Object Oriented Knowledge Bases in Logic Programming

Vinay K. Chaudhri
Stijn Heymans
Michael Weseel
Son Cao Tran

Acknowledgment

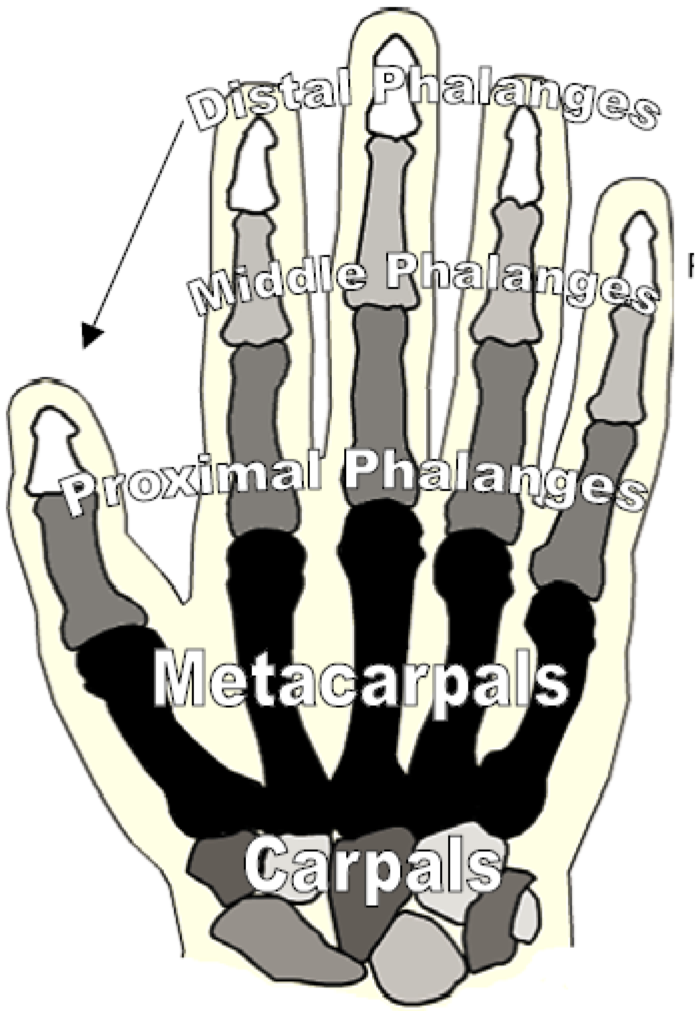
AURA

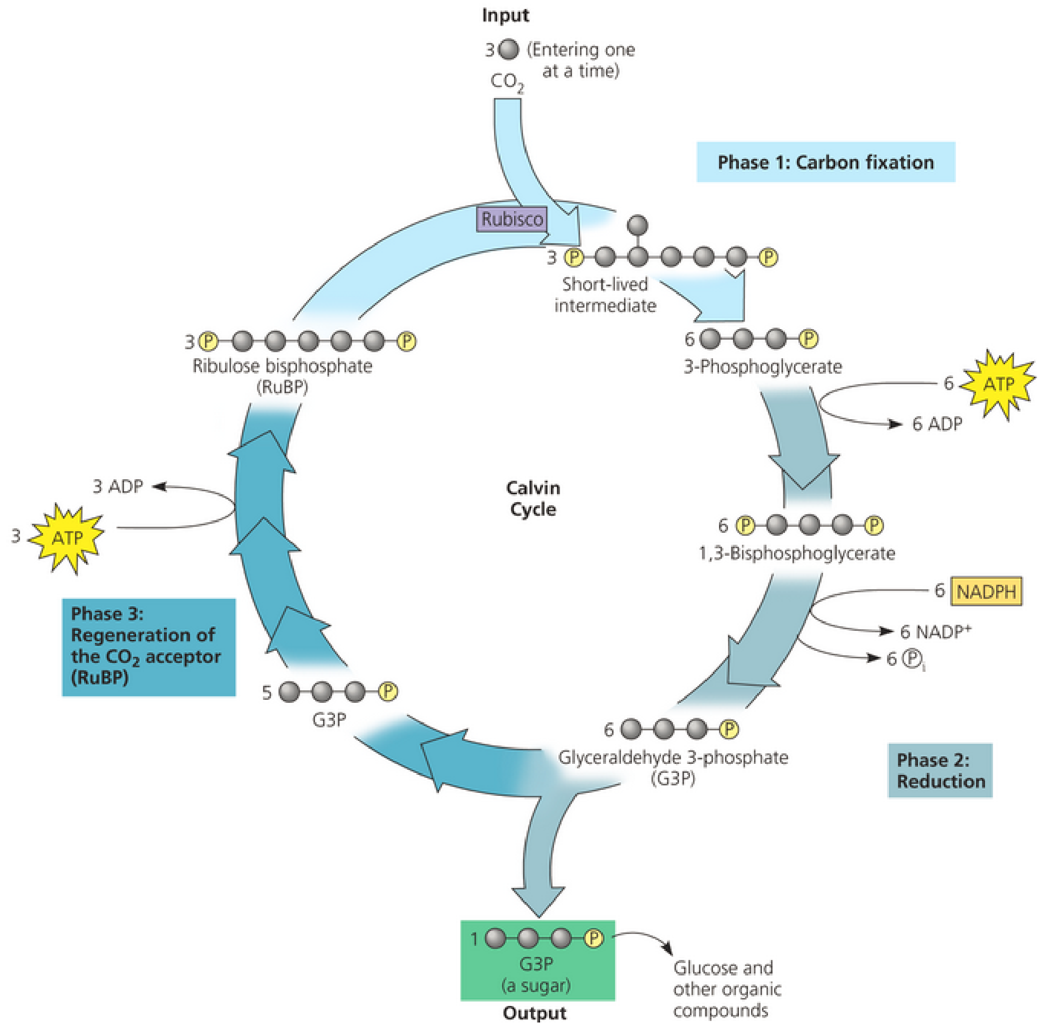
- This work is funded by Vulcan Inc. and SRI International
  - See http://www.vulcan.com and www.projecthalo.com
- AURA/Inquire Development Team
  - Eva Banik, Peter Clark, Roger Corman, Nikhil Dinesh, Debbie Frazier, Stijn Heymans, Sue Hinojoza, David Margolies, Adam Overholtzer, Aaron Spaulding, Ethan 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

