Knowledge Representation in Protégé –OWL

Please install from CDs or USB pens provided:

- Protégé 3 Beta – complete installation
- Racer – plus a shortcut to start it easily
- GraphViz – please install in default location
- Example ontologies
- Optional: Long version of Pizza tutorial
  “Pizza finder” application

1

Ontology Design Patterns and Problems: Practical Ontology Engineering using Protege-OWL

Alan Rector[1], Natasha Noy[2], Holger Knublauch[2],
Guus Schreiber,[3] Mark Musen[2]

[1]University of Manchester
[2]Stanford University
[3] Free University of Amsterdam

rector@cs.man.ac.uk
{noy, holger}@smi.stanford.edu
schreiber@cs.vu.nl
musen@smi.stanford.edu

2

Program

I Ontologies and “Best Practice”
II Creating an ontology – useful patterns
III Hands on examples
IV Patterns: n-ary relations
V Patterns: classes as values
VI Patterns: part-whole relations
VII Summary

3

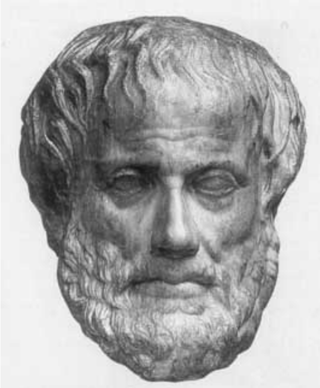
Part I: Ontologies & “Best Practice”

- What are Ontologies & a review of History
- Semantic Web
- OWL
- “Best Practice”
  - Semantic Web Best Practice & Deployment Working Group (SWBP)

4

What Is An Ontology?

- Ontology (Socrates & Aristotle 400-360 BC)
- The study of being
- Word borrowed by computing for the explicit description of the conceptualisation of a domain:
  - concepts
  - properties and attributes of concepts
  - constraints on properties and attributes
  - Individuals (often, but not always)
- An ontology defines
  - a common vocabulary
  - a shared understanding



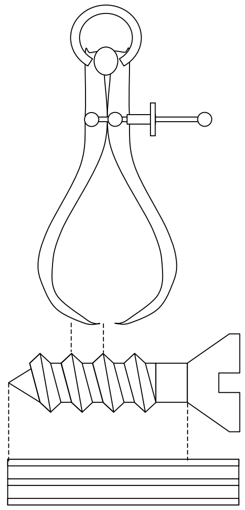
5

Why Develop an Ontology?

- To share common understanding of the structure of descriptive information
  - among people
  - among software agents
  - between people and software
- To enable reuse of domain knowledge
  - to avoid "re-inventing the wheel"
  - to introduce standards to allow interoperability

6

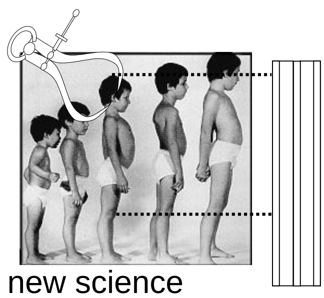
Measure the world…*quantitative models* *(not ontologies)*



- Quantitative
  - Numerical data:
    - 2mm, 2.4V, between 4 and 5 feet
  - Unambiguous tokens
  - Main problem is accuracy at initial capture
  - Numerical analysis (e.g. statistics) well understood
- Examples:
  - How big is this breast lump?
  - What is the average age of patients with cancer ?
  - How much time elapsed between original referral and first appointment at the hospital ?

7

describe the our understanding of the world - *ontologies*



- Qualitative
  - Descriptive data
    - Cold, colder, blueish, not pink, drunk
  - Ambiguous tokens
    - What's wrong with being drunk ?
      - Ask a glass of water.
  - Accuracy poorly defined
  - Automated analysis or aggregation is a new science
- Examples
  - Which animals are dangerous ?
  - What is their coat like?
  - What do animals eat ?

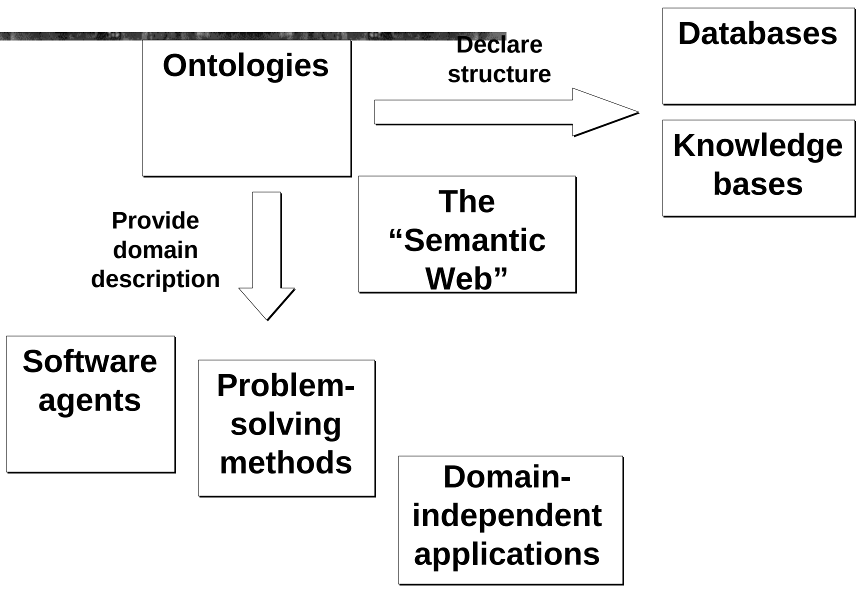
8

More Reasons

- To make domain assumptions explicit
  - easier to change domain assumptions (consider a genetics knowledge base)
  - easier to understand and update legacy data
- To separate domain knowledge from the operational knowledge
  - re-use domain and operational knowledge separately (e.g., configuration based on constraints)
- To manage the combinatorial explosion

9

An Ontology should be just the Beginning



10

Outline

- What are Ontologies
- Semantic Web
- OWL
- Best Practice

11

The semantic web

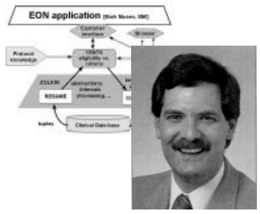
- Tim Berners-Lee’s dream of a computable meaningful web
  - Now critical to Web Services and Grid computing
- Metadata with everything
  - Machine understandable!
    - Ontologies are one of the keys

12

Understanding rather than text matching

- Google image results for
  - Charlie Safran

  - Mark Musen



  - Alan Rector

13

Ontology Examples

- Taxonomies on the Web
  - Yahoo! categories
- Catalogs for on-line shopping
  - Amazon.com product catalog
- Dublin Core and other standards for the Web
- Domain independent examples
  - Ontoclean
  - Sumo

