Semantic Web Technologies and Automated Auctions

Papers:

"Implementing Semantic Interoperability in Electronic Auctions"
- Juha Puustjarvi (2007)

"Ontologies for supporting negotiation in e-commerce"
- Tamma, et Al (2004)

Presentation by
M. Meyer

What is the Semantic Web?

- The Semantic Web is a proposed evolution of the Web in which the semantics of the information and services available on the Web will be formally defined and made available.
- Broad implementation of the Semantic Web would:
  1. Provide a common framework to allow data to be shared and reused across application, platform and enterprise boundaries.
  2. Make it possible for machines (Intelligent Agents) "to access, understand and manipulate web content as readily as humans do". {Berners-Lee, 2001 #1}.

Intelligent Agents and the Semantic Web

1. The current incarnation of the World Wide Web (WWW) is designed almost exclusively to present data and information to humans.
2.
3. An AI agent, on the current WWW, will find it very difficult too:
   - Answer a complex query based on background knowledge.
   - Locate and use information contained within data repositories.
   - Find and use “web services”.

Semantic Web - Basic Requirements

In order to make information on the WWW accessible and processable by machines and intelligent agents, we need to add two things to the information stored on the WWW:

1. Formal Structure:
   - Humans are able to determine the meaning of words by context (ex: the word "address").
   - Machines will need data encoded so that context can be determined (markup languages).
2. Semantics (Meaning):
   - Definitions, relationship and property information for data needs to be formally defined and made available over the internet.

1. Formal Structure (XML, XSD & XPath, XQuery)

- **XML** it is a markup language (like HTML), which allows us to create our own tags, and use those tags to organize, and structure text data.
- **XSD** (XML-Schema) provides a means for defining the structure, content and semantics of an entire XML document.
- **XPATH** and **XQUERY** are languages that allow us to treat XML documents as hierarchical databases and query these databases for subsets of information.

Example HTML vs. XML (XHTML)

HTML:

```
<b>Contact Us:</b>
100 Riverside Parkway<br />
Suite 123<br />
Fredericksburg, VA 22406<br />
```

XML (XHTML):

```
<b>Contact Us:</b>
<address type="mailing">
    <street>100 Riverside Parkway<br />
    Suite 123</street><br />
        <city>Fredericksburg</city>, <state>VA</state>
<zip>22406</zip><br />
</address>
```

Example XSD

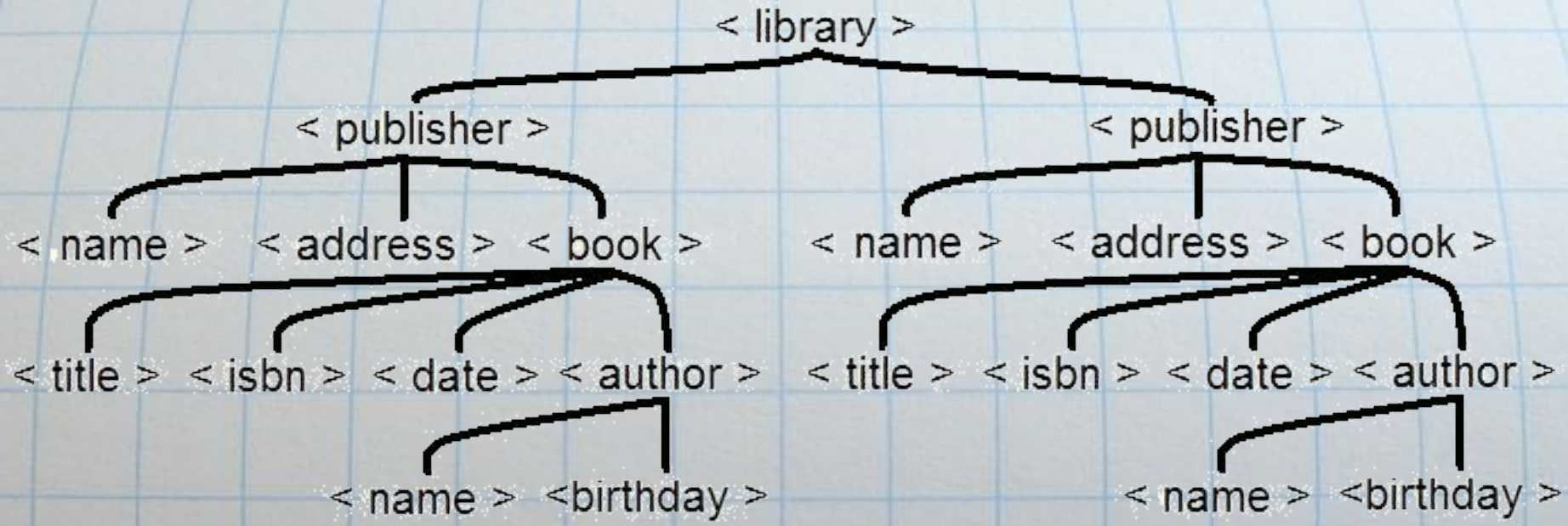
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.
org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:attribute name="type" />
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="City" type="xs:string" />
        <xs:element name="State" type="xs:string" minOccurs="1" maxOccurs="1" />
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="AL" />
                                                    .....
              <xs:enumeration value="WY" />
            </xs:restriction>
        <xs:element name="Zip" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Data Trees (1)

