Logic for CS

黃瀚萱
Department of Computer Science
National Chengchi University
2020 Spring

Schedule, Part I

Date	Topic
3/6	Introduction to this course
3/13	Thinking as computation
3/20	Propositional Logic
3/27	Logic Inference
4/3	Off
4/10	First Order Logic
4/17	Interpretation of FOL
4/24	Inference in FOL (Online)

Schedule, Part II

Date	Topic
5/1	**Prolog Basics & KR (Online)**
5/8	**Midterm Exam**
5/15	**上課**
5/22	Logic Programming
5/29	Logic Programming
6/5	Applications of logic in computation
6/12	Final Project Presentation
6/19	Term Exam

Logic Programming Basics

Logic Programming

- Logic programming is a technology that comes fairly close to embodying the declarative ideal.
  - Systems are constructed by expressing knowledge in a formal language and the problem should be solved by running inference processes.
- Algorithm = Logic + Control

Procedural Programming Language

```
def factorial(n):
    r = 1
    for i in range(1, n+1):
        r = r * i
    return r
```

Declarative Programming Language

```
mother_child(trude, sally).

father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).

sibling(X, Y)      :- parent_child(Z, X), parent_child(Z, Y).

parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).


?- sibling(sally, erica).
Yes
```

Declarative Programming Language

Mercury
Haskell

Prolog

SQL

Procedural (Imperative) Programming Language

C
Pascal

C++
JAVA

Python
Ruby
PHP

Why Declarative?

- AI researchers attempt to distinguish between *declarative* and *imperative* knowledge.
- Declarative knowledge
  - Explicitly encoded in the machine
- Imperative (procedure) knowledge
  - manifested in programs in the machine

Why Declarative?

- When knowledge is represented as declarative sentences, the sentences are manipulated by reasoning process when the machine attempts to use the knowledge.

- The component that decides how to use declarative knowledge is separated from the knowledge itself.

- Algorithm = Logic + Control

Prolog

- The most widely used logic programming language.
  - Many expert systems have been developed in Prolog
- Other languages can also be used for logic programming
  - Scheme
  - Lisp
  - ACL2

Syntax

- A Prolog program are sets of **definite clauses** written in a notation somewhat different from standard FOL.
- Uppercase letters: variables
- Lowercase letters: constants
- Commas separate conjuncts in a clause
- The clause is written in the backward direction

Example

- $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$
- criminal(X) :- american(X), weapon(Y), sells(X, Y, Z), hostile(Z).

Example

```
criminal(X) :- american(X), weapon(Y), sells(X, Y, Z), hostile(Z).

sells(West, X, Nono) :- missle(X), owns(Nono, X).

weapon(X) :- missle(X).

hostile(X) :- enemy(X, America).

owns(Nono, M1).

missle(M1).

american(West).

enemy(Nono, America).


query(criminal(West)).
```

Execution of Prolog Programs

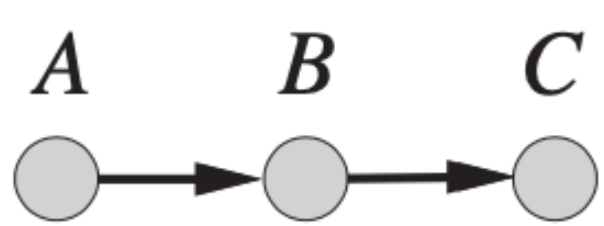
- The execution of Prolog programs is done via DFS backward chaining, where clauses are tried in the order in which they are written in the knowledge base.
- Some aspects of Prolog do not meet the standard logical inference.
  - Only Datalog (database semantics but not standard FL)
  - No occur check; some unsound inferences can be made.
  - Depth-first backward-chaining algorithm is incomplete when given the wrong axioms.

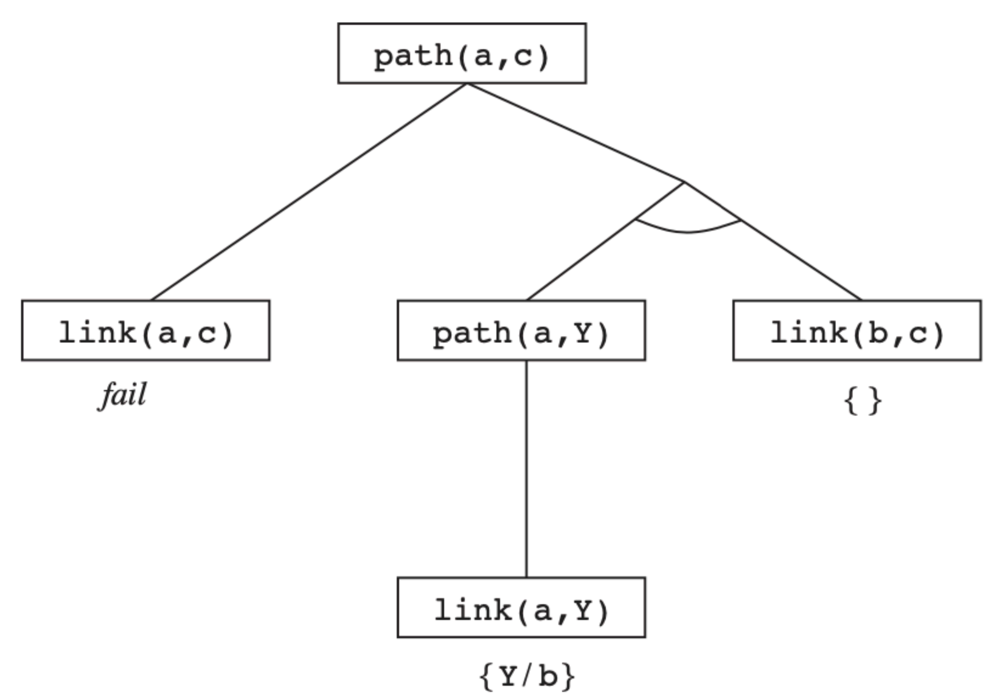
Redundant Inference

- The mismatch between depth-first search and search trees that include repeated states and infinite paths.