14

Ontology Technology

- "Ontology" covers a range of things
  - Controlled vocabularies – e.g. MeSH
    - Linguistic structures – e.g. WordNet
  - Hierarchies (with bells and whistles) – e.g. Gene Ontology
  - Frame representations – e.g. FMA
  - Description logic formalisms – Snomed-CT, GALEN, OWL-DL based ontologies
  - Philosophically inspired e.g. Ontoclean and SUMO

Outline

- What are Ontologies
- Semantic Web
- OWL
- Best Practice

OWL
The Web Ontology Language

- W3C standard
- Collision of DAML (frames) and Oil (DLs in Frame clothing)
- Three 'flavours'
  - OWL-Lite –simple but limited
  - OWL-DL – complex but deliverable (real soon now)
  - OWL-Full – fully expressive but serious logical/computational problems
    - Russel Paradox etc etc
  - All layered (awkwardly) on RDF Schema
- Still work in progress – see Semantic Web Best Practices & Deployment Working Group (SWBP)

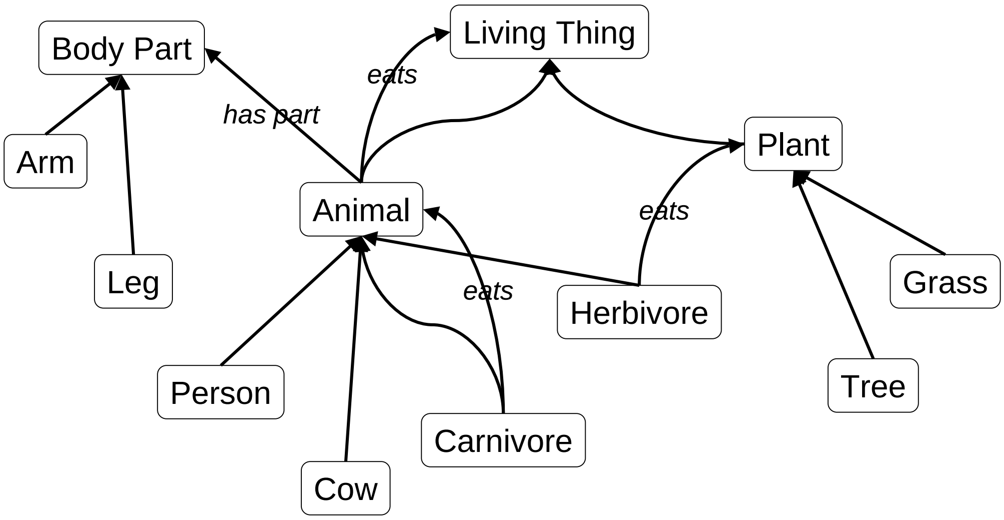
17

Note on syntaxes for OWL

- Three official syntaxes + Protégé-OWL syntax
  - Abstract syntax -Specific to OWL
  - N3 -OWL & RDF
-used in all SWBP documents
  - XML/RDF -very verbose
  - Protégé-OWL -Compact, derived from DL syntax
- This tutorial uses simplified abstract syntax
  - someValuesFrom → *some*
  - allValuesFrom → *only*
  - intersectionOf → AND
  - unionOf → OR
  - complementOf → not
- Protégé/OWL can generate all syntaxes

18

A simple ontology: Animals



19

Description Logics

- What the logicians made of Frames
  - Greater expressivity and semantic precision
    - Compositional definitions
      - "Conceptual Lego" – define new concepts from old
- To allow automatic classification & consistency checking
  - The mathematics of classification is tricky
    - Some seriously counter-intuitive results
      - The basics are simple – devil in the detail

20

Description Logics

- Underneath:
  - computationally tractable subsets of first order logic
- Describes relations between Concepts/Classes
  - Individuals secondary
    - ***DL Ontologies are NOT databases!***

21

Description Logics: A brief history

- Informal Semantic Networks and Frames (pre 1980)
  - Wood: *What's in a Link*; Brachman *What IS-A is and IS-A isn't*.
- First Formalisation (1980)
  - Bobrow *KRL*, Brachman: *KL-ONE*
- All useful systems are intractable (1983)
  - Brachman & Levesque: *A fundamental tradeoff*
    - Hybrid systems: T-Box and A-Box
- All tractable systems are useless (1987-1990)
  - Doyle and Patel: *Two dogmas of Knowledge Representation*

22

A brief history of KR

- ‘Maverick’ incomplete/intractable logic systems (1985-90)
  - GRAIL, LOOM, Cyc, Apelon, …,
- Practical knowledge management systems based on frames
  - Protégé
- The German School: Description Logics (1988-98)
  - Complete decidable algorithms using tableaux methods (1991-1992)
  - Detailed catalogue of complexity of family – “alphabet soup of systems”
- Optimised systems for practical cases (1996-)
- Emergence of the Semantic Web
  - Development of DAML (frames), OIL (DLs) → DAML+OIL → OWL
    - **Development of Protégé-OWL**
    - **A dynamic field – constant new developments & possibilities**

23

Outline

- What are Ontologies
- Semantic Web
- OWL
- “Best Practice”
  - Semantic Web Best Practice & Deployment Working Group (SWBP)

24

Why the “Best Practice working Group”?

- There is no established “best practice”
  - It is new; We are all learning
  - A place to gather experience
  - A catalogue of things that work – Analogue of Software Patterns
    - Some pitfalls to avoid
      - ***…but there is no one way***
- Learning to build ontologies
  - Too many choices
    - Need starting points for gaining experience
- Provide requirements for tool builders

25

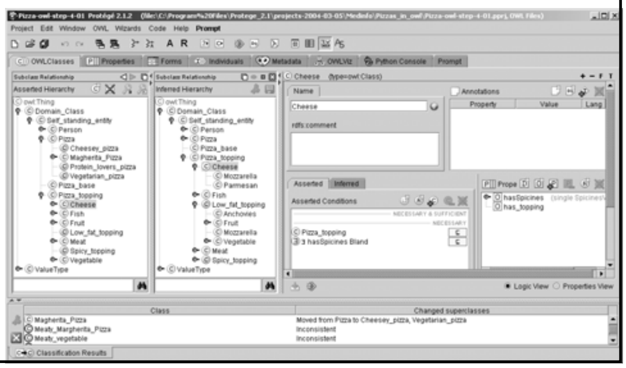
Contributing to “best practice”

- Please give us feedback
  - Your questions and experience
    - On the SW in general: **semanticweb@yahoogroups.com**
    - For specific feedback to SWBP
      - Home & Mail Archive: **http://www.w3.org/2001/sw/BestPractices/** **public-swbp-wg@w3.org**