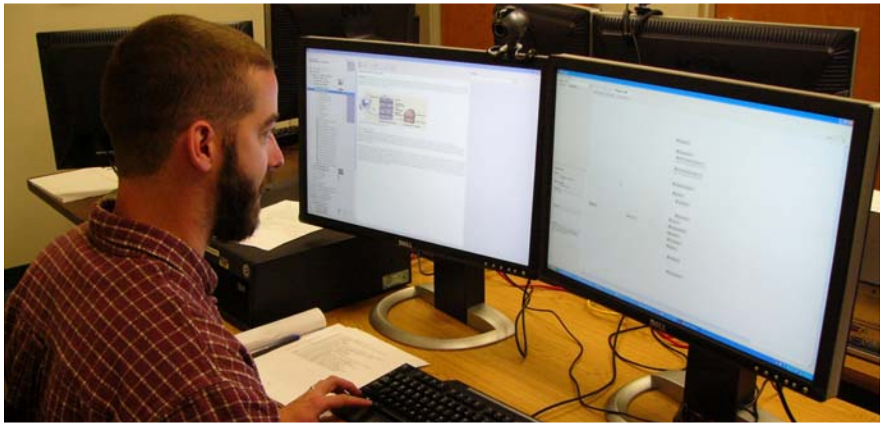
Process flow diagrams

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

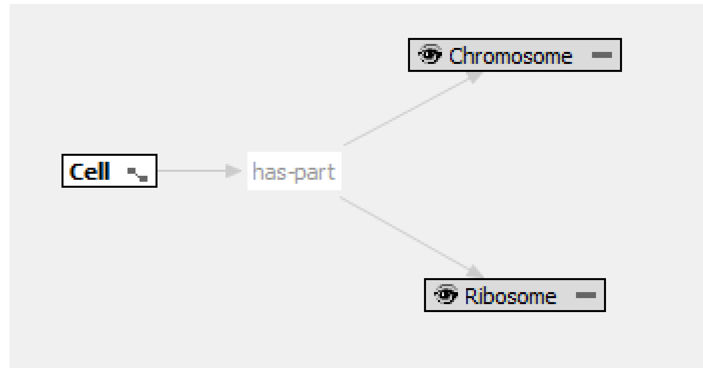
Need

- 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



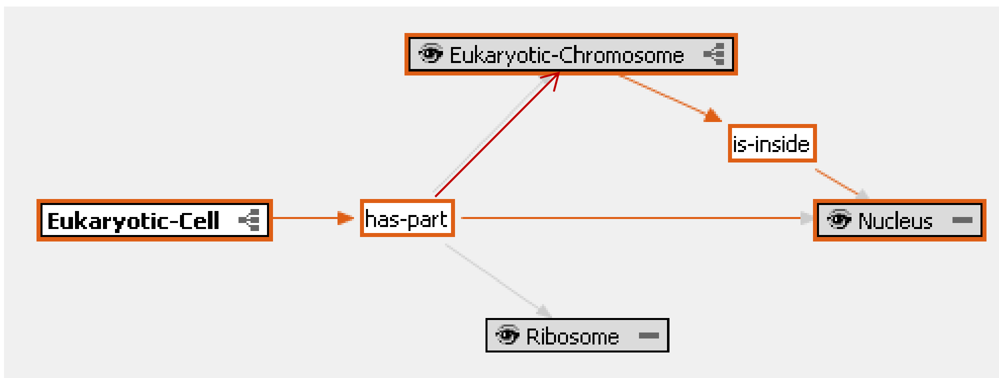
$$\begin{aligned}&\forall\, x : Cell(x) \\ &\qquad \rightarrow \exists\, y_1, y_2\!: LivingEntity(x) \wedge hasPart(x, y_1) \\ &\qquad \wedge\ hasPart(x, y_2)\ \wedge\ Ribosome(y_1) \\ &\qquad \wedge\ Chromosome(y_2)\end{aligned}$$

$$\begin{aligned}&\text{Cell} \sqsubseteq \text{Living-Entity} \sqcap \\ &\qquad (\exists \text{hasPart.Ribosome}) \sqcap \\ &\qquad (\exists \text{hasPart.Chromosome})\end{aligned}$$

KR in OOKB