  - Path(x, z) :- Link(x, z).

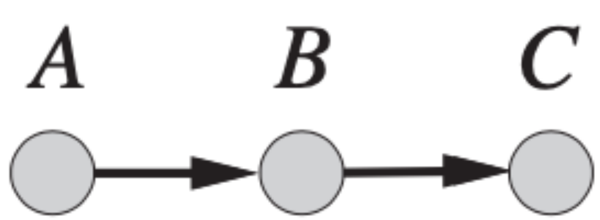
  - Path(x, z) :- Path(x, y) $\wedge$ Link(y, z).

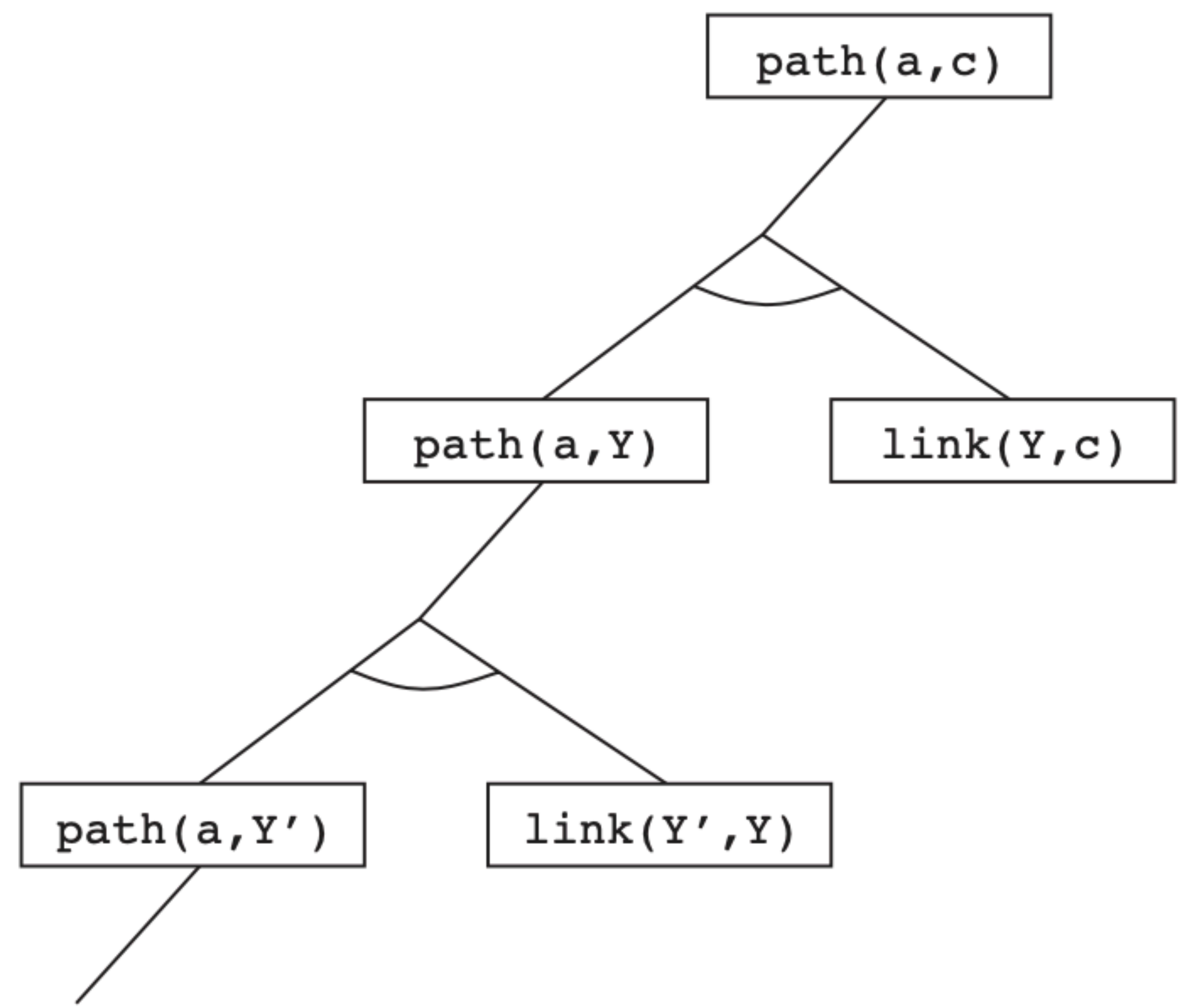




Redundant Inference

- With a different order, the Prolog program follows the infinite paths.
  - Path(x, z) :- Path(x, y) ∧ Link(y, z).
  - Path(x, z) :- Link(x, z).