26

Protégé OWL: New tools for ontologies

- Transatlantic collaboration
- Implement robust OWL environment within PROTÉGÉ framework
- Shared UI components
- Enables hybrid working



27

Protégé-OWL & CO-ODE

- Joint work: Stanford & U Manchester + Southampton & Epistemics
  - Please give us feedback on tools – mailing lists & forums at:
    - **protege.stanford.edu**
    - **www.co-ode.org**
- Don’t beat your head against a brick wall!
  - Look to see if others have had the same problem; If not…
  - ***ASK!***
    - ***We are all learning.***

28

Part II – Creating an ontology Useful patterns

- *Upper ontologies & Domain ontologies*
- Building from trees and untangling
- Using a classifier
- Closure axioms
- Specifying Values
- n-ary relations
- Classes as values – using the ontology
- Part-whole relations

29

Upper Ontologies

- Ontology Schemas
  - High level abstractions to constrain construction
    - e.g. There are “Objects” & “Processes”
  - Highly controversial
    - Sumo, Dolce, Onions, GALEN, SBU,…
  - Needed when you work with many people together
  - NOT in this tutorial – a different tutorial

30

Domain Ontologies

- Concepts specific to a field
  - Diseases, animals, food, art work, languages, …
  - The place to start
    - Understand ontologies from the bottom up
      - Or middle out
- Levels
  - Top domain ontologies – the starting points for the field
    - Living Things, Geographic Region, Geographic_feature
  - Domain ontologies – the concepts in the field
    - Cat, Country, Mountain
  - Instances – the things in the world
    - Felix the cat, Japan, Mt Fuji

31

Part II – Useful Patterns (continued)

- Upper ontologies & Domain ontologies
- *Building from trees and untangling*
- *Using a classifier*
- *Closure axioms & Open World Reasoning*
- Specifying Values
- n-ary relations
- Classes as values – using the ontology

32

Example: Animals & Plants

- Dog
- Cat
- Cow
- Person
- Tree
- Grass
- Herbivore
- Male
- Female

- Carnivore
- Plant
- Animal
- Fur
- Child
- Parent
- Mother
- Father

- Dangerous
- Pet
- Domestic Animal
- Farm animal
- Draft animal
- Food animal
- Fish
- Carp
- Goldfish

33

Example: Animals & Plants

- Dog
- Cat
- Cow
- Person
- Tree
- Grass
- Herbivore
- Male
- Female

- Carnivore
- Plant
- Animal
- Fur
- Child
- Parent
- Mother
- Father

- Healthy
- Pet
- Domestic Animal
- Farm animal
- Draft animal
- Food animal
- Fish
- Carp
- Goldfish

34

Choose some main axes
Add abstractions where needed; identify relations;
Identify definable things, make names explicit

- Living Thing
  - Animal
    - Mammal
      - Cat
      - Dog
      - Cow
      - Person
    - Fish
      - Carp
      - Goldfish
  - Plant
    - Tree
    - Grass
    - Fruit

- Modifiers
  - domestic
    - pet
    - Farmed
      - Draft
      - Food
  - Wild
  - Health
    - healthy
    - sick
  - Sex
    - Male
    - Female
  - Age
    - Adult
    - Child

- Relations
  - eats
  - owns
  - parent-of
  - …
- Definable
  - Carinvore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

35

Reorganise everything but "definable" things into pure trees – these will be the "primitives"

- Primitives
  - Living Thing
    - Animal
      - Mammal
        - Cat
        - Dog
        - Cow
        - Person
      - Fish
        - Carp
          Goldfish
    - Plant
      - Tree
      - Grass
      - Fruit

- Modifiers
  - **Domestication**
    - **Domestic**
    - **Wild**
  - **Use**
    - **Draft**
    - **Food**
    - **pet**
  - **Risk**
    - **Dangerous**
    - **Safe**
  - **Sex**
    - **Male**
    - **Female**
  - **Age**
    - **Adult**
    - **Child**

- Relations
  - eats
  - owns
  - parent-of
  - …
- Definables
  - Carnivore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

36

Set domain and range constraints for properties

- Animal *eats* Living_thing
  - *eats* domain: Animal;
    range: Living_thing
- Person *owns* Living_thing except person
  - *owns* domain: Person
    range: Living_thing & not Person
- Living_thing *parent_of* Living_thing
  - *parent_of:* domain: Animal
    range: Animal

37

Define the things that are definable from the primitives and relations

- Parent =
  Animal and *parent_of* some Animal
- Herbivore=
  Animal and *eats* only Plant
- Carnivore =
  Animal and *eats* only Animal

38

Which properties can be filled in at the class level now?

- What can we say about *all* members of a class
  - *eats* is the only one
    - *All* cows eat *some* plants
    - *All* cats eat *some* animals
    - *All* dogs eat *some* animals &
      eat *some* plants

39

Fill in the details
(can use property matrix wizard)

Class	eats
Cat	Ⓒ Animal
Dog	Ⓒ Animal
Cow	Ⓒ Grass Ⓒ Leafy_plant
Person	Ⓒ Animal Ⓒ Plant

40

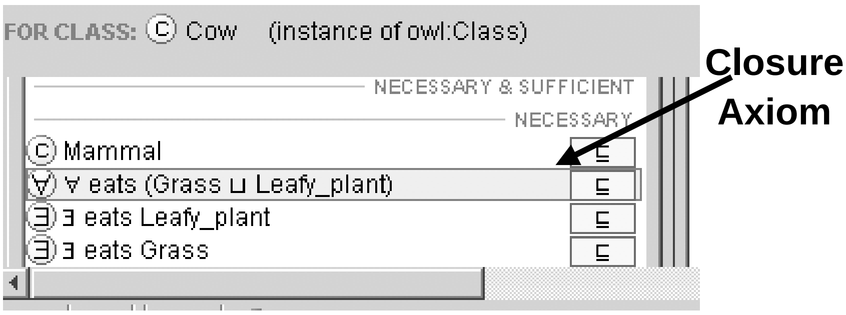
Check with classifier

- Cows should be Herbivores
  - Are they? why not?
    - What have we said?
      - Cows are animals and, *amongst other things*,
        eat *some* grass and
        eat some leafy_plants
    - What do we need to say:
      Closure axiom
      - Cows are animals and, *amongst other things,*
        eat *some* plants and eat *only* plants