- Unique Features in relation to DLs (and FDNC, Datalog±, ASP[fs])
  - Graph structured descriptions

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



$$\forall x : EukaryoticCell(x) \rightarrow \exists y_1, y_2, y_3 : Cell(x) \wedge hasPart(x, y_1) \wedge hasPart(x, y_2) \wedge hasPart(x, y_3) \wedge isInside(y_2, y_3) \wedge Ribosome(y_1) \wedge EukaryoticChromosome(y_2) \wedge Nucleus(y_3)$$

The knowledge shown in red is not expressible in known decidable description logics such as OWL 2
This can be captured in Rule Languages

KR in OOKB

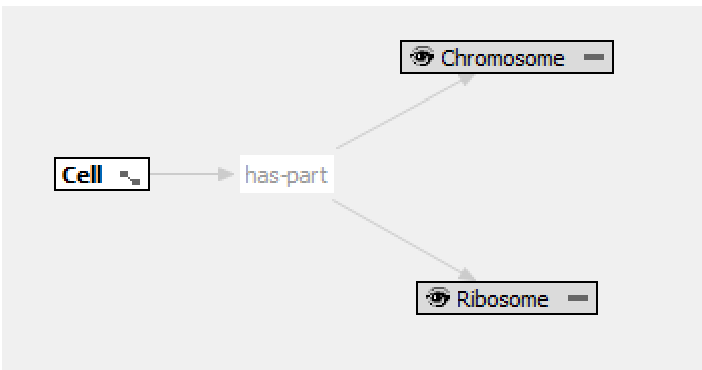
AURA

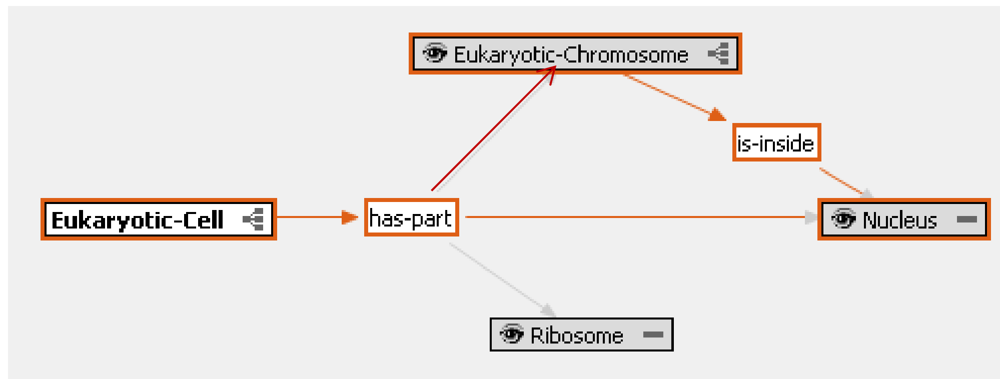
- Unique Features in relation to DLs
  - Inherit and Specialize

