Distributed Semantic Web Knowledge Representation and Inferencing

Harold Boley
UNB, Faculty of Computer Science

Keynote at ICDIM 2010
6 July 2010
Update: 3 October 2016
Introduction

- Interdisciplinary approach to the (Social) Semantic Web
  - Computer, Information, and Data Science, AI, Logic, Graph Theory, Linguistics, ...

- Representation & Inferencing Techniques for Distributed (Internet/Web-networked) Knowledge Management, Visualization, Interoperation (e.g., Object-Relational), and Access to (Big) Data

- Applications in eHealth, eLearning, eBusiness, Ecosystem Research, ...
### Three Levels of Knowledge: Visual and Symbolic Representations

<table>
<thead>
<tr>
<th>Knowledge elicitation as gradual formalization</th>
<th>visual</th>
<th>symbolic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>formal</strong></td>
<td>graph theory</td>
<td>predicate logic</td>
</tr>
<tr>
<td><strong>semi-formal</strong></td>
<td>standardized graphics</td>
<td>controlled natural language</td>
</tr>
<tr>
<td><strong>informal</strong></td>
<td>hand drawing</td>
<td>natural language</td>
</tr>
</tbody>
</table>
Three Levels of Knowledge: Described by Formal Metadata

<table>
<thead>
<tr>
<th></th>
<th>visual</th>
<th>symbolic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>formal</strong></td>
<td>graph theory</td>
<td>predicate logic</td>
</tr>
<tr>
<td><strong>semi-formal</strong></td>
<td>standardized graphics</td>
<td>controlled natural language</td>
</tr>
<tr>
<td><strong>informal</strong></td>
<td>hand drawing</td>
<td>natural language</td>
</tr>
</tbody>
</table>
Web as Standard Distributed Knowledge Medium for Collaboration

- Social Semantic Web (Web 3.0, e.g. semantic wikis)
- Social Web (Web 2.0, e.g. wikis for collaboration)
- Semantic Web (formal knowledge)
- Web 1.0 (informal to semi-formal knowledge)
Knowledge and, Specifically, Data have Semantics, Based on Syntax

Knowledge

Semantics

SubClassOf

Data

Syntax

Knowledge subsumes Data by inferring Knowledge (e.g. Data) from other Knowledge (e.g. Data).

Semantics based on Syntax by distinguishing subsets of (‘true’) formulas from the set of all formulas.

Engines compute Inferences

Inferences preserve truth distinction

Via ‘Meaning Function’ (part of Interpretation)
Example: Data (Ground Facts)

Croco(c)
Horse(h)
Mule(m)
Pony(p)

Ground: No variables as arguments
Example: Knowledge (Beyond Data: Implication Rules)

Mule(x) ⇒ Horse(x)
Pony(x) ⇒ Horse(x)
Example: Inference

Pony(\(x) \Rightarrow \text{Horse}(x)\)
Pony(p)

\[\vdash \text{Horse}(p)\]
Example: Syntax

Mule(x) \Rightarrow Horse(x)
Pony(x) \Rightarrow Horse(x)
Croco(c)
Horse(h)
Mule(m)
Pony(p)

\ldots

pred(var) \Rightarrow pred(var)

pred(const)
Example: Semantics (Truth Directly Distinguished)

\[
\text{Pony}(x) \Rightarrow \text{Horse}(x) \quad \text{“Each pony is a horse”}
\]

Mule(m) \quad \text{“m is a mule”}

Pony(p) \quad \text{“p is a pony”}

Asserted by an authority or
Found by a sensor-based IoT system or

3-Oct-16
Example: Semantics (Directly Distinguished, Fully Interpreted)

\[ \text{Pony} \subseteq \text{Horse} \]

- "Each pony is a horse"

- "m is a mule"
- "p is a pony"

Italics font indicates individuals and their (extensional) sets

Asserted by an authority or
Found by a sensor-based IoT system or ...

\[ m \in \text{Mule} \]
\[ p \in \text{Pony} \]
Example: Semantics (Truth Including Inferred)

Pony(x) ⇒ Horse(x)

Mule(m)
Pony(p)Horse(p)
Example: Semantics
(Including Inferred, Fully Interpreted)

\[ \text{Pony} \subseteq \text{Horse} \]

\[ m \in \text{Mule} \]
\[ p \in \text{Pony} \]
\[ p \in \text{Horse} \]
Species of Formal Knowledge on the Web

Making distributed formal knowledge a universal commodity on the Web
Formal Knowledge as Ontologies or Rules

FormalKnowledge

OntologyKnowledge

RuleKnowledge

TaxonomyKnowledge

FactKnowledge/Data

All arrows are understood as labeled **SubClassOf**

Datalog facts with **unary/binary** predicates used for ontology ABoxes
Taxonomy Knowledge: TBox (1)