41

Closure Axiom

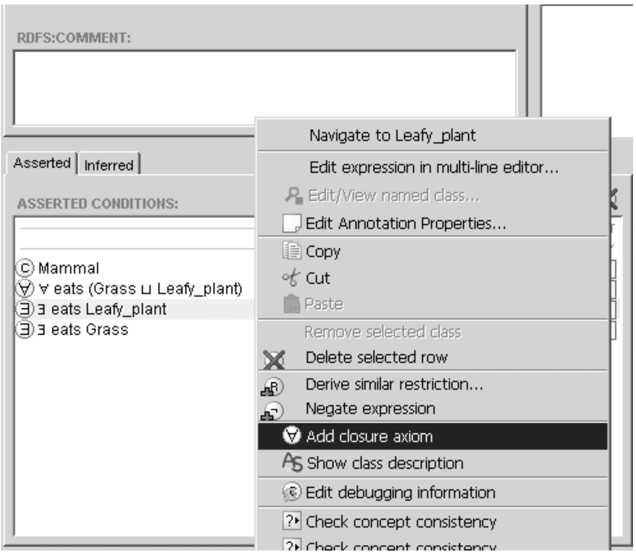
- Cows are animals and, *amongst other things,* eat *some* plants and eat *only* plants



42

In the tool

- Right mouse button short cut for closure axioms
  - for any existential restriction



43

Open vs Closed World reasoning

- Open world reasoning
  - Negation as contradiction
    - Anything might be true unless it can be proven false
      - Reasoning about *any world consistent with this one*
- Closed world reasoning
  - Negation as failure
    - Anything that cannot be found is false
      - Reasoning about *this world*

44

Normalisation and Untangling

Let the reasoner do multiple classification

- Tree
  - Everything has just one parent
    - A 'strict hierarchy'
- Directed Acyclic Graph (DAG)
  - Things can have multiple parents
    - A 'Polyhierarchy'
- Normalisation
  - Separate primitives into disjoint trees
  - Link the trees with restrictions
    - Fill in the values

45

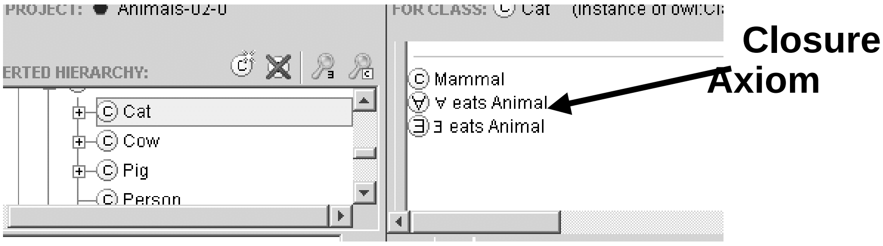
Tables are easier to manage than DAGs / Polyhierarchies

Class	eats
Cat	Animal
Cow	Grass Leafy_plant
Pig	Animal Plant
Person	
Dog	Animal

…and get the benefit of inference:
Grass and Leafy_plants are both kinds of Plant

46

Remember to add any closure axioms

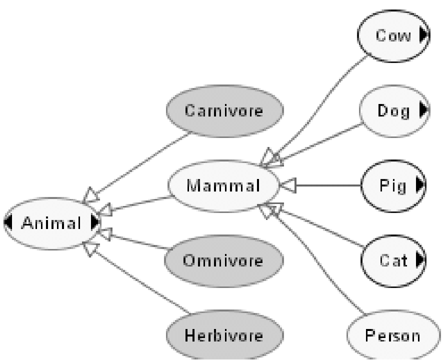


Then let the reasoner do the work

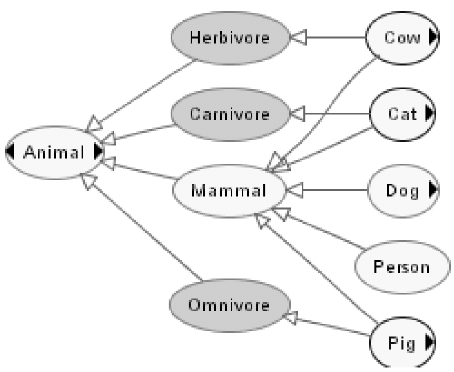
47

Normalisation: From Trees to DAGs

- Before classification
  - A tree



- After classification
  - A DAG
    - **D**irected **A**cyclic **G**raph



48

Part II – Useful Patterns (continued)

- Upper ontologies & Domain ontologies
- Building from trees and untangling
- Using a classifier
- Closure axioms & Open World Reasoning
- *Specifying Values*
- n-ary relations
- Classes as values – using the ontology

49

Examine the modifier list

- Modifiers
  - **Domestication**
    - **Domestic**
    - **Wild**
  - **Use**
    - **Draft**
    - **Food**
  - **Risk**
    - **Dangerous**
    - **Safe**
  - **Sex**
    - **Male**
    - **Female**
  - **Age**
    - **Adult**
    - **Child**

- Identify modifiers that are mutually exclusive
  - Domestication
  - Risk
  - Sex
  - Age
- Make meaning precise
  - Age → Age_group
- NB Uses are not mutually exclusive
  - Can be both a draft and a food animal

50

Extend and complete lists of values

- Modifiers
  - **Domestication**
    - **Domestic**
    - **Wild**
    - **Feral**
  - **Risk**
    - **Dangerous**
    - **Risky**
    - **Safe**
  - **Sex**
    - **Male**
    - **Female**
  - **Age**
    - **Infant**
    - **Toddler**
    - **Child**
    - **Adult**
    - **Elderly**

- Identify modifiers that are mutually exclusive
  - Domestication
  - Risk
  - Sex
  - Age
- Make meaning precise
  - Age → Age_group
- NB Uses are not mutually exclusive
  - Can be both a draft and a food animal

Note any hierarchies of values

- Modifiers
  - **Domestication**
    - **Domestic**
    - **Wild**
    - **Feral**
  - **Risk**
    - **Dangerous**
    - **Risky**
    - **Safe**
  - **Sex**
    - **Male**
    - **Female**
  - **Age**
    - **Child**
      - **Infant**
      - **Toddler**
    - **Adult**
      - **Elderly**

- Identify modifiers that are mutually exclusive
  - Domestication
  - Risk
  - Sex
  - Age
- Make meaning precise
  - Age → Age_group
- NB Uses are not mutually exclusive
  - Can be both a draft and a food animal

Specify Values for each

- Value partitions
  - Classes that partition a Quality
    - The disjunction of the partition classes equals the quality class
- Symbolic values
  - Individuals that enumerate all states of a Quality
    - The enumeration of the values equals the quality class