In the Eukaryotic Cell

Ribosome was inherited from Cell

Chromosome was inherited from Cell and specialized to Eukaryotic Chromosome





KR in OOKB

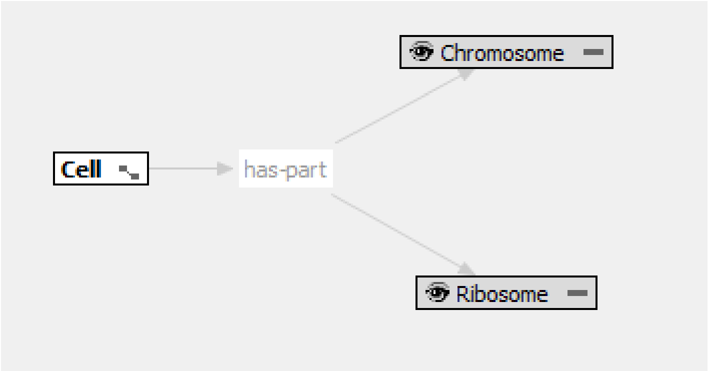
AURA

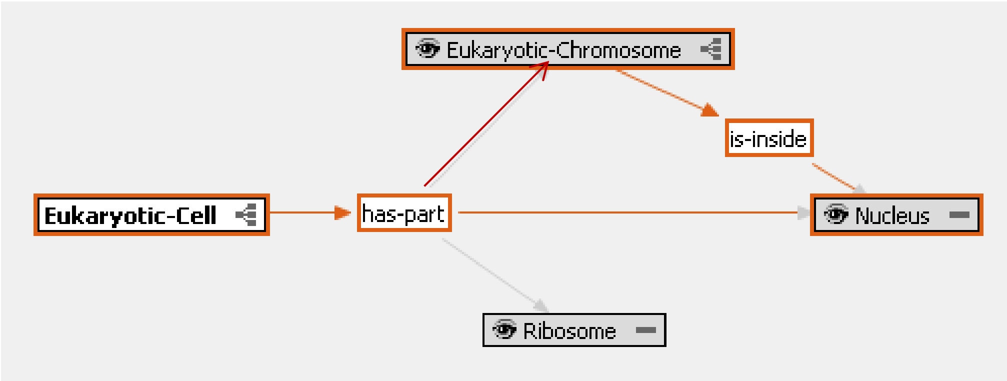
- Unique Features in relation to DLs
  - Inherit and Specialize

In the Eukaryotic Cell

Ribosome was inherited from Cell

Chromosome was inherited from Cell and specialized to Eukaryotic Chromosome





$$\forall x : Cell(x) \rightarrow hasPart(x, f_{Cell}\#1(x)) \wedge hasPart(x, f_{Cell}\#2(x)) \wedge Ribosome(f_{Cell}\#1(x)) \wedge Chromosome(f_{Cell}\#2(x))$$

$$\forall x : ECell(x) \rightarrow hasPart(x, f_{ECell}\#1(x)) \wedge hasPart(x, f_{ECell}\#2(x)) \; hasPart(x, f_{ECell}\#3(x)) \wedge isInside(f_{ECell}\#2(x), f_{ECell}\#3(x)) \wedge Ribosome(f_{ECell}\#1(x)) \wedge EukaryoticChromosome(f_{ECell}\#2(x)) \wedge Nucleus(f_{ECell}\#3(x))$$