(An example XML library document, "library.xml")

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="library.xsl" ?>
<library name="Example">
    <publisher>
        <name>Ebury</name>
        <address>20 V. Bridge Road, London</address>
        <book>
            <title>The Science of Discworld</title>
            <ISBN>0091865158</ISBN>
            <date>1999</date>
            <author>
                <name>Terry Pratchett</name>
                <birthday>28-Apr-48</birthday>
            </author>
        </book>
    </publisher>
    <publisher>
        <name>O'Reilly Media, Inc.</name>
.....
```

XML Data Trees (2)



Examples XPath & XQuery

XML documents are hierarchical databases (like file systems, phone books) and can be queried using languages like XPATH and XQuery.

XPath expression	Interprets as.
/library/publisher/book/title	"Select all title nodes"
/library/publisher[2]/book[1]	"1st book from 2nd publisher"

XQuery adds FLWOR ("For, Let, Where, Order by, Return").

XQuery Statement:

```
for $x in doc("library.xml")//author
where $x/birthday>1-Jan-20
order by $x/name
return $x/name
```

XQuery Result:

```
<name>Ellen Siever</name>
<name>Terry Pratchett</name>
```

1. Formal Structure - Summary

XML (XHTML, XSD)

- Text markup languages used to present information in a structured way.
- XML documents are easy to transfer over the Internet.
- Allow very simple parsing agents to locate specific pieces of information.

XPATH, XSLT, XQUERY

- Query languages that allow us to treat XML documents like databases.
- Allow us to manipulate the "document tree".

2. Semantics (Meaning)

- XML provides us with a structure to use to encode data in a format that is machine processable.
- But tags do not by themselves provide a definition of the text information that they contain.

Example: Find all the "people" in the library.xml file.

What is needed:

1. A way to provide definitions and relationship information about an object that we can refer to from anywhere on the internet.
2. A way to formalize definition and relationship information so that we can communicate that information to other systems which may use a different vocabulary.

RDF (web accessible definitions)

- RDF (Resource Description Framework) is a language for describing information about resources using the WWW.
- RDF uses Internationalized Resource Identifiers (IRIs) to describe objects in terms of simple properties, property values, and relationships.
- At it's heart RDF is composed of triples which specify a subject, predicate and object.

Example (in abbreviated notation):

<http://www.example.org/example#Pratchett>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.example.org/example#Human>

RDF cont.

- Sets of RDF tuples form directed graphs, where subjects and objects form the nodes of the graph and the predicates become the directed edges.

- RDF has a formal XML-Schema so RDF sets can be expressed in XML format and operated on with XML tools.

- RDF sets can be embedded in XHTML, associated with existing databases, or used as stand-alone databases.

Ontologies (relationships and meaning)

- RDF provides a formal method for attaching structured and reusable information to data objects, but by itself does not address the underlying problem of understanding the semantics, or meaning, of the terms used.

- In Computer Science in simple terms, an ontology is a set of terms and rules used together to express a knowledge base.
- An ontology describes a domain, and all the possible conditions (states) within that domain.
- A knowledge base (constructed using the ontology) describes a particular state of affairs within that domain.

Ontologies continued

Suppose we would like to create an ontology for libraries. To do that we will need to be able to express certain facts:

1. "people" is another term for "human"
2. authors, co-authors and editors are types of humans.
3. authors and co-authors create written works.

These facts are all part of the ontology that we wish to create.

The contents of our library.xml file, expressed using this ontology would then become our knowledge base.

Requirements for an Ontology

Ontologies are created using ontology languages.

An ontology language, to be useful and effective, must enable us to do several things:

1. It must give us a way to express the current facts of our ontology, as well as add additional facts quickly and easily (books are types of written works).
2. Our ontology language must allow us to reason within those facts to make new discoveries (authors are people).
3. Our ontology must allow us to communicate our own personal knowledge base to other library owners who also use this ontology.

RDF-Schema

- RDF-Schema (RDF vocabulary description language) extends basic RDF into a rudimentary ontology language.
- Adds IRI resources including: rdfs:Class, rdfs:subClassof, rdfs:subPropertyOf, rdfs:domain and rdfs:range (rdfs = www.w3.org/2000/01/rdf-schema#).
- Using these properties we could define some of the facts of our library ontology.

Author	rdfs:subClassOf	Human
creatorOf	rdfs:domain	Human
creatorOf	rdfs:range	WrittenWork

RDF-Schema Limitations

RDF-Schema allow us to create and define: classes, properties, class hierarchies, property hierarchies, domain and range restrictions.

However RDF-Schema cannot express:

- Non-Binary relations
- Property Characteristics (transitive, symmetric, inverse)
- Cardinality restraints (Human may have at most 1 name)
- Disjointness axioms (Human cannot be a Man and Woman)
- Granular range restrictions (property hasName has range xsd:string)
- Complex concept descriptions (Human is defined by both Man and Woman)

OWL (Web Ontology Language)

The family of OWL ontology languages (OWL-Lite, OWL-DL, OWL Full, as well as OWL 2:

- extend the capabilities of RDF-Schema.
- can be expressed using XML and RDF notation.
- are based on Description Logic (DL) languages which in turn are a subset of first order logic (FOL) languages.

Query answering in a OWL language (as well as in a DL) is comparable to theorem proving in an FOL.

Powerful reasoning engines for DLs exist and most work by implementing the analytic tableau method.

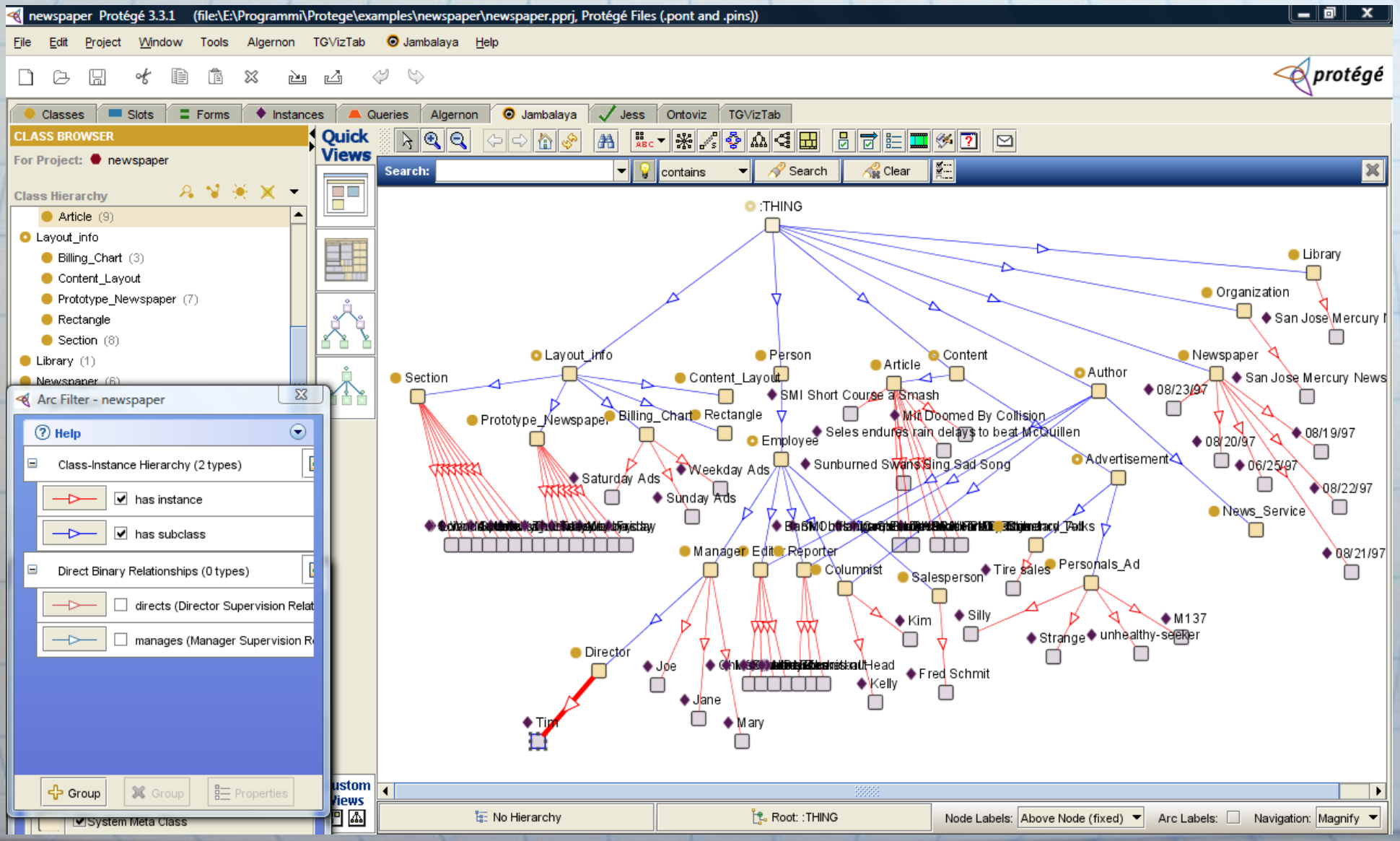
Description Languages

DLs come in a variety of 'flavors' and there is an informal naming convention that uses letters to roughly describe what logical operators are allowed. As an example OWL-DL can be described as a SHOIN(D) DL.

Class : Description	Logical Properties	OWL representations
S : Basic Operators	AND, OR, NOT, ∃, ∀, transitivity (properties)	owl:intersectionOf, owl:unionOf, owl:complementOf, owl:someValuesFrom, owl:allValuesFrom
H : Hierarchies	subproperty, subclass	owl:subClassOf, owl:subPropertyOf
O : nOminals	class extentions	owl:oneOf, owl:hasValue
I : Inverse	inverse properties	owl:inverseOf
N : Numbering	Cardinality Restrictions	owl:Cardinality, owl:MaxCardinality
D : Datatypes	data values or data types	string, integer

Ontology Managers

Ontology Managers (Protege, Jena) facilitate the creation and reuse of Ontologies.



Ontology Enabled Auction System

Juha Puustjarvi created a model of an auction system using a shared OWL ontology to represent the negotiation protocol. System relies on WS, SOAP and BPEL.

SOAP (Simple Object Access Protocol)
Uses HTTP and RPC protocols to transfer messages.

SOAP message example:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <! -- Header tags -->
   </soap:Header>
   <soap:Body>
      <!-- Application data -->
   </soap:Body>
</soap:Envelope>
```