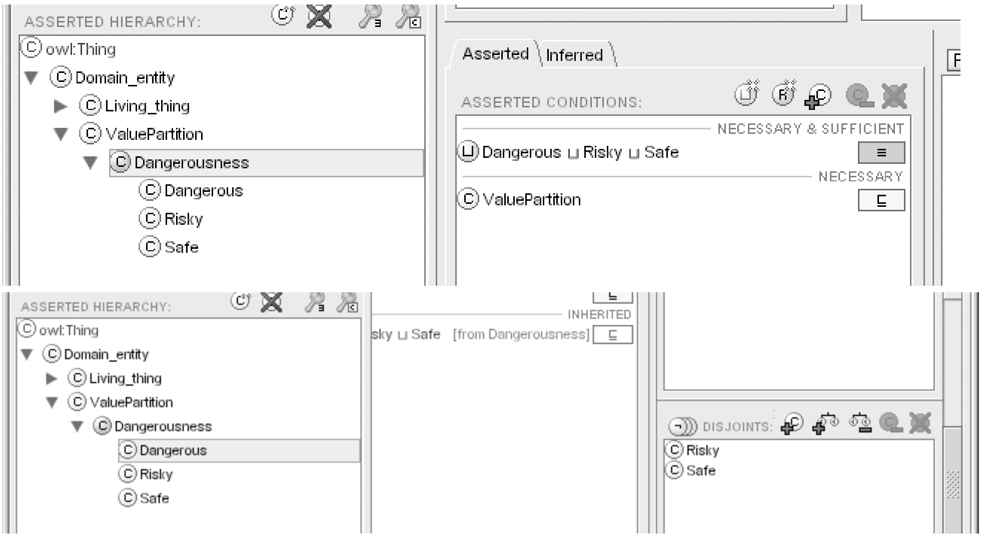
53

Value Partitions: example Dangerousness

- A parent quality – Dangerousness
- Subqualities for each degree
  - Dangerous, Risky, Safe
- All subqualities disjoint
- Subqualities ‘cover’ parent quality
  - Dangerousness = Dangerous OR Risky OR Safe
- A functional property has_dangerousness
  - Range is parent quality, e.g. Dangerousness
  - Domain must be specified separately
- Dangerous_animal =
  Animal *and* has_dangerousness *some* Dangerous

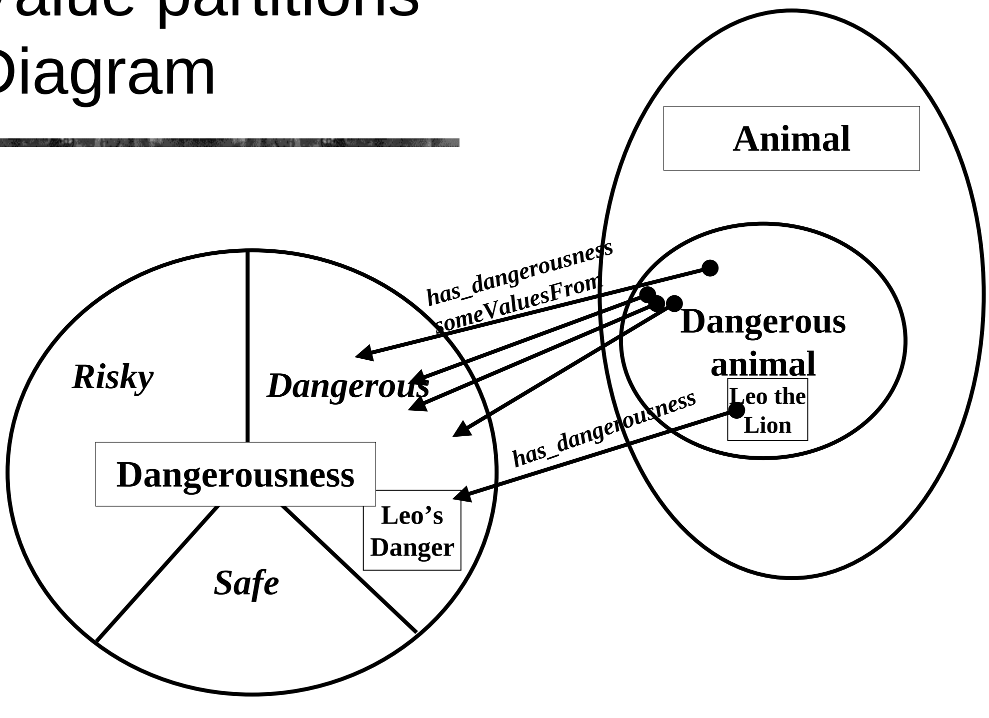
54

as created by Value Partition wizard



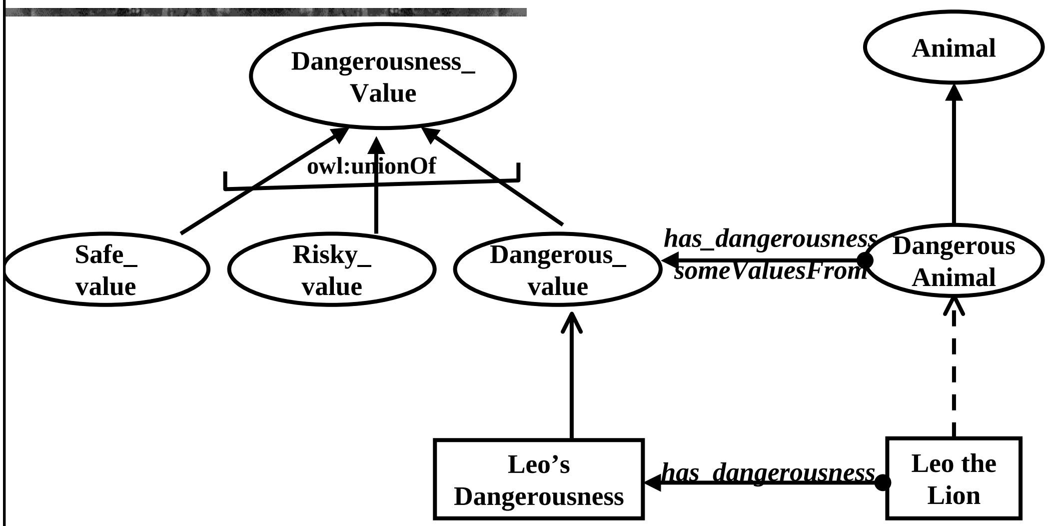
55

Value partitions Diagram



56

Value partitions UML style



57

Values as individuals: Example Sex

- There are only two sexes
  - Can argue that they are things
    - "Administrative sex" definitely a thing
    - "Biological sex" is more complicated