- Class hierarchies for conceptual classification
- Example: Above classification of FormalKnowledge
- Discover subsumptions/implications for inference; e.g., \( \text{TaxonomyKnowledge} \subseteq \text{RuleKnowledge} \)
  - i.e., \( \text{TaxonomyKnowledge}(x) \Rightarrow \text{RuleKnowledge}(x) \)
- Thus allowing multiple parents (shown above):
  - From trees to Directed Acyclic Graphs (DAGs)
    - Here, taxonomies as ‘intersection’ of ontologies and rules
- Realized several taxonomies in projects, including ‘Computing’ classification in FindXpRT and ‘Tourism’ classification in eTourPlan
Taxonomy Knowledge: TBox (2)

- With the **metamodel** about FormalKnowledge defined, it is instructive to separate the representation method (a taxonomy) from what is represented:
  - Earlier: FormalKnowledge, containing TaxonomyKnowledge
  - Now: A ‘folksonomy’ of Equus, containing Mule

- Structurally a subDAG of the FormalKnowledge taxonomy, but completely different content

- Again discover subsumptions/implications which enable inferences, e.g. about mules as horses; e.g., \( Mule \subseteq Horse \)
  
  i.e., \( Mule(x) \Rightarrow Horse(x) \)

- Thus also allowing **multiple** parents (shown below)

- But ‘commonsense’: Much simplified biologically!

---

Single-premise rules whose predicates have one and the same variable argument
Equi as Donkies or Horses: Visual (DAG)
Equi as Donkies or Horses: Visual (Venn Diagram)
Equi as Donkies or Horses (DAG): ABox Asserting Instances d, e, h, m, p

Equus

Donkey  
\[d\] 

\[\text{Mule}\] 
\[m\]

Horse  
\[e\]

Pony  
\[p\]

‘Populated’ Taxonomy
Equi as Donkies or Horses (Venn): ABox Asserting Instances d, e, h, m, p

Equus

Donkey

Horse

Mule

Pony

d e h m p
### Equi as Donkies or Horses: Symbolic (1)

#### Semantics:
- **Subsumptions**
  - Donkey $\subseteq$ Equus
  - Horse $\subseteq$ Equus
  - Mule $\subseteq$ Donkey
  - Mule $\subseteq$ Horse
  - Pony $\subseteq$ Horse

#### Rule syntax:
- **Implications**
  - Donkey$(x)$ $\Rightarrow$ Equus$(x)$
  - Horse$(x)$ $\Rightarrow$ Equus$(x)$
  - Mule$(x)$ $\Rightarrow$ Donkey$(x)$
  - Mule$(x)$ $\Rightarrow$ Horse$(x)$
  - Pony$(x)$ $\Rightarrow$ Horse$(x)$

*Italics font indicates (extensional) sets*

*Normal font indicates (intensional) predicates*
Equi as Donkies or Horses: Symbolic (2)

Ontology syntax:
Classifications

Donkey \subseteq Equus
Horse \subseteq Equus
Mule \subseteq Donkey
Mule \subseteq Horse
Pony \subseteq Horse

Rule syntax:
Implications

Donkey(x) \implies Equus(x)
Horse(x) \implies Equus(x)
Mule(x) \implies Donkey(x)
Mule(x) \implies Horse(x)
Pony(x) \implies Horse(x)

Normal font indicates (intensional) classes
Normal font indicates (intensional) predicates
Equi as Donkies or Horses: Symbolic (3)

Logic rule syntax:

Backward implications
Equus(x) ↔ Donkey(x)
Equus(x) ↔ Horse(x)
Donkey(x) ↔ Mule(x)
Horse(x) ↔ Mule(x)
Horse(x) ↔ Pony(x)

Forward implications
Donkey(x) ⇒ Equus(x)
Horse(x) ⇒ Equus(x)
Mule(x) ⇒ Donkey(x)
Mule(x) ⇒ Horse(x)
Pony(x) ⇒ Horse(x)
Equi as Donkies or Horses: Symbolic (4)

**Prolog rule syntax:**
Backward implications

- `equus(X) :- donkey(X).`
- `equus(X) :- horse(X).`
- `donkey(X) :- mule(X).`
- `horse(X) :- mule(X).`
- `horse(X) :- pony(X).`

**Logic rule syntax:**
Forward implications

- `Donkey(x) \implies \text{Equus}(x)`
- `Horse(x) \implies \text{Equus}(x)`
- `Mule(x) \implies \text{Donkey}(x)`
- `Mule(x) \implies \text{Horse}(x)`
- `Pony(x) \implies \text{Horse}(x)`

*Upper-case (first) letter indicates (∀) variables; so predicates are lower-cased*

*Letters x, y, and z often used as (∀) variables*
Inference: Modus Ponens, Bottom-Up (Two ‘Sequential’ Applications)

TBox rules

\[ \text{equus}(X) \dashv \vdash \text{horse}(X). \]
\[ \text{horse}(X) \dashv \vdash \text{pony}(X). \]

ABox instance/fact (datum)

\text{pony}(p).