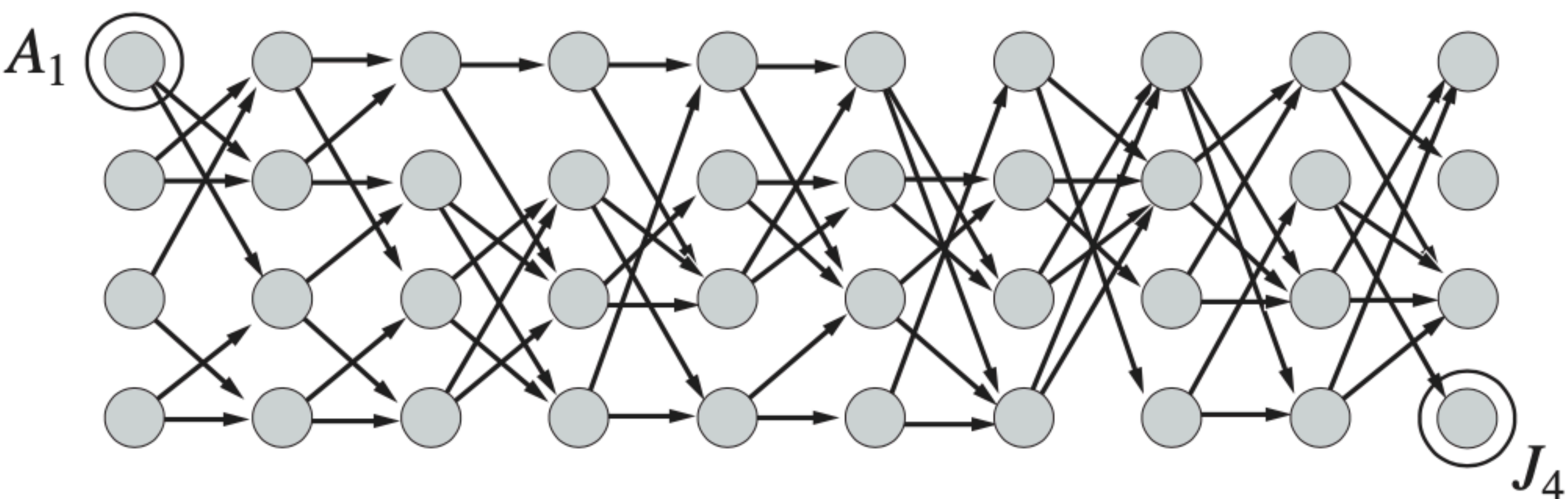
Prolog is Incomplete

- Prolog is incomplete as a theorem prover for definite clauses--even for Datalog programs.
- For some knowledge bases, it fails to prove sentences that are entailed.
- Forward chaining does not suffer from this problem because it performs BFS search.

Inefficiency of DFS

- DFS is slow to find the solution because repeated states will be explored multiple times.

- Memoization (dynamic programming) can be used for speed up.



Database Semantics of Prolog

- Prolog employs the database semantics, rather than first-order semantics. (Datalog)
- Unique names assumption
  - Every Prolog constant and every ground term refers to a **distinct** object.
- Closed world assumption
  - Only sentences that are true those that are entailed by KB.
  - No way to assert a sentence is false!
- Less general and expressive than FOL, but more efficient and more concise.

Unique Names Assumption

- CS and EE are different.
- 101, 102, and 106 are all different.
- There are four distinct courses.

```
Course(CS, 101).
Course(CS, 102).
Course(CS, 106).
Course(EE, 101).
```

Closed World Assumption

- There are no other courses.
- So there are exactly four courses.

```
Course(CS, 101).
Course(CS, 102).
Course(CS, 106).
Course(EE, 101).
```

Datalog vs FOL

- There are no other courses.
- So there are exactly four courses.

```
Course(CS, 101).
Course(CS, 102).
Course(CS, 106).
Course(EE, 101).
```

At most four courses

$\text{Course}(d, n) \leftrightarrow (d=CS \wedge n=101) \vee (d=CS \wedge n=102) \vee (d=CS \wedge n=106) \vee (d=EE \wedge n=101)$

$x = y \leftrightarrow (x = CS \wedge y = CS) \vee (x = EE \wedge y = EE) \vee (x = 101 \wedge y = 101) \vee (x = 102 \wedge y = 102) \vee (x = 106 \wedge y = 106)$

At least four courses

Install problog

- pip3 install problog

https://dtai.cs.kuleuven.be/problog/index.html

Your First Prolog Program

- Edit a file such as test.pl

```
die(x) :- person(x).
person(John).

query(die(John)).
```

Run Your First Prolog Program

- problog test.pl

```
die(x): 1
```

Knowledge Representation in FOL

Knowledge Representation

- What content to put into a knowledge base
- How to represent facts about the world
- Representation is still a very hot research topic in the AI area.

Call for Papers of AAAI 2020

These include (but are not limited to) traditional topics such as search, planning, **knowledge representation**, reasoning, natural language processing, robotics and perception, multiagent systems, statistical learning, and deep learning.