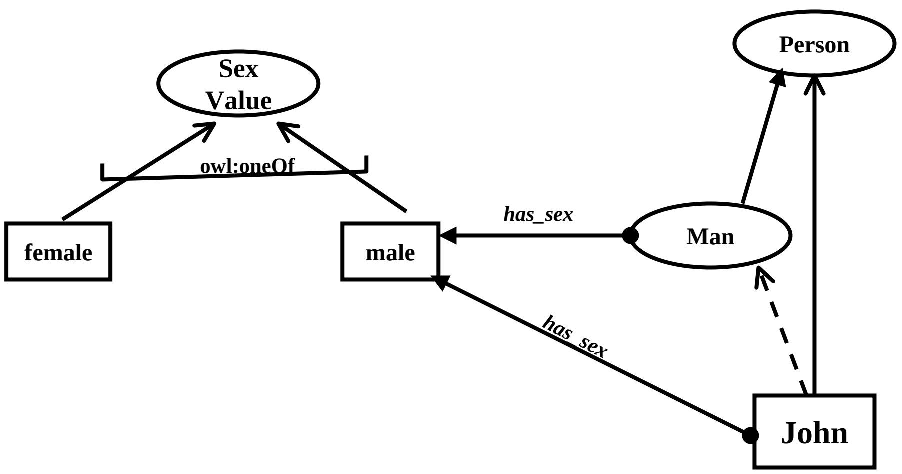
58

Value sets for specifying values

- A parent quality – Sex_value
- Individuals for each value
  - male, female
- Values all different (NOT assumed by OWL)
- Value type is enumeration of values
  - Sex_value = {male, female}
- A functional property has_sex
  - Range is parent quality, e.g. Sex_value
  - Domain must be specified separately
- Male_animal =
  Animal *and* has_sex *is* Dangerous

59

Value sets UML style



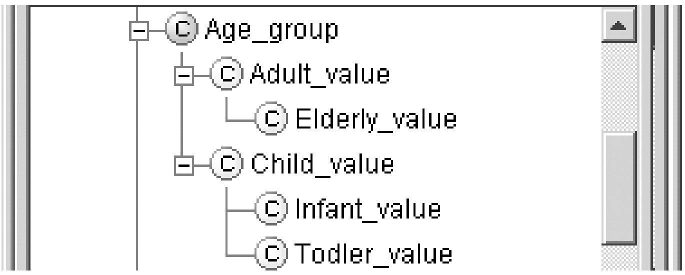
60

Issues in specifying values

- Value Partitions
  - Can be subdivided and specialised
  - Fit with philosophical notion of a quality space
  - Require interpretation to go in databases as values
    - **in theory but rarely considered in practice**
  - Work better with existing classifiers in OWL-DL
- **Value Sets**
  - Cannot be subdivided
  - Fit with intuitions
  - More similar to data bases – no interpretation
  - Work less well with existing classifiers

61

Value partitions – practical reasons for subdivisions

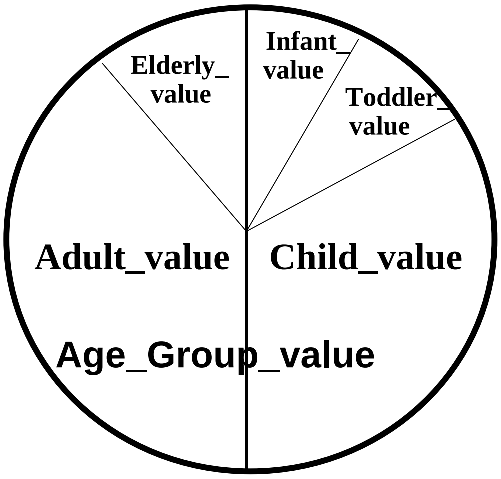


- "All elderly are adults"
- "All infants are children"
- etc.

- See also "Normality_status" in http://www.cs.man.ac.uk/~rector/ontologies/mini-top-bio
  - One can have complicated value partitions if needed.

62

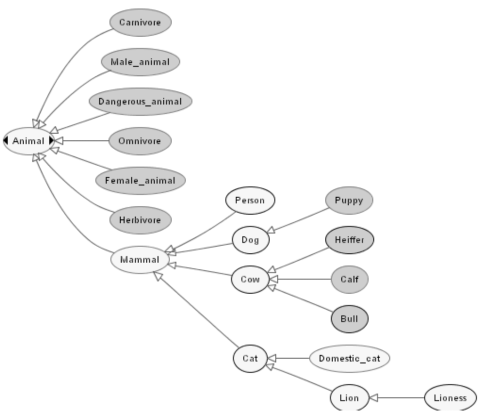
Picture of subdivided value partition



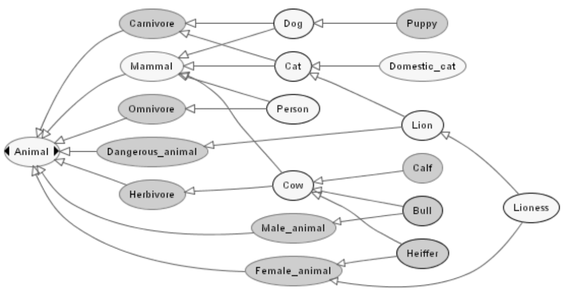
63

More defined kinds of animals

- Before classification, trees



- After classification, DAGs



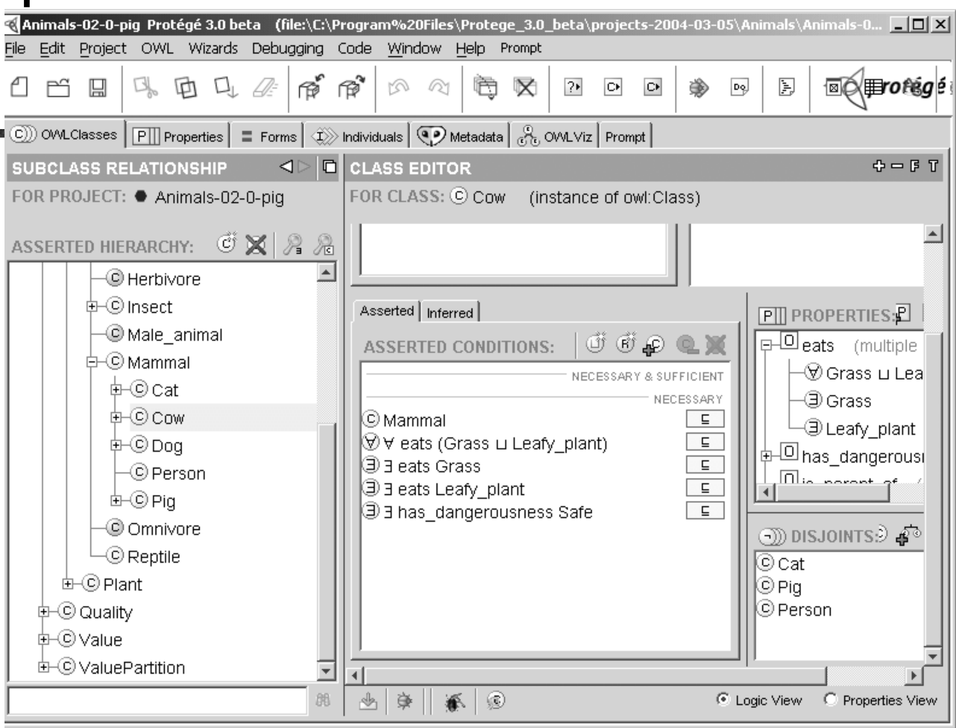
64

Part III – Hands On

- Be sure you have installed the software
  - (See front page)
- Open Animals-tutorial-step-1

