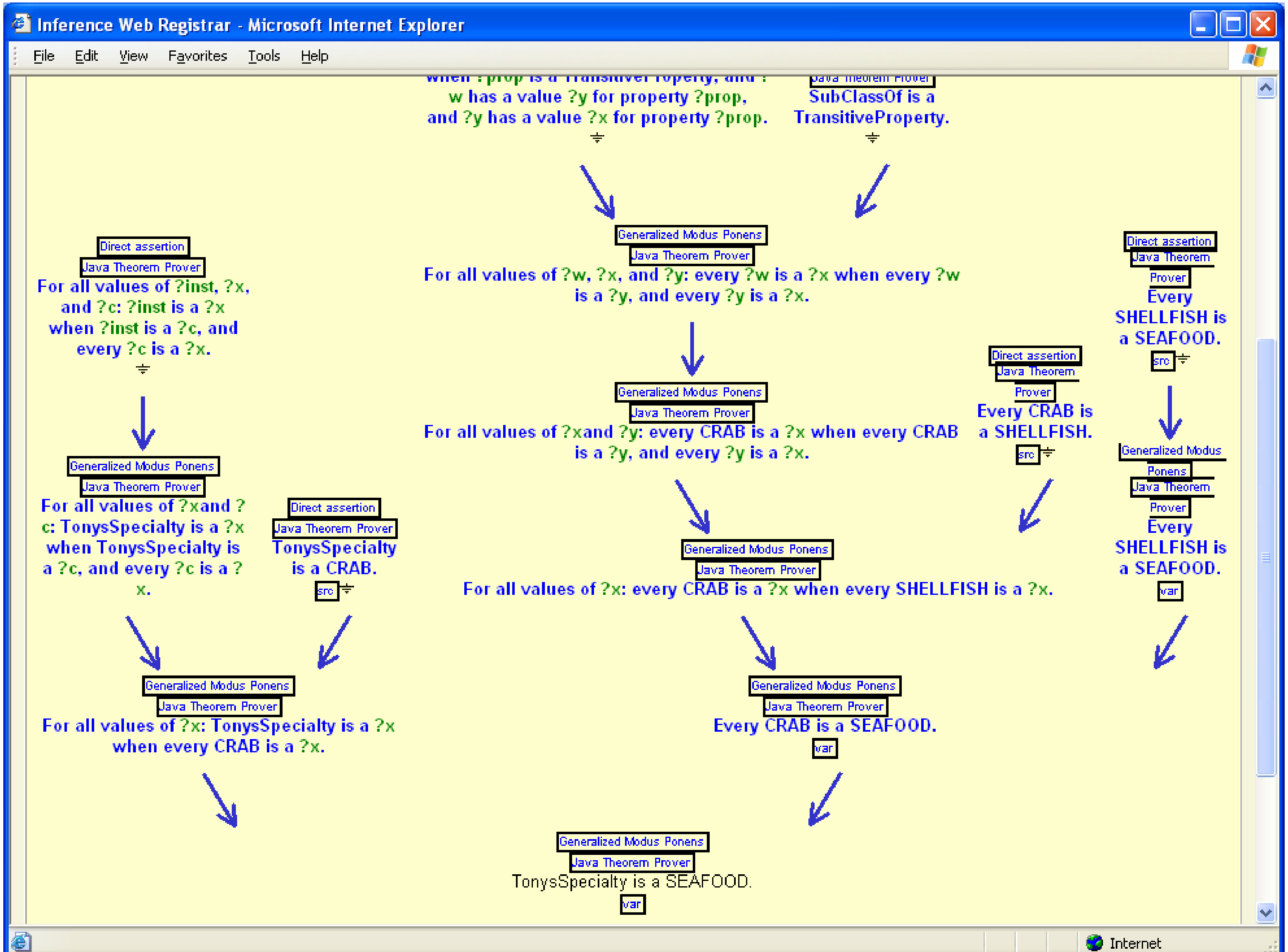
(textbook) proof style



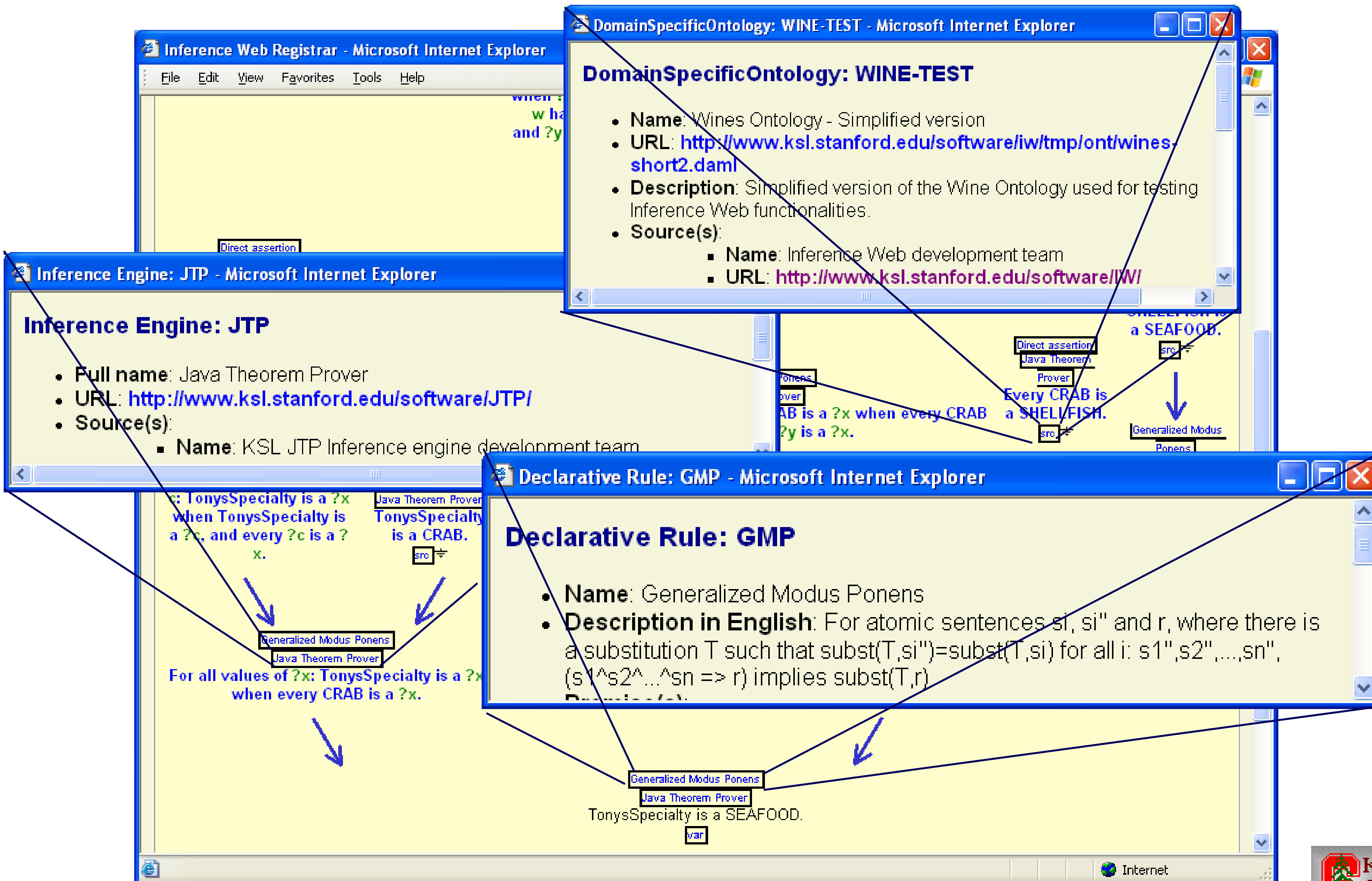
(restricted) English style



Diving Deep in a Proof



Asking Follow Up Questions



Knowledge Provenance

Inference Web Browser - Microsoft Internet Explorer

Knowledge Provenance Elicitation

Current sentence

- TonysSpecialty is a SEAFOOD.