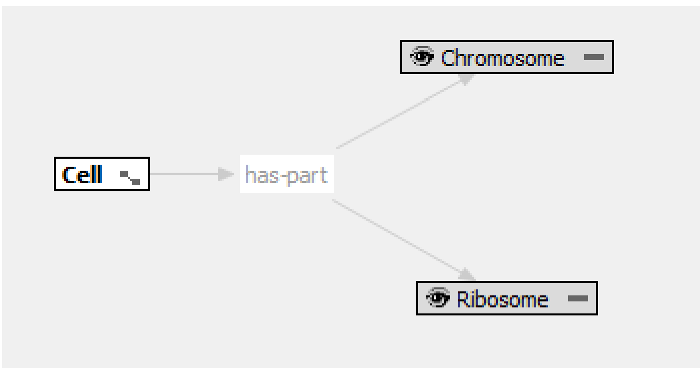
KR in OOKB

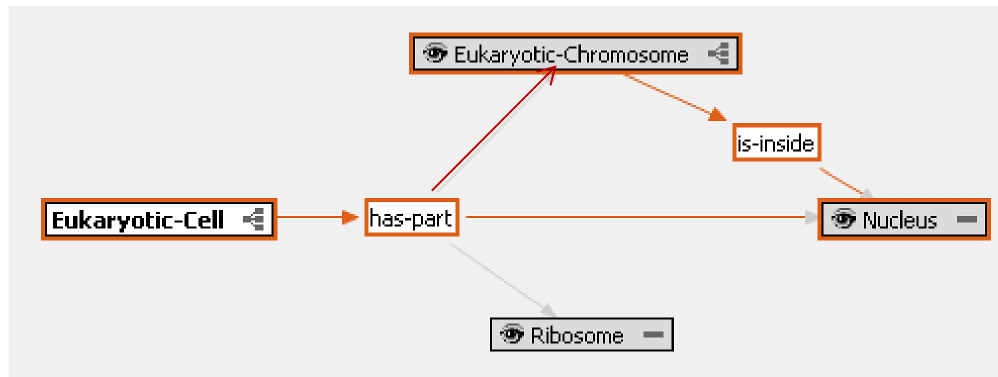
- Unique Features in relation to DLs
  - Inherit and Specialize

In the Eukaryotic Cell

Ribosome was inherited from Cell

Chromosome was inherited from Cell and specialized to Eukaryotic Chromosome





$$\forall\, x : Cell(x) \rightarrow$$
$$hasPart\big(x, f_{Cell}\#1(x)\big) \wedge hasPart\big(x, f_{Cell}\#2(x)\big)$$
$$\wedge\ Ribosome(f_{Cell}\#1(x))$$
$$\wedge\ Chromosome(f_{Cell}\#2(x))$$

$$\forall\, x : ECell(x) \rightarrow$$
$$hasPart\big(x, f_{ECell}\#1(x)\big) \wedge hasPart\big(x, f_{ECell}\#2(x)\big)$$
$$hasPart\big(x, f_{ECell}\#3(x)\big) \wedge isInside(f_{ECell}\#2(x), f_{ECell}\#3(x))$$
$$\wedge\ Ribosome(f_{ECell}\#1(x))$$
$$\wedge\ EukaryoticChromosome(f_{ECell}\#2(x))$$
$$\wedge\ Nucleus(f_{ECell}\#3(x))$$

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

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

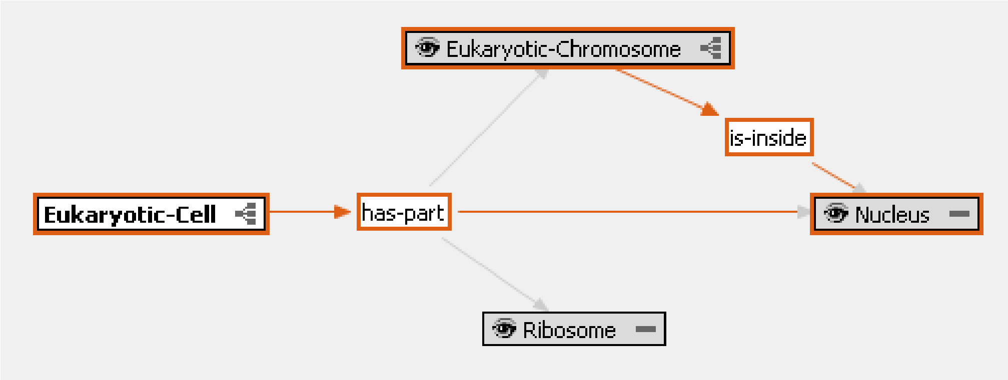
- Computational properties
  - Reasoning with OOKB in general, un-decidable
  - There are, however, some decidable fragments that introduce guardedness and acyclic structure in the KB
- To consider the computational properties in detail, let us specify the full language and consider its sub-languages.

