Object Oriented Knowledge Bases in Logic Programming

Vinay K. Chaudhri
Stijn Heymans
Michael Weseel
Son Cao Tran
Acknowledgment

- This work is funded by Vulcan Inc. and SRI International
  - See http://www.vulcan.com and www.projecthalo.com

- AURA/Inquire Development Team
  - Eva Banik, Peter Clark, Roger Corman, Nikhil Dinesh, Debbie Frazier, Stijn Heymans, Sue Hinojoza, David Margolies, Adam Overholtzer, Aaron Spaulding, Ethan Stone, William Webb, Michael Wessel and Neil Yorke-Smith
  - Ashutosh Pande, Naveen Sharma, Rahul Katragadda, Umangi Oza
Outline

- Need
- KR in OOKB
- Decidability results
Need for OO Representation

Intricately connected entities

Process flow diagrams
Abstract characterization of the problem

- Naturally existential rules
- If you write a rule describing an object structure and create the following graph
  - Each variable in the body of the rule is a node
  - Each predicate in the body of the rule is a labeled edge between the variables participating in the predicate
  - The resulting graph is not tree-structured
- Objects are so intricate that it is not possible to write a single rule
  - You need a mechanism of abstraction to create the graph structures separately and then let them refer to each other
Need

Project Halo: A reasoning system to answer a wide variety of questions on science topics

Inquire
Electronic book for biology students
http://www.aaaivideos.org/2012/inquire_intelligent_textbook/

AURA
A knowledge authoring environment for domain experts

AURA is based on KM Description Logic + Rules

KB Bio 101 is a significant KB
Need

- We want to share this KB with other reasoners
  - To leverage the recent advances in description logic and logic programming systems
  - To support the community in AI research by advancing the state of the art in reasoning systems

- Problems faced
  - Rule languages do not directly support all modeling primitives
  - DL systems cannot handle graph structured rules
  - Lack of appreciation for why reasoning is hard
**KR in OOKB**

- All the favorite features
  - Classes
    - Necessary and sufficient conditions
    - Disjoint-ness
    - Multiple Inheritance
  - Relations and Property Values
    - domain, range
    - inverse relations
    - transitivity
    - Relation hierarchy
    - Relation composition
    - qualified number restrictions
    - Nominals

Every Cell is a Living Entity and has a Ribosome and a Chromosome part.

∀ x : Cell(x)
→ ∃ y₁, y₂: LivingEntity(x) ∧ hasPart(x, y₁)
∧ hasPart(x, y₂) ∧ Ribosome(y₁)
∧ Chromosome(y₂)

Cell ⊑ Living-Entity ⊓ (□ hasPart.Ribosome) ⊓ (□ hasPart.Chromosome)
Unique Features in relation to DLs (and FDNC, Datalog±, ASPfs)

- Graph structured descriptions

Every Eukaryotic Cell is a Cell and has parts Eukaryotic Chromosome, Nucleus and a Ribosome such that the Eukaryotic Chromosome is inside the Nucleus.

\[
\forall x: \text{EukaryoticCell}(x) \rightarrow \\
\exists y_1, y_2, y_3: \text{Cell}(x) \land \\
\text{hasPart}(x, y_1) \land \text{hasPart}(x, y_2) \land \text{hasPart}(x, y_3) \land \\
\text{isInside}(y_2, y_3) \land \text{Ribosome}(y_1) \land \text{EukaryoticChromosome}(y_2) \land \\
\text{Nucleus}(y_3)
\]

The knowledge shown in red is not expressible in known decidable description logics such as OWL 2.

This can be captured in Rule Languages.
Unique Features in relation to DLs

- Inherit and Specialize

In the Eukaryotic Cell
Ribosome was inherited from Cell
Chromosome was inherited from Cell and specialized to Eukaryotic Chromosome
KR in OOKB

- Unique Features in relation to DLs
  - Inherit and Specialize

In the Eukaryotic Cell

Ribosome was inherited from Cell
Chromosome was inherited from Cell and specialized to Eukaryotic Chromosome

∀ x : Cell(x) → hasPart(x, f_{Cell}#1(x)) ∧ hasPart(x, f_{Cell}#2(x)) ∧ Ribosome(f_{Cell}#1(x)) ∧ Chromosome(f_{Cell}#2(x))

∀ x : ECell(x) → hasPart(x, f_{ECell}#1(x)) ∧ hasPart(x, f_{ECell}#2(x)) ∧ hasPart(x, f_{ECell}#3(x)) ∧ isInside(f_{ECell}#2(x), f_{ECell}#3(x)) ∧ Ribosome(f_{ECell}#1(x)) ∧ EukaryoticChromosome(f_{ECell}#2(x)) ∧ Nucleus(f_{ECell}#3(x))
• Unique Features in relation to DLs
  • Inherit and Specialize

In the Eukaryotic Cell:
Ribosome was inherited from Cell.
Chromosome was inherited from Cell and specialized to Eukaryotic Chromosome.