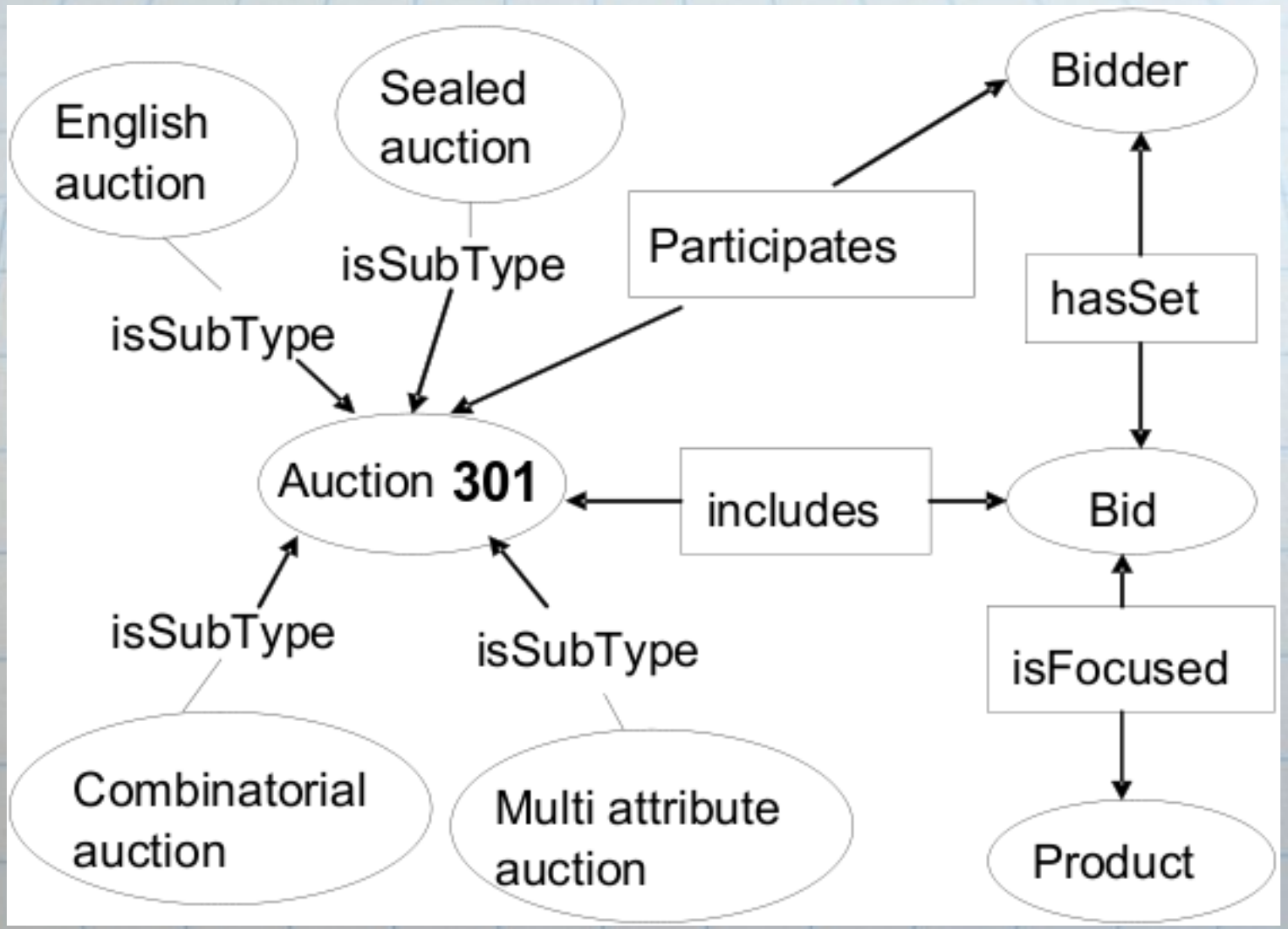
WS-BPEL (Business Process Execution Language)

Every BPEL process is defined by a valid BPEL XML document, and that document deals the orchestration of a series of events between a related set of web services.