Sub-languages of OOKB

AURA

Axiom type	Referent	Axiom Specification
Taxonomic	$\mathcal{T}$	class(c), individual(o), subclass_of(c.c'), instance_of (i,c)
Relation Hierarchy	$\mathcal{H}$	subrelation_of(s_1,s_2)
Inverse relations	$\mathcal{I}$	inverse(s_1,s_2)
Existential rules	$\mathcal{E}$	See next slide
Sufficient conditions	$\mathcal{P}$	See next slide
Qualified number constraints	$\mathcal{Q}$	constraint(<u>*exac*</u>t,f(X),s,n) ← instance_of(X,c)
Domain/range	$\mathcal{C}$	domain(s,r), range(s,d)
Disjointness	$\mathcal{J}$	disjoint(c,c')
Relation composition	$(\circ)$	compose(s_1,s_2,s_3)
Equality/Inequality		eq(t1,t2), neq(t1,t2)

Graph-structured existential rules



$$
\begin{aligned}
&\forall\, x : ECell(x) \rightarrow \\
&\quad hasPart\big(x, f_{ECell}\#1(x)\big) \wedge hasPart\big(x, f_{ECell}\#2(x)\big) \\
&\quad hasPart\big(x, f_{ECell}\#3(x)\big) \wedge isInside(f_{ECell}\#2(x), f_{ECell}\#3(x)) \\
&\quad \wedge Ribosome(f_{ECell}\#1(x)) \\
&\quad \wedge EukaryoticChromosome(f_{ECell}\#2(x)) \\
&\quad \wedge Nucleus(f_{ECell}\#3(x))
\end{aligned}
$$

$$2\big\{value\big(hasPart, x, f_{ECell}\#1(x)\big), instance\_of\,(f_{ECell}\#1(x), Ribosome)\big\}2 \leftarrow ECell(x)$$

$$2\big\{value\big(hasPart, x, f_{ECell}\#2(x)\big), instance\_of\,(f_{ECell}\#2(x), EukaryoticChromosome)\big\}2 \leftarrow ECell(x)$$

$$3\big\{value\big(hasPart, f_{ECell}\#2(x), f_{ECell}\#3(x)\big), value\big(hasPart, x, f_{ECell}\#3(x)\big), instance\_of\,(f_{ECell}\#3(x), Nucleus)\big\}3 \leftarrow ECell(x)$$

Sufficient Conditions

AURA

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

$$instance\_of(X, C) \leftarrow Body(\widehat{X})$$

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

$$\begin{aligned} instance\_of(X, ECell) \leftarrow \\ instance\_of(x, Nuclues), \\ instance\_of(y, Cell) \\ value(has\_part, y, x) \end{aligned}$$

Sub-languages of OOKB

AURA

Axiom type	Referent	Axiom Specification
Taxonomic	T	class(c), individual(o), subclass_of(c.c'), instance_of (i,c)
Relation Hierarchy	H	subrelation_of(s_1,s_2)
Inverse relations	I	inverse(s_1,s_2)
Existential rules	E	Already explained
Sufficient conditions	P	Already explained
Qualified number constraints	Q	constraint(*exac*t,f(X),s,n) ← instance_of(X,c)
Domain/range	C	domain(s,r), range(s,d)
Disjointness	J	disjoint(c,c')
Relation composition	(o)	compose(s_1,s_2,s_3)
Equality/Inequality		eq(t1,t2), neq(t1,t2)

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

Reasoning problems

AURA

- ***Consistency (C_1)***: Whether the KB has an answer set
- ***Concept Satisfiability (C_2)*** : given an instance I and a class C, determine if $KB \cup instance\_of(I, C)$ has an answer set
- ***Entailment (E):*** Given a ground atom *a*, determine *if KB* |= *a*
- ***Question answering (QA)***: Determine if $KB \models q(\delta)$ where q is a query atom, $\{X_1, \ldots, X_n\}$ are distinct variables occurring in it, and $\delta = \{X_1/a_1, \ldots, X_n/a_n\}$ 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 $\sum_1^1 complete$ 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

AURA

- 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!