∀ x : Cell(x) →
  hasPart(x, f_{Cell}#1(x)) ∧ hasPart(x, f_{Cell}#2(x))
  ∧ Ribosome(f_{Cell}#1(x))
  ∧ Chromosome(f_{Cell}#2(x))

∀ x : ECell(x) →
  hasPart(x, f_{ECell}#1(x)) ∧ hasPart(x, f_{ECell}#2(x))
  hasPart(x, f_{ECell}#3(x)) ∧ isInside(f_{ECell}#2(x), f_{ECell}#3(x))
  ∧ Ribosome(f_{ECell}#1(x))
  ∧ EukaryoticChromosome(f_{ECell}#2(x))
  ∧ Nucleus(f_{ECell}#3(x))

(f_{Cell}#1(x)) = (f_{ECell}#1(x))
(f_{Cell}#2(x)) = (f_{ECell}#2(x))

(f_{ECell}#2(x)) = (f_{ECell}#2(x))
Note on default negation

- The current representation in OOKB does not use default negation
  - We could have used FOL, WFS, or datalog with function symbols, and true negation to capture the knowledge content
    - Default negation needed for formalizing the queries
- Using a language with a default negation is a big plus for future work
Computational Properties of OOKB

- Computational properties
  - Reasoning with OOKB in general, un-decidable
  - There are, however, some decidable fragments that introduce guardedness and acyclic structure in the KB
- To consider the computational properties in detail, let us specify the full language and consider its sub-languages.
### Sub-languages of OOKB

<table>
<thead>
<tr>
<th>Axiom type</th>
<th>Referent</th>
<th>Axiom Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxonomic</td>
<td>T</td>
<td>class(c), individual(o), subclass_of(c,c’), instance_of (i,c)</td>
</tr>
<tr>
<td>Relation Hierarchy</td>
<td>H</td>
<td>subrelation_of(s₁,s₂)</td>
</tr>
<tr>
<td>Inverse relations</td>
<td>I</td>
<td>inverse(s₁,s₂)</td>
</tr>
<tr>
<td>Existential rules</td>
<td>E</td>
<td>See next slide</td>
</tr>
<tr>
<td>Sufficient conditions</td>
<td>P</td>
<td>See next slide</td>
</tr>
<tr>
<td>Qualified number constraints</td>
<td>Q</td>
<td>constraint(exact,f(X),s,n) ← instance_of(X,c)</td>
</tr>
<tr>
<td>Domain/range</td>
<td>C</td>
<td>domain(s,r), range(s,d)</td>
</tr>
<tr>
<td>Disjointness</td>
<td>J</td>
<td>disjoint(c,c’)</td>
</tr>
<tr>
<td>Relation composition</td>
<td>(o)</td>
<td>compose(s₁,s₂,s₃)</td>
</tr>
<tr>
<td>Equality/Inequality</td>
<td></td>
<td>eq(t₁,t₂), neq(t₁,t₂)</td>
</tr>
</tbody>
</table>
∀ x : ECell(x) →
  hasPart(x, f_{ECell#1}(x)) ∧ hasPart(x, f_{ECell#2}(x))
  hasPart(x, f_{ECell#3}(x)) ∧ inside(f_{ECell#2}(x), f_{ECell#3}(x))
  ∧ Ribosome(f_{ECell#1}(x))
  ∧ EukaryoticChromosome(f_{ECell#2}(x))
  ∧ Nucleus(f_{ECell#3}(x))

2{value(hasPart, x, f_{ECell#1}(x)), instance_of(f_{ECell#1}(x), Ribosome)}2 ← ECell(x)
2{value(hasPart, x, f_{ECell#2}(x)), instance_of(f_{ECell#2}(x), EukaryoticChromosome)}2 ← ECell(x)
3{value(hasPart, f_{ECell#2}(x), f_{ECell#3}(x)), value(hasPart, x, f_{ECell#3}(x)), instance_of(f_{ECell#3}(x), Nucleus)}3 ← ECell(x)
Sufficient Conditions