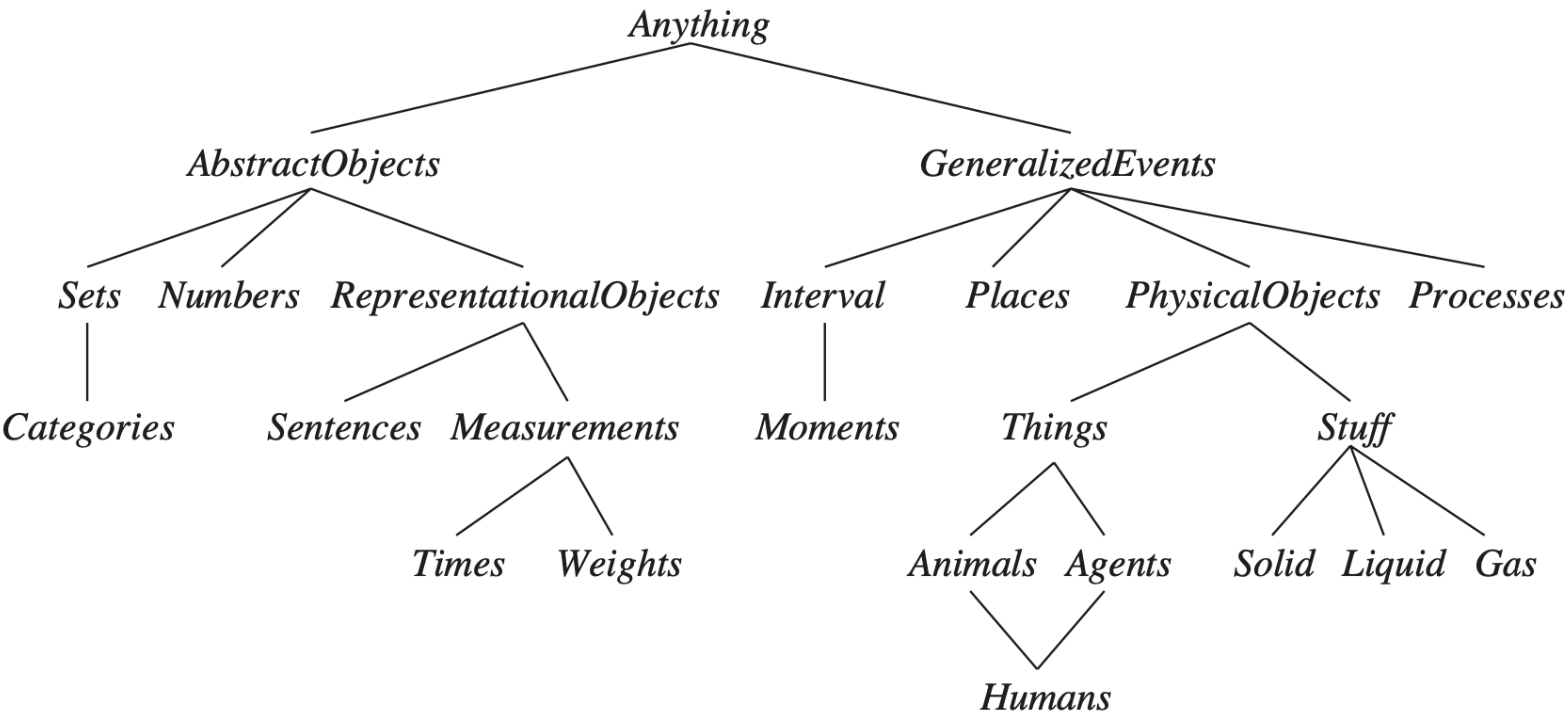
General Concepts of Knowledge Representation

- Complex domains such as clinical support systems or auto driving require more general and flexible representations.
- General concepts
  - Actions
  - Time
  - Physical objects
  - Beliefs
- Ontological engineering