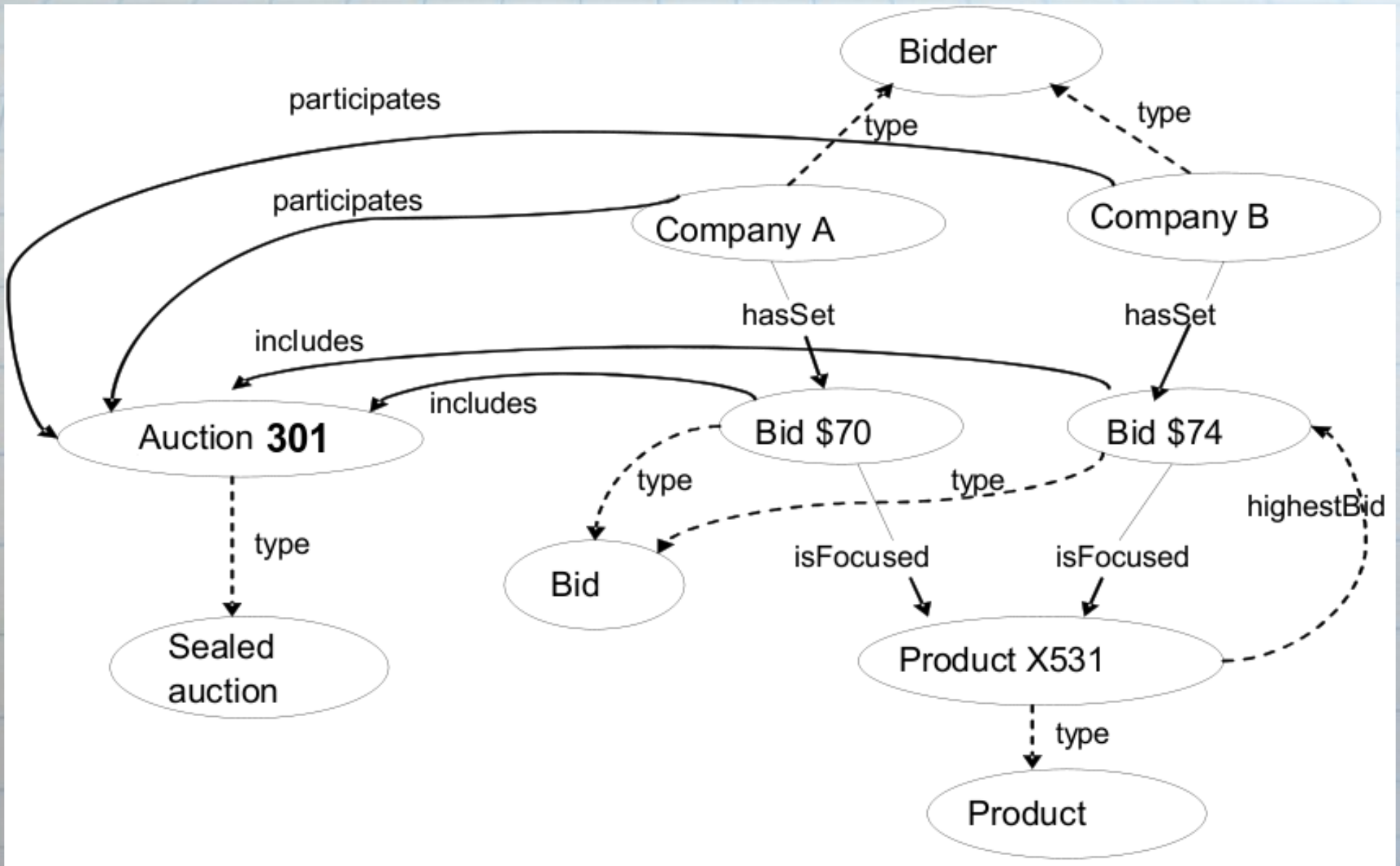
Skeleton of a BPEL document

```
<process name="ProcessName"
  targetNamespace="http://www.xmltc.com/ptc/process/"
     <!-- Additional namespace declarations for each external process involved -->
   ... >
   <partnerLinks>
     <!-- Identify the processes that will be contacted and their WSDL documents -->
  </partnerLinks>
  <variables>
     <!-- Used to store global variables and variables from received messages -->
  </variables>
  <sequence>
     <!-- Defines the flow of the business process, between links and local logic.  -->
  </sequence>
</process>
```