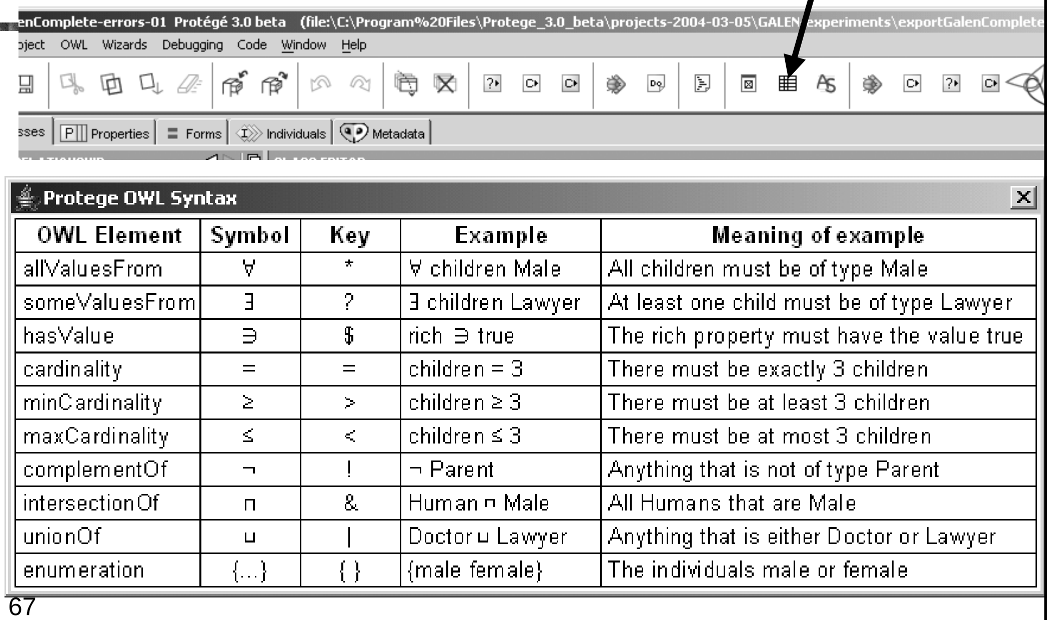
65

Explore the interface



66

Protégé Syntax



OWL Element	Symbol	Key	Example	Meaning of example
allValuesFrom	∀	*	∀ children Male	All children must be of type Male
someValuesFrom	∃	?	∃ children Lawyer	At least one child must be of type Lawyer
hasValue	∋	$	rich ∋ true	The rich property must have the value true
cardinality	=	=	children = 3	There must be exactly 3 children
minCardinality	≥	>	children ≥ 3	There must be at least 3 children
maxCardinality	≤	<	children ≤ 3	There must be at most 3 children
complementOf	¬	!	¬ Parent	Anything that is not of type Parent
intersectionOf	⊓	&	Human ⊓ Male	All Humans that are Male
unionOf	⊔	|	Doctor ⊔ Lawyer	Anything that is either Doctor or Lawyer
enumeration	{...}	{ }	{male female}	The individuals male or female