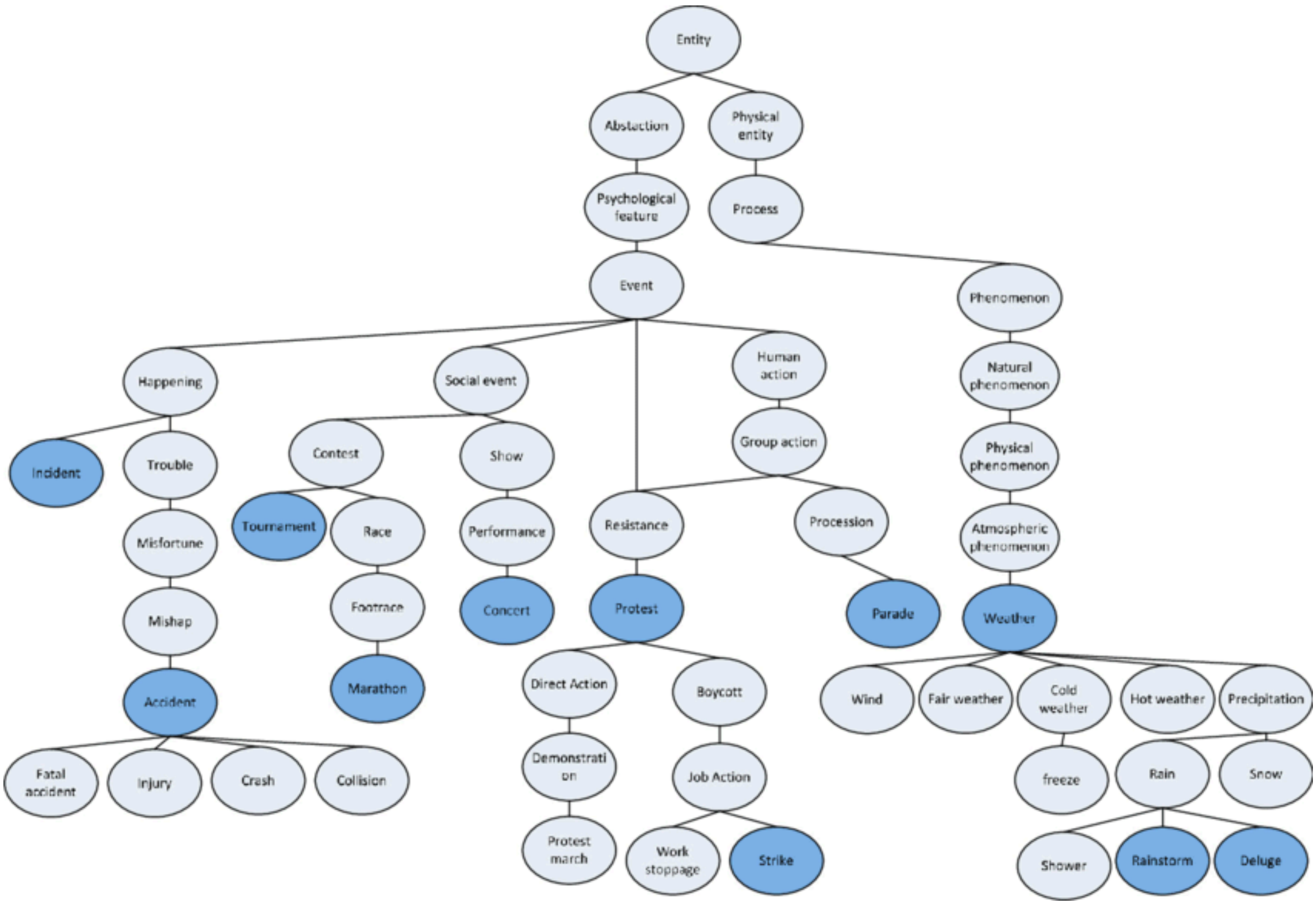
Upper Ontology

- To represent everything in the world is impractical
  - Some attempts have been made to do that
- Ontology of words
  - WordNet
  - EHowNet
- Upper ontology: general framework of concepts

Upper Ontology of the World



WordNet: Ontology of Words



Difficulty of Using FOL in KR

- A rule may have exceptions
  - Tomatoes are red
    - Some tomatoes are green, yellow, or orange
- A rule may hold only to a degree
- The ability to handle exceptions and uncertainty

General-purpose Ontology

- Do all these ontologies converge on a general purpose ontology?
  - Maybe
- A general-purpose ontology should be applicable in more or less any special-purpose domain.
  - No representation issue can be brushed under the carpet.
- In a specific domain, different areas of knowledge must be unified.
  - A auto driving system should deal with the traffic and the weather knowledge at the same time.

General-purpose Ontology in Real AI Applications

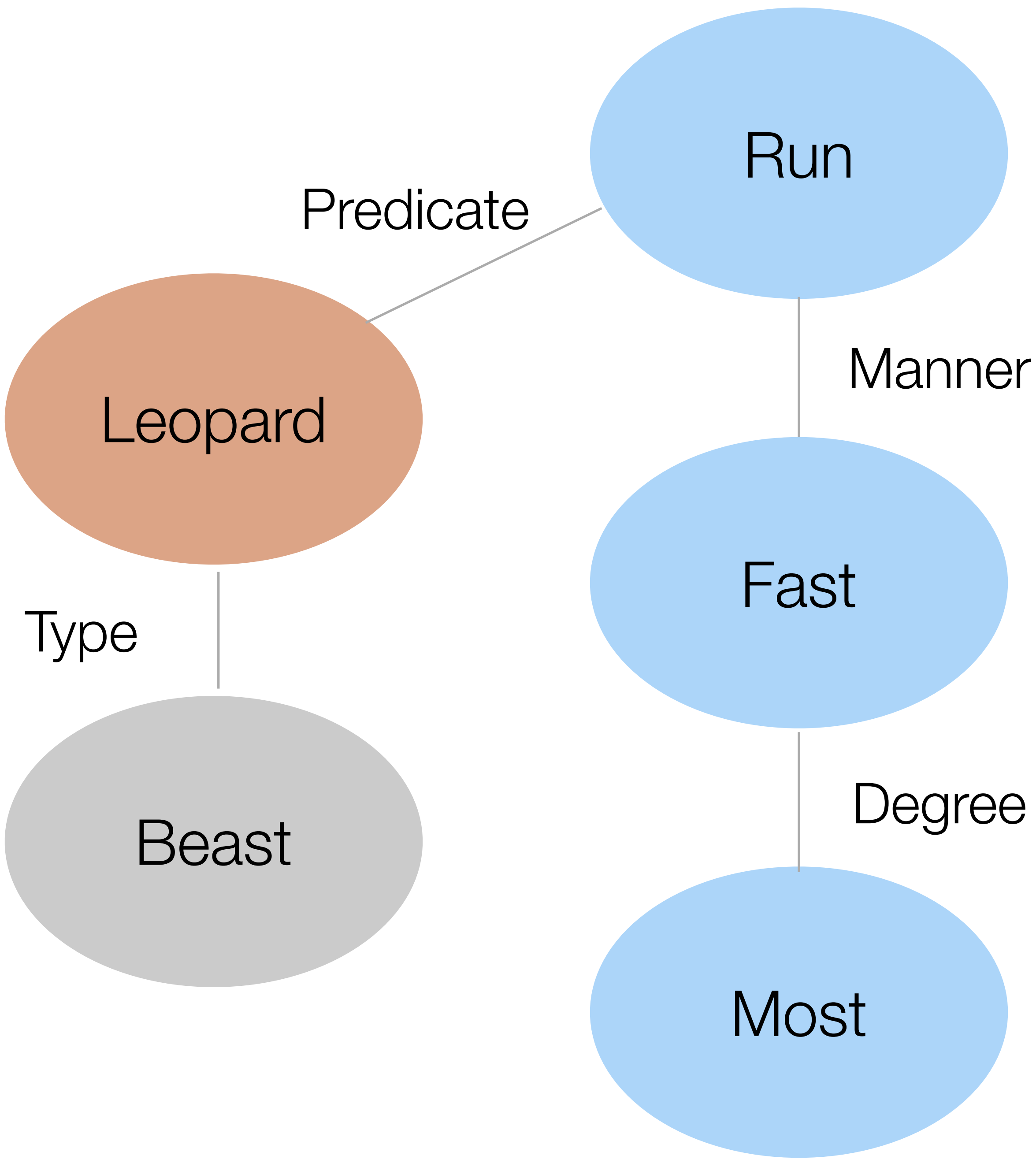
- Nearly none of the top AI applications make use of a shared ontology.
  - They all use special-purpose knowledge engineering.
  - Question answering, auto-driving, debating, etc.
- Every ontology is a social agreement among people with some common motive in sharing.