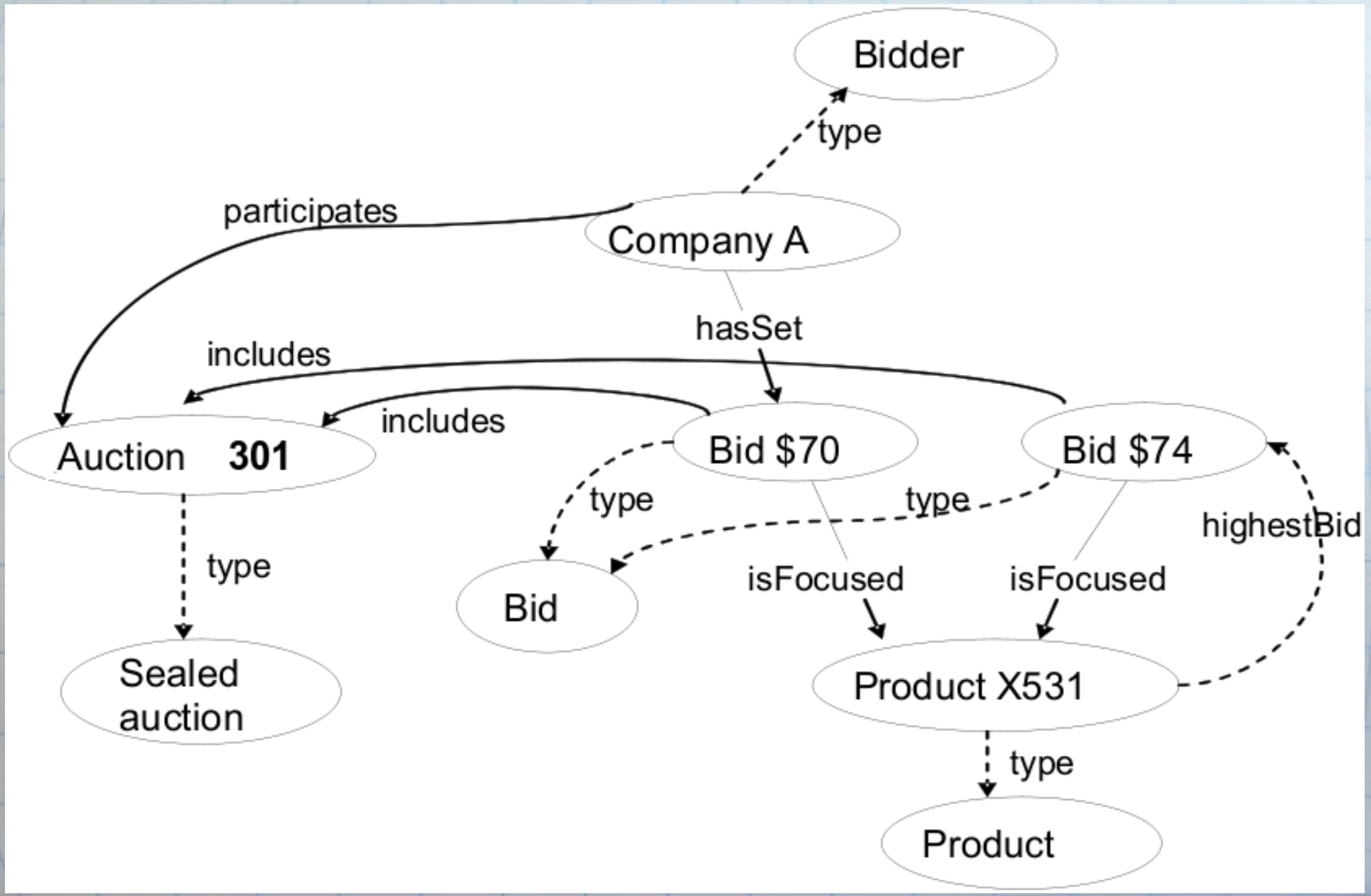
Auction Specific Ontology