Bottom-up (\( \Rightarrow \)) derivation, i.e. forward-chaining, realizes inheritance (via ‘\( \dashv \vdash \)’ transitivity)

\[ \text{pony}(p) \Rightarrow \text{horse}(p) \Rightarrow \text{equus}(p) \]

fact-to-fact
Inference: Modus Ponens, Top-Down (Two ‘Sequential’ Applications)

TBox rules

\[
equus(X) :\rightarrow \text{horse}(X).
\]

\[
\text{horse}(X) :\rightarrow \text{pony}(X).
\]

ABox instance/fact (datum)

\[
\text{pony}(p).
\]

Top-down (\(\Rightarrow\)) reduction, i.e. backward-chaining, realizes inheritance (via ‘:-’ transitivity)

\[
equus(p) \Rightarrow \text{horse}(p) \Rightarrow \text{pony}(p) \Rightarrow \text{true}
\]

query-to-query

\[
equus(W) \Rightarrow \text{horse}(W) \Rightarrow \text{pony}(W) \Rightarrow \text{true}, \quad W=p
\]
Inference: Modus Ponens, Bottom-Up (Two ‘Parallel’ Applications)

TBox rules

\[ \text{equus}(X) \leftarrow \text{donkey}(X). \]
\[ \text{equus}(X) \leftarrow \text{horse}(X). \]

ABox instances/facts (data)

\text{donkey}(d).
\text{horse}(h).

Bottom-up (\(\Rightarrow\)) derivation/inheritance

\text{donkey}(d) \Rightarrow \text{equus}(d)
\text{horse}(h) \Rightarrow \text{equus}(h)
Inference: Modus Ponens, Top-Down (Two ‘Parallel’ Applications)

TBox rules
equus(X) :- donkey(X).
equus(X) :- horse(X).

ABox instances/facts (data)
donkey(d).
horse(h).

Top-down (⇒) reduction/inheritance
equus(d) ⇒ donkey(d) ⇒ true
equus(h) ⇒ horse(h) ⇒ true
equus(W) ⇒ donkey(W) ⇒ true, W=d
⇒ horse(W) ⇒ true, W=h
Ontology Knowledge

- Ontologies extend taxonomies by property hierarchies, \( \forall/\exists \)-restricted properties, etc. of description logics
- Int'l standards:
  - ISO: Common Logic (CL 2, incl. CGs: Conceptual Graphs)
  - OMG: Ontology Definition Metamodel (ODM 1.1)
  - W3C: Web Ontology Language (OWL 2)
- **Datalog\(^+/-\)** and **Deliberation RuleML 1.02** allow to represent ontologies as (existential) rules, e.g. for Rule-Based Data Access and \( \Delta \)Forest (RBDA)
- Target representation for knowledge discovery (e.g. business intelligence/analytics) from instances
  - Background knowledge for further discovery
Fact Knowledge

- Facts (data) can be asserted in two paradigms:

<table>
<thead>
<tr>
<th>Positional</th>
<th>Slotted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational-table (SQL) rows</td>
<td>Object-centered instances</td>
</tr>
<tr>
<td>(column headers = signatures)</td>
<td>(o-c directed labeled graphs)</td>
</tr>
<tr>
<td>XML elements</td>
<td>RDF triples / XML attributes</td>
</tr>
<tr>
<td>n-ary predicates (Prolog)</td>
<td>AI frames (F-logic)</td>
</tr>
</tbody>
</table>

- **POSL** and **OO RuleML** combine these paradigms; cross-paradigm translators enable interoperation
  - Used in projects **SymposiumPlanner**, **WellnessRules2**, **PatientSupporter**, and **EnviroPlanner**

- The paradigms and translators have been lifted to object-relational rules, as in **PSOA RuleML**
Rule Knowledge

- Rules generalize facts by making them conditional on other facts (often via chaining through further rules)
- Rules generalize taxonomies via multiple premises, n-ary predicates, structured arguments, etc.
- Two uses of rules – *top-down* (backward-chaining) and *bottom-up* (forward-chaining) – represented only once
- To avoid $n^2$–$n$ pairwise translators:
  - Int'l standards with $2n$–2 *in-and-out* translators:
    - RuleML: Rule Markup Language (work with ISO, OMG, W3C, OASIS)
      - Deliberation RuleML 1.02 / Reaction RuleML 1.0 released as de facto standards
    - ISO: Common Logic (incl. CGs & KIF: Knowledge Interchange Format)
      - Collaboration on Relax NG schemas for XCL 2 / CL RuleML
    - OMG: Production Rules Representation (PRR), SBVR, and API4KP
    - W3C: Rule Interchange Format (RIF)
      - Gave rise to open-source and commercial RIF implementations
    - OASIS: LegalRuleML
- Target representation and background knowledge for discovery from facts: Inductive Programming
Ontology-Rule Synthesis: Hybrid and Homogeneous