WordNet vs EHowNet

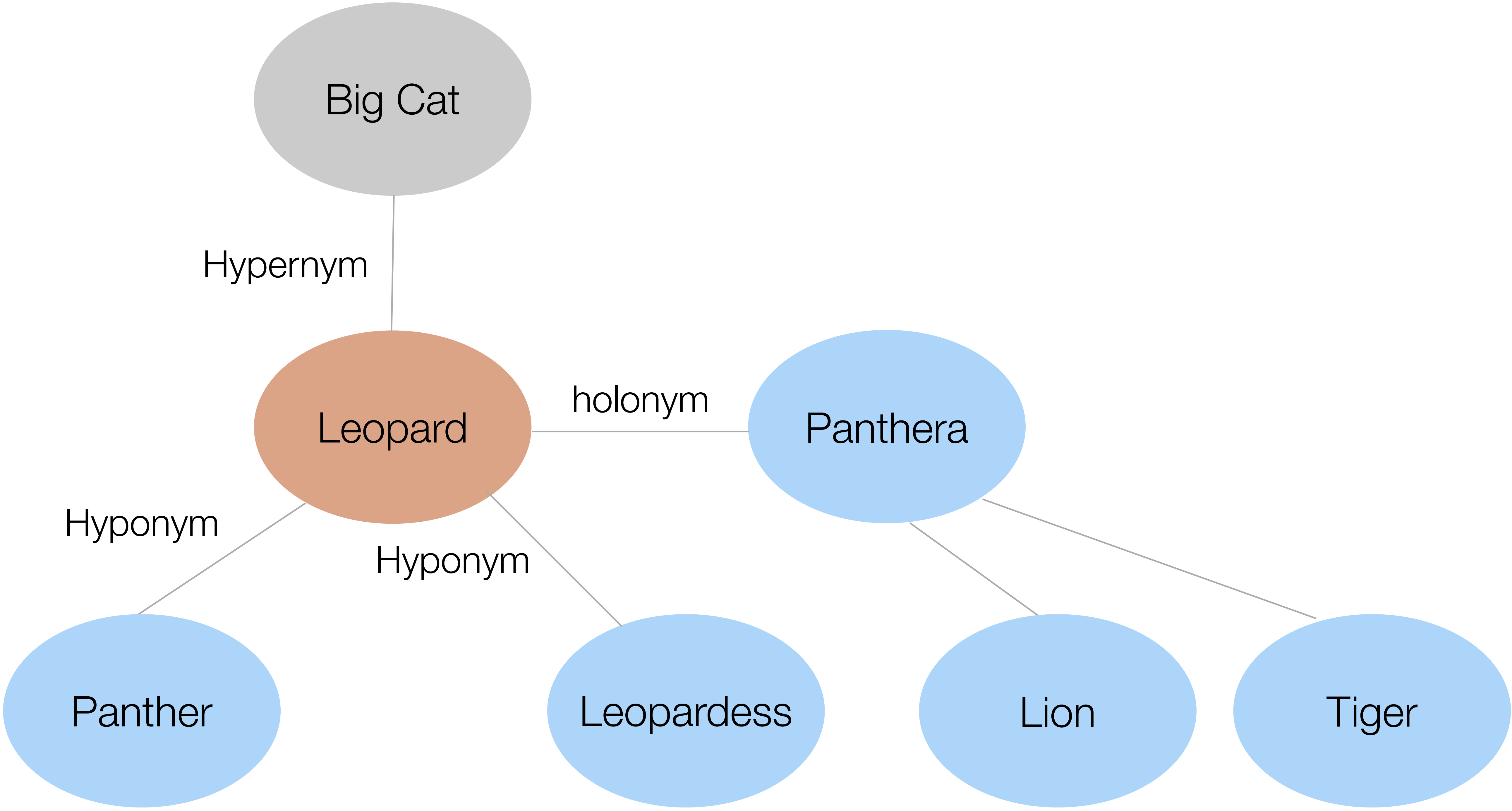
- EHowNet
  - Tiger: {beast|走獸:qualification={fierce|暴}}
  - Lion: {beast|走獸:qualification={HighRank|高等:degree={most|最}}}
  - Leopard: {beast|走獸:predication={run|跑:manner={fast|快:degree={most|最}},agent={~}}}
- WordNet

- S: (n) **leopard**, Panthera pardus (large feline of African and Asian forests usually having a tawny coat with black spots)
  - *direct hyponym* / *full hyponym*
    - S: (n) leopardess (female leopard)
    - S: (n) panther (a leopard in the black color phase)
  - *member holonym*
    - S: (n) Panthera, genus Panthera (lions; leopards; snow leopards; jaguars; tigers; cheetahs; saber-toothed tigers)
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) big cat, cat (any of several large cats typically able to roar and living in the wild)