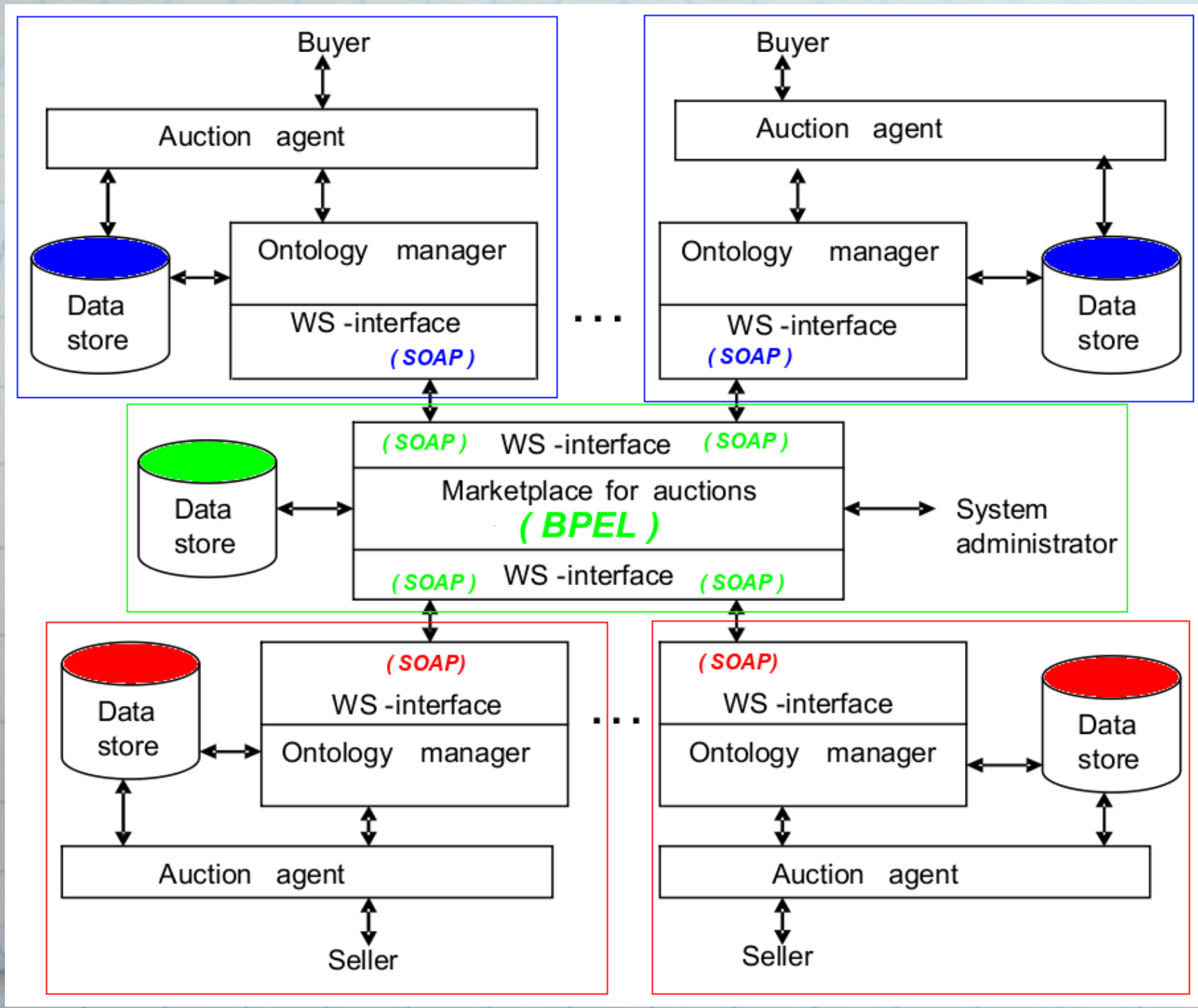
Ontology - Global Instance



Ontology - Local Instance (View by A)