- Sufficient conditions define class membership criterion based on relation values and constraints

\[
\text{instance_of}(X, C) \leftarrow \text{Body}(\overline{X})
\]

If a cell has part nucleus, it is a eukaryotic cell

\[
\text{instance_of}(X, ECell) \leftarrow \\
\text{instance_of}(x, \text{Nuclues}), \\
\text{instance_of}(y, \text{Cell}) \\
\text{value}(\text{has_part}, y, x)
\]
<table>
<thead>
<tr>
<th>Axiom type</th>
<th>Referent</th>
<th>Axiom Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxonomic</td>
<td>T</td>
<td>class(c), individual(o), subclass_of(c,c’), instance_of (i,c)</td>
</tr>
<tr>
<td>Relation Hierarchy</td>
<td>H</td>
<td>subrelation_of(s₁,s₂)</td>
</tr>
<tr>
<td>Inverse relations</td>
<td>I</td>
<td>inverse(s₁,s₂)</td>
</tr>
<tr>
<td>Existential rules</td>
<td>E</td>
<td>Already explained</td>
</tr>
<tr>
<td>Sufficient conditions</td>
<td>P</td>
<td>Already explained</td>
</tr>
<tr>
<td>Qualified number constraints</td>
<td>Q</td>
<td>constraint(exact,f(X),s,n) ← instance_of(X,c)</td>
</tr>
<tr>
<td>Domain/range</td>
<td>C</td>
<td>domain(s,r), range(s,d)</td>
</tr>
<tr>
<td>Disjointness</td>
<td>J</td>
<td>disjoint(c,c’)</td>
</tr>
<tr>
<td>Relation composition</td>
<td>(o)</td>
<td>compose(s₁,s₂,s₃)</td>
</tr>
<tr>
<td>Equality/Inequality</td>
<td></td>
<td>eq(t₁,t₂), neq(t₁,t₂)</td>
</tr>
</tbody>
</table>

A THIEP KB contains taxonomic, relation hierarchy, inverse relations, existential rules, and sufficient properties.
Reasoning problems

- **Consistency (C₁):** Whether the KB has an answer set
- **Concept Satisfiability (C₂):** Given an instance I and a class C, determine if $KB \cup \text{instance}_of(I, C)$ has an answer set
- **Entailment (E):** Given a ground atom $a$, determine if $KB \models a$
- **Question answering (QA):** Determine if $KB \models q(\delta)$ where $q$ is a query atom, $\{X₁,…,Xₙ\}$ are distinct variables occurring in it, and $\delta = \{X₁/a₁,…,Xₙ/aₙ\}$ are all ground substitutions.
Prior Work

- Deciding whether a logic program has an answer set is un-decidable in full generality
  - C1 and C2 are $\Sigma_1^1$ complete and E is un-decidable
    - (Marek et. al. 1992) (Schlipf 1995)
  - The QA task is not usually considered in ASPs
Restriction 1: No constraints

Result 1:

C1 and C2 are decidable for THIEP(o) knowledge bases

The only way a KB can be inconsistent, and hence, not have an answer set is by violating:

- domain and range constraints (C)
- disjointness (J)
- Qualified number constraints (Q)

Since the answer set of THIEP(o) can be infinite, E, and QA are problematic
Restriction 2: Guardedness

Result 2:

E is decidable for guarded and consistent THIEPQCJ(o) KBs

Immediate consequence operator results in larger terms (in size) if the terms appearing in the head of rules are more complex than the terms in the body.

This is already the case for all rules in any TKB except possibly for sufficient conditions where the head contains a variable but the body possibly more complex terms.

If the sufficient conditions are restricted to be guarded, E becomes decidable.

Guardedness does not help with QA because the Herbrand Universe can still be infinite.
Restriction 3: Acyclicity

Result 3:

C1, C2, E and QA are decidable for acyclic THIEQCJ(o) KBs

We say that a class c1 refers to class c2, or c1 < c2 if

- KB contains a rule whose head contains some instance_of(x,c2) and whose body contains instance_of(x,c1); or
- KB contains the subclass statement subclass_of(c1,c2).

<* is the transitive closure of <
The KB is acyclic if there is no class c that c <* c

Acyclicity conditions forces out sufficient properties

Similar in spirit to definatorial KBs
Summary

- Ability to represent hierarchically organized graphs is an important requirement from many applications
- We have a vocabulary to model such knowledge in ASPs
- It is theoretically (as well as empirically) an open question whether we could efficiently reason with this representation in a principled manner
Thank You!