EHowNet



WordNet


Big Cat
Hypernym
Leopard
holonym
Panthera
Hyponym
Hyponym
Panther
Leopardess
Lion
Tiger

Creation of Ontologies

- By a team of trained ontologist and logicians
  - Creating the architecture and writing the axioms.
- By importing categories, attributes, and values from an existing database.
  - DBpedia was built by importing structured facts from Wikipedia.
- By parsing text documents and extracting information from them.
  - TextRunner
- By enticing unskilled amateur to compose commonsense knowledge.
  - OpenMind

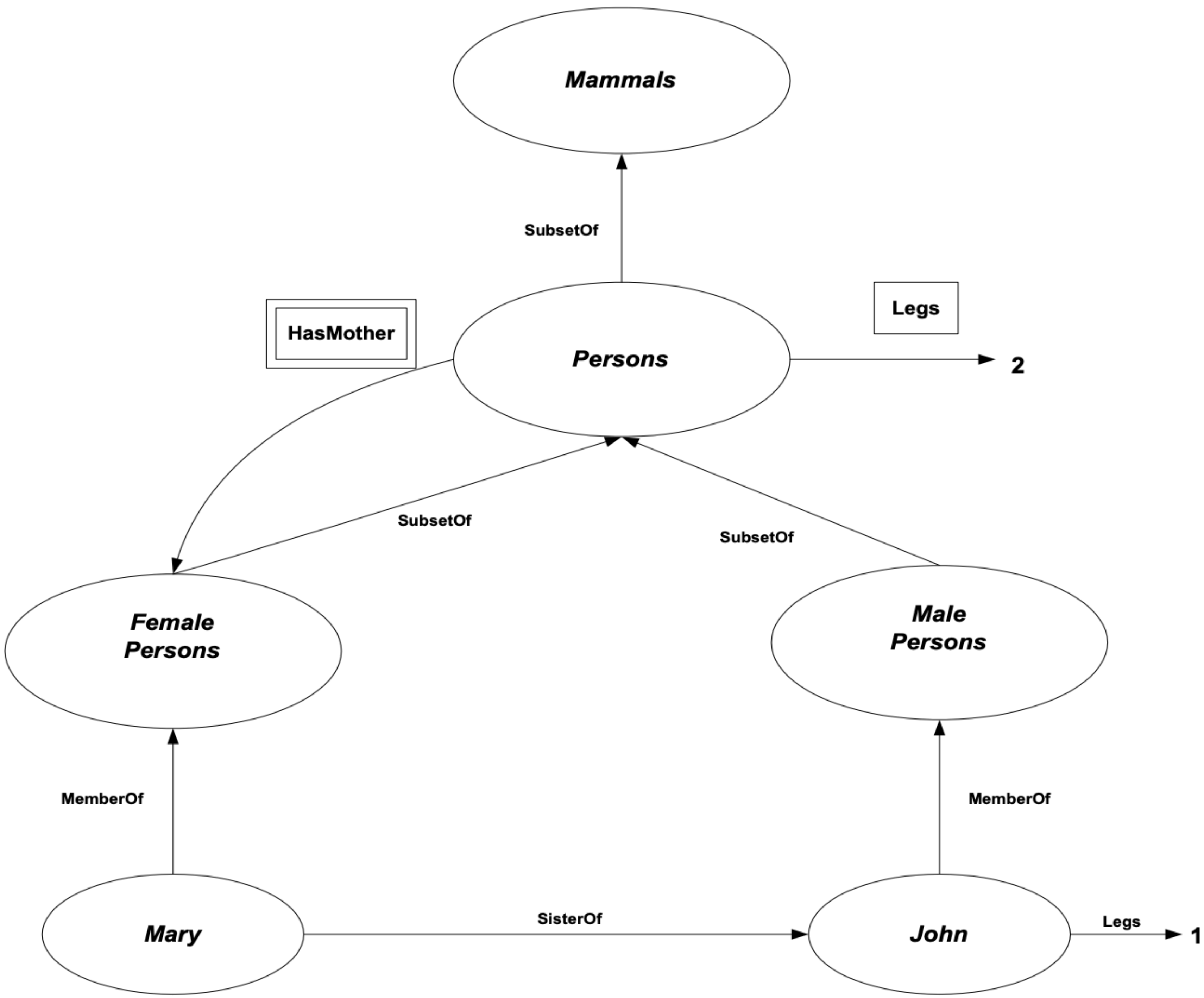
Category and Objects

- The first step to create a ontology is to organize objects into categories.
    - Much reasoning takes place at the level of categories.
    - A shopper would normally have to goal of buying a wireless mouse, rather than a particular mouse.
- Categories also serve to make predictions about objects.
    - Fruits are edible.
    - Watermelon is a kind of fruits.
    - Watermelon is edible.