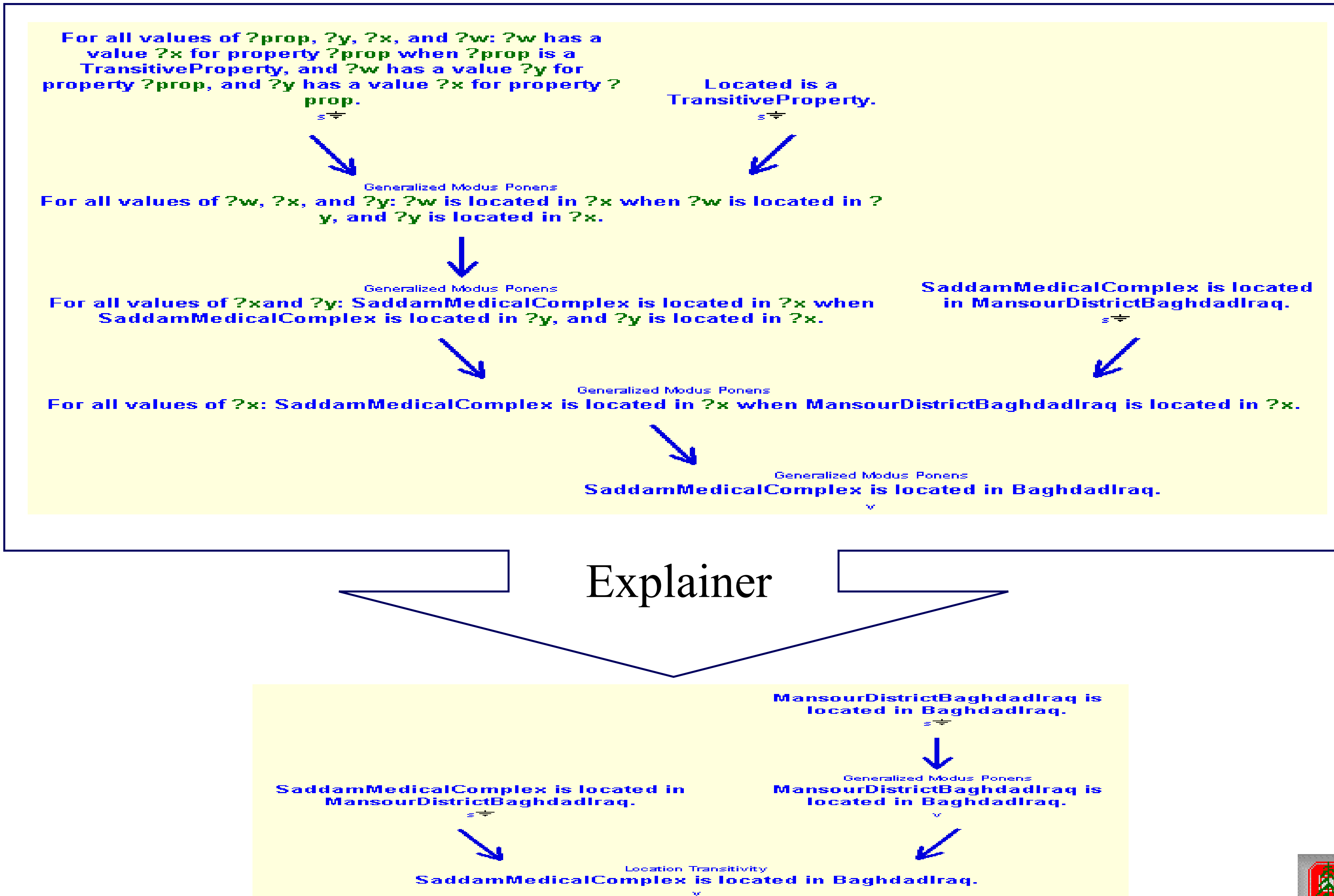
Ground axioms

- Every CRAB is a SHELLFISH.[1] **Direct assertion**
- SubClassOf is a TransitiveProperty.**Direct assertion**
- For all values of ?inst, ?x, and ?c: ?inst is a ?x when ?inst is a ?c, and every ?c is a ?x.**Direct assertion**
- Every SHELLFISH is a SEAFOOD.[1] **Direct assertion**
- For all values of ?y, ?x, ?prop, and ?w: ?w has a value ?x for property ?prop when ?prop is a TransitiveProperty, and ?w has a value ?y for property ?prop, and ?y has a value ?x for property ?prop.**Direct assertion**
- TonysSpecialty is a CRAB.[1] **Direct assertion**

Sources of the ground axioms

1. Wines Ontology - Simplified version
   - **URL**: http://www.ksl.stanford.edu/software/iw/tmp/ont/wines-short2.daml
   - **Description**: Simplified version of the Wine Ontology used for testing Inference Web functionalities.
   - **Source(s)**:
     - **Name**: Inference Web development team
     - **URL**: http://www.ksl.stanford.edu/software/IW/

Explanation Generation



Conclusion

- Proof specification (DAML Proof) ready for feedback/use

  http://www.ksl.stanford.edu/software/iw/

- Proof browser prototype operational and expanding (aggregation views, multiple formats, simplification, pruning, …)
- Registration service expansion - integration with XML database, use in PAL, registration of services (with Fetch)
- Inference engine integration work JTP functional, SNARK mostly done, KM under investigation.
- Integration with web services – current: KSL Wine Agent, KSL DQL client (NIMD implementation), begin with registration of web services (TAP, Fetch), begin explanation of service composition (with McIlraith) and query planning (Knoblock)
- More comments solicited (thanks so far to Berners-Lee, Chalupsky, Chaudhri, Clark, Connolly, Forbus, Hawke, Hayes, Lenat, Murray, Porter, Reed, Waldinger, …)