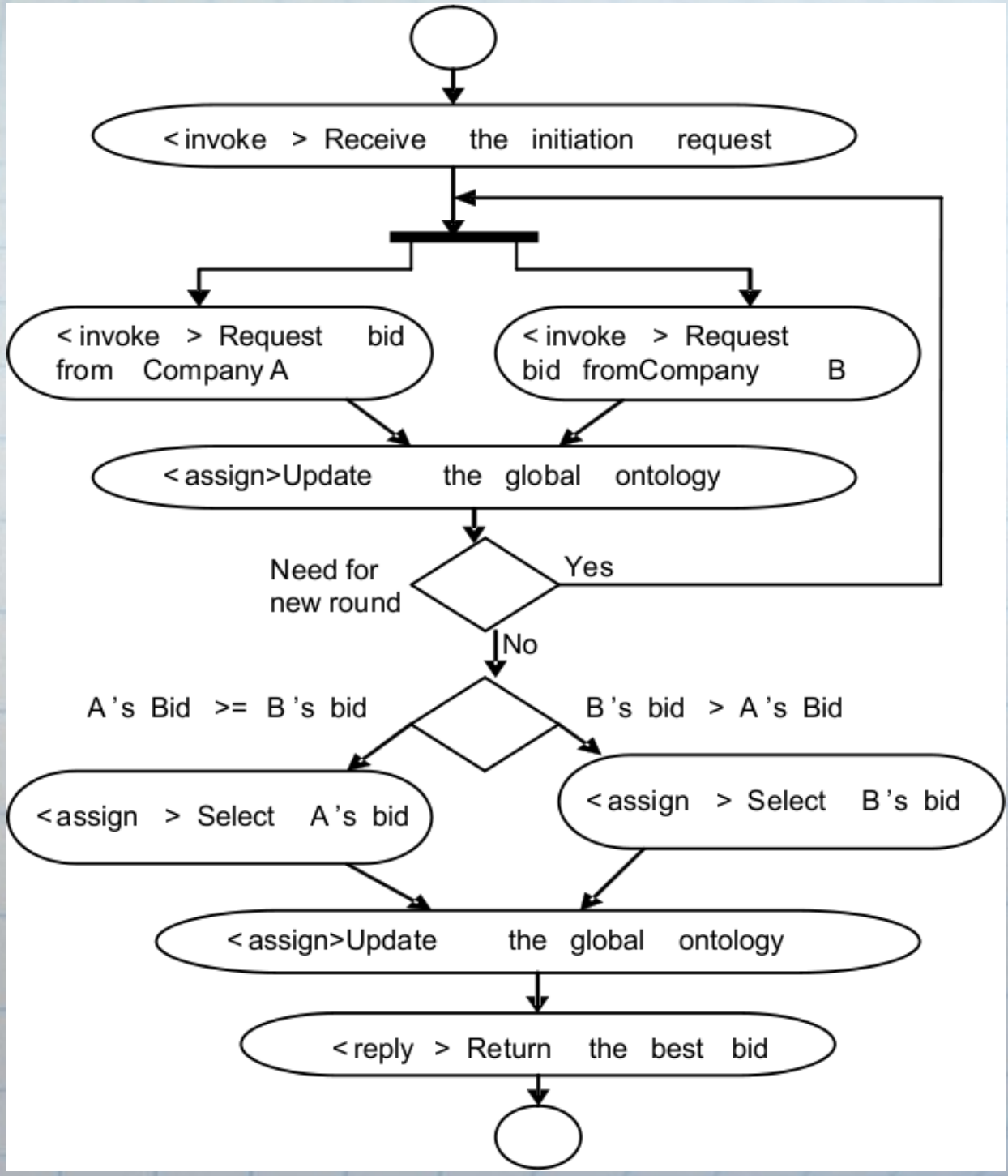


System Architecture



BPEL Diagram

Entire process is managed by the BPEL engine which invokes all responses from participants, and processes those responses in order to update the ontology instance.



Notes on the System.

System is fairly complex and because of the use of SOAP the system (or auction manager) would require WSDL (Web Services Description Language) documents describing the protocol bindings and message formats required to interact with each of the web agents participating in an auction.

Heavy overhead limits the possibilities for it being a truly dynamic system where agents can freely (even randomly) participate in auctions.

This entire system could be redesigned much smaller and simpler, allow for choreopgraphies, and dynamic auction creation by using REST (REpresentational State Transfer).

Ontology Based Auction System

Ontologies created in OWL and DAML + OIL used to:

- Describe the negotiation domain.
- Model the negotiation process.
- Provide a single shared view of the objects of the domain.

Heart of paper is about how agents can use the ontologies to try and evolve optimal negotiation strategies. Specifically, the chosen auction ontology is used as input to an algorithm used by agents to decide on an opening strategy and a reinforcement learning algorithm to use to tune that strategy.

System Architecture

- Upper-level parts of ontology constructed in Protege and created by by reusing parts of existing ontologies (Example: REA (Resource, Event, Agent) ontology.

- Execution of an auction is handled by treating the ontology areas that describe negotiation protocols as processes and using PSL (process specification language) JESS (Java Expert System Shell) and JASA (Java Auction Simulator) to execute them.

- A fair amount of logic still had to be hard-coded into the system and into the agents to enable auctions to take place.

Trading Agents

- Determining the "best" strategy to play, based on a description of the game is impossible in non-trivial cases.
- Trading agents in the system were provided with learning algorithms in an attempt to evolve efficient strategies.
- Agents did not always discover the "optimal" strategy for a given mechanism but acquired "reasonably good" strategies in only a few iterations.
- What was discovered was that certain classes of learning algorithms did perform better under identifiable auction parameters.

Agent Strategies

Three broad classes of learning algorithms were identified:

1. Mimicry learning strategies (Cliffs ZIP strategy).
2. Myopic stimuli-response learning algorithms (Roth-Erev algorithm.)
3. MDP (Markoff Decision Problem) procedures (Q-learning algorithm.

MDP strategies proved effective in double-sided mechanisms where quotes were issued at every round.

Notes on the System.

- The negotiation host (auction manager) specifically advertises the URL of the ontology used to govern the marketplace.
- The negotiation host also advertises (via URL) ontology instance information to insure that participants are working from the same view of the domain.
- System could be adapted to be RestFul fairly easy, as it already revolves around exchanging updated onology instances which are defacto state transition records.