- **Hybrid combinations**
  - Reuse existing ontology and rule standards
  - Allow rule conditions to refer to ontologies
  - Explored in projects:
    - **Object Oriented RuleML**: RDF Schema taxonomies
    - **Datalog\textsuperscript{DL}**: Datalog with Description Logics

- **Homogeneous integrations**
  - Merge ontologies and rules into a single representation
  - Explored in projects:
    - **ALC\textsuperscript{u}P**: ALC/Datalog merger with safeness condition
    - Semantic Web Rule Language: OWL/RuleML merger as W3C Member Submission (http://scholar.google.ca/scholar?q=SWRL)
    - **PSOA** (Positional-Slotted, Object-Applicative) RuleML semantics allows taxonomic subclass relationships
RuleML Tools from the Semantic Technology Stack

Foundational and extended RuleML technology available online
Rule Responder: Reference Architecture for Distributed Query Engines

- Enables expert finding and query-based knowledge discovery in distributed virtual organizations
- Queries and answers exchanged in RuleML/XML
- Supported rule engines (int’l collaboration): Prova, OO jDREW, Euler, and DR-Device
- Based on the Mule Enterprise Service Bus
- Instantiated, e.g., in deployed SymposiumPlanner and prototyped WellnessRules2 / PatientSupporter
- Foundation for Master’s projects on EnviroPlanner and SP-2012 at UNB. Also used in PhD projects in Fredericton, Berlin, Vienna, and Thessaloniki
Rule Responder is a tool for creating virtual organizations as multi-agent systems that support collaborative teams on the Semantic Web. It provides the infrastructure for rule-based collaboration between the distributed members of such a virtual organization. Human members of an organization are assisted by semi-autonomous rule-based agents, which use Semantic Web rules to describe aspects of their owners' derivation and reaction logic.

Each Rule Responder instantiation employs three classes of agents, an Organizational Agent (OA), Personal Agents (PAs), and External Agents (EAs). The OA represents goals and strategies shared by its virtual organization as a whole, using a global rule base that describes its policies, regulations, opportunities, etc. Each PA assists a single person of the organization, (semi-autonomously) acting on his/her behalf by using a local knowledge base of derivation rules defined by the person. Each EA uses a Web (HTTP) interface, accepting queries from users and passing them to the OA.

The OA employs an OWL ontology as a "role assignment matrix" to find a PA that can handle an incoming query. The OA uses reaction rules to send the query to this PA, receive its answer(s), do validation(s), and send answer(s) back to the EA. For example, the Rule Responder instantiation of [http://ruleml.org/WellnessRules/RuleResponder.html WellnessRules] answers queries about planned activities of participants in a wellness organization.

[[Category:Semantic agent system]]
[[Category:Reasoner]]
Rule Responder is a tool for creating virtual organizations as multi-agent systems that support collaborative teams on the Semantic Web. It provides the infrastructure for rule-based collaboration between the distributed members of such a virtual organization. Human members of an organization are assisted by semi-autonomous rule-based agents, which use Semantic Web rules to describe aspects of their owners' derivation and reaction logic.

Each Rule Responder instantiation employs three classes of agents, an Organizational Agent (OA), Personal Agents (PAs), and External Agents (EAs). The OA represents goals and strategies shared by its virtual organization as a whole, using a global rule base that describes its policies, regulations, opportunities, etc. Each PA assists a single person of the organization, (semi-autonomously) acting on his/her behalf by using a local knowledge base of derivation rules defined by the person. Each EA uses a Web (HTTP) interface, accepting queries from users and passing them to the OA.

The OA employs an OWL ontology as a "role assignment matrix" to find a PA that can handle an incoming query. The OA uses reaction rules to send the query to this PA, receive its answer(s), do validation(s), and send answer(s) back to the EA. For example, the Rule Responder instantiation of WellnessRules answers queries about planned activities of participants in a wellness organization.
Conclusion

- Conceived a joint semantics of objects+relations and ontologies+rules for distributed knowledge querying
  - Developed standard languages, compatible engines, and reference architectures (visualized with Grailog)
- Used to study expert knowledge and communication topologies of virtual organizations
  - Gradual formalization as distributed knowledge and agent-mediated communication (cf. Rule Responder)
- Applied to knowledge representation and inferencing on the Social Semantic Web
  - Use cases in symposium organization, wellness groups, patient support, and environmental querying