Categories in FOL

- In FOL, two ways to represent categories.
  - Predicates
    - Person(x), Monkey(y)
  - Reification of objects with membership predicates
    - *MemberOf*(x, Persons)
    - *SubsetOf*(Persons, Mammals)

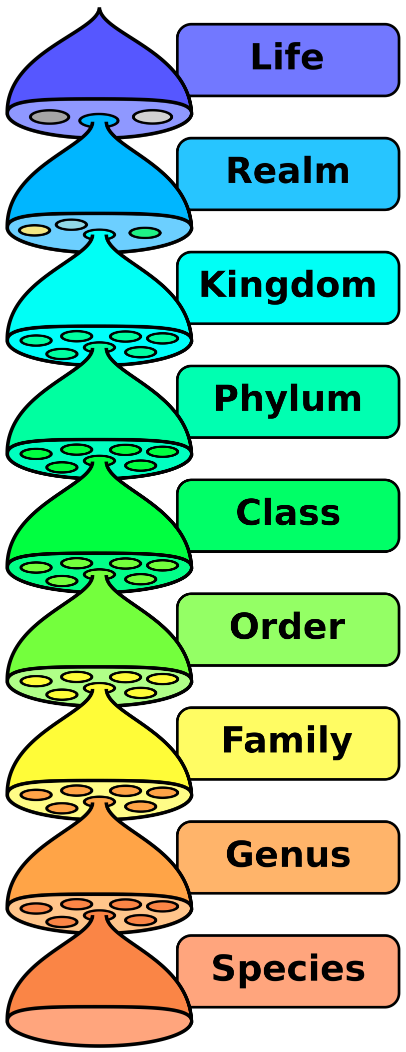
Categories in FOL



http://www.intelligence.tuc.gr/~agents/previous/AI_grad_2003/kr-fol-revisited.pdf

Knowledge Representation of Categories

- Categories serve to organize and simplify the knowledge base through **inheritance**.
- The concept of category is an important abstraction in knowledge representation and reasoning.
- Taxonomies is an organization of categories.
    - Biology
    - Library science
    - E-commerce



Examples

- An object as a member of a category
  - MemberOf(John, People)
- A category is a subclass of another class
  - SubsetOf(People, Animals)
- A property of a category
  - $\forall x \, (\mathit{MemberOf}(x, \text{Tiger}) \rightarrow \text{Run}(x))$

Example

- Members of a category can be assigned with more properties.
  - $\forall x\ (Yellow(x) \land MemberOf(x, Panthera) \land \neg Spotty(x) \land \neg Striped(x) \land Roar(x) \rightarrow MemberOf(x, Lion))$
  - *MemberOf*(Panthera, BigCat)
    - Panthera is a category and is a member of BigCat, so BigCat must also be a category.




Disjoint (無交集)

- To state relations between categories that are not subclasses of each other.
  - Males and Females are subclasses of Animals
  - A male cannot be a female
- **Disjoint** categories have no members in common.
  - However, we still do not know that an animal that is not a male must be a female.

Exhaustive Decomposition (窮盡分解)

- If we say that males and females constitute an **exhaustive decomposition** of the animals, then we can know that if an animal is not a male then it must be a female.

- A disjoint exhaustive decomposition is known as a partition

  - Disjoint({Animals, Vegetables})

  - ExhaustiveDecomposition({Americans, Canadians, Mexicans}, NorthAmericans) **(Not a partition!)**

  - Partition({Males, Females}, Animals)

Categories in FOL

- $Disjoint(s) \leftrightarrow (\forall x,y\ MemberOf(x, s) \wedge MemberOf(y, s) \wedge (\neg x=y) \rightarrow Intersection(x, y) = \{\})$
- $ExhaustiveDecomposition(s, x) \leftrightarrow (\forall i\ MemberOf(i, x) \leftrightarrow \exists y\ MemberOf(y, s) \wedge MemberOf(i, y))$
- $Partition(s, x) \leftrightarrow Disjoint(s) \wedge ExhaustiveDecomposition(s, x)$

Defined by Providing Conditions

- Categories can also be defined by providing necessary and sufficient conditions for membership.
- Member(x, Bachelors) $\leftrightarrow$ $\neg$Married(x) $\wedge$ MemberOf(x, Adults) $\wedge$ MemberOf(x, Males)

Physical Composition

- The idea that one object can be part of another.
  - Taipei is part of Taiwan
- PartOf relation
  - PartOf(Taipei, Taiwan)
  - PartOf(Taiwan, EasternAsia)
  - PartOf(Japan, EasternAsia)
  - PartOf(EasternAsia, Asia)

Properties of the PartOf Relation

- Transitive (遞移律)
  - $PartOf(x, y) \land PartOf(y, z) \rightarrow PartOf(x, z)$
- Reflexive (反身性)
  - $PartOf(x, x)$
- $PartOf(Taipei, Taiwan) \land PartOf(Taiwan, EasternAsia) \land PartOf(EasternAsia, Asia) \rightarrow PartOf(Taipei, Asia)$

Composite Object

- Composite objects are often characterized by structural relations among parts.

- $Biped(x) \rightarrow \exists l, r, b\ Leg(l) \wedge Leg(r) \wedge Body(b) \wedge$
  $PartOf(l, x) \wedge PartOf(r, x) \wedge PartOf(b, x) \wedge$
  $Attached(l, b) \wedge Attached(r, b) \wedge$
  $\neg l{=}r \wedge (\forall y\ Leg(y) \wedge$
  $PartOf(y, x) \rightarrow (y{=}l \vee y{=}r)\ )$

Bunch

- The oranges in this bag weigh $100 NTD.
  - The set of oranges does not have a weight because a set is an abstract concept.
- BunchOf({x}) = x
  - BunchOf(Oranges) is the composite object consisting of all oranges.
- Each element of s is part of BunchOf(s):
  - $\forall x\ MemberOf(x, s) \rightarrow PartOf(x, BunchOf(s))$

Limitation of FOL

- FOL is very general and based on very primitive concepts.
  - Constants, variables, function symbols, predicates, and quantifiers
  - FOL is general and flexible
- FOL provides no explicit help for higher-level abstraction.

Weakness of FOL

- FOL does not allow
  - Non-monotonicity (非單調性)
  - Uncertainty
  - Belief revision