67

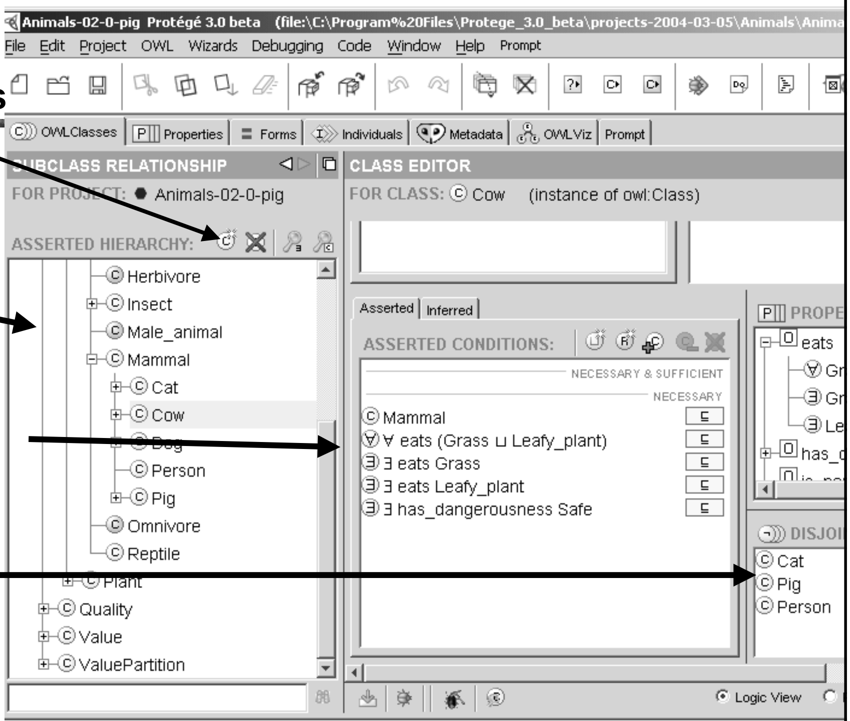
Explore the interface

ew Subclass icon

Asserted Hierarchy

Class escription

Disjoint Classes



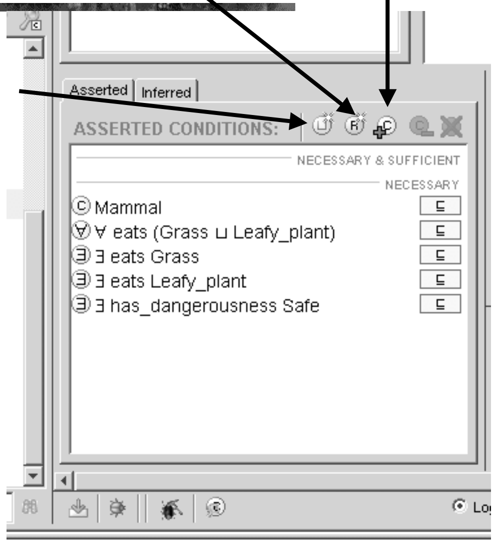
68

Explore the interface

New ***restriction***

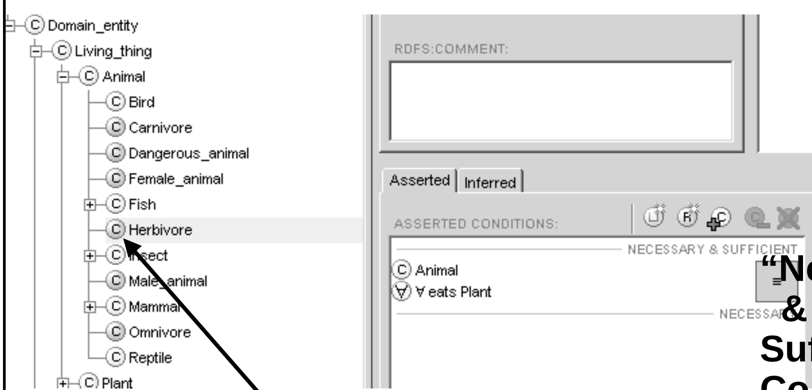
Add ***superclass***

New ***expression***



Description "Necessary Conditions"

Explore the interface

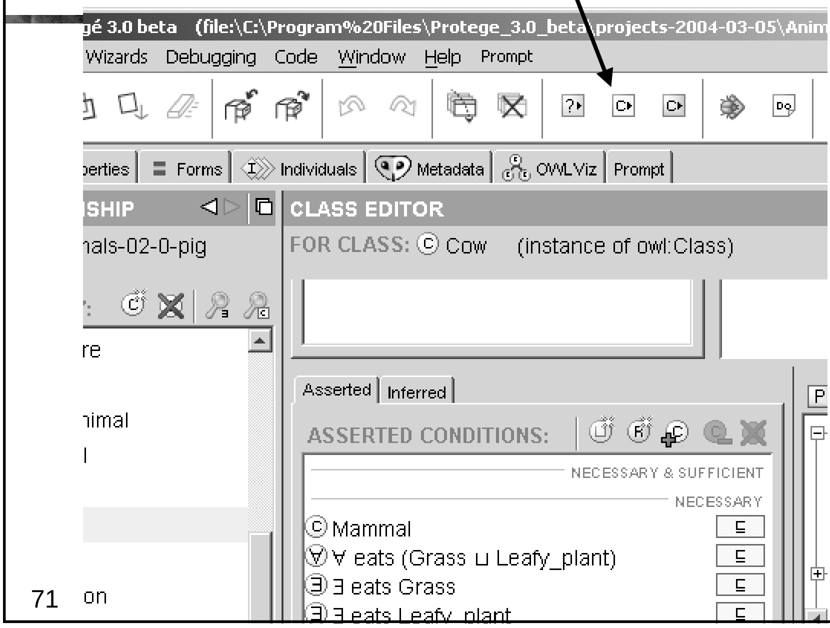


Definition "Necessary & Sufficient Conditions"

"Defined class" (orange/red circle)

Explore the interface

Classify button (racer must be running



71

Exercise 1

- Create a new animal, a Elephant and a Ape
  - Make them disjoint from the other animals
  - Make the ape an omnivore
    - eats animals and plants
  - Make the sheep a herbivore
    - eats plants and only plants

72

Exercise 1b: Classification

- Check it with the classifier
- Is Sheep classified under Herbivore
  - If not, have you forgot the closure axiom?
- Did it all turn red?
  - Do you have too many disjoint axioms?

73

Exercise 1c: checking disjoints – make things that should be inconsistent

- Create a Probe_Sheep_and_Cow that is a kind of both Sheep and Cow
- Create a Probe_Ape_and_Man that is a kind of both Ape and Man
- Run the classifier
- Did both probes turn red?
  - If not, check the disjoints

74

Exercise 2: A new value partition

- Create a new value partition
  - Size_partition
    - Big
    - Medium
    - Small
- Describe
  - Lions, Cows, and Elephants as Big
    domestic_cat as Small
    the rest Medium

75

Exercise 2b

- Define Big_animal and Small_animal
  - Does the classification work
- Extra
  - Make a subdivision of Big for Huge and make elephants Huge
    - Do elephants still classify as “Big Animal

76

Part IV – Patterns: n-ary relations

- Upper ontologies & Domain ontologies
- Building from trees and untangling
- Using a classifier
- Closure axioms & Open World Reasoning
- Specifying Values
- *n-ary relations*
- Classes as values – using the ontology

77

Saying something about a restriction

- Not just
  - that an animal is dangerous,
  - but why
  - And how dangerous
  - And how to avoid
- But can say nothing about properties
  - except special thing
    - Super and subproperties
    - Functional, transitive, symmetric

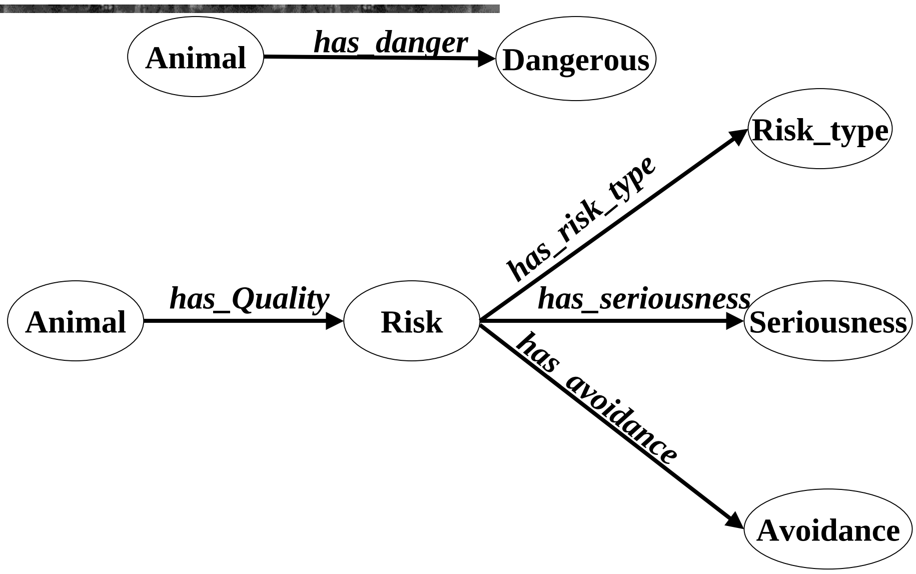
78

Re-representing properties as classes

- To say something about a property it must be re-represented as a class
  - property:has_danger → Class: Danger
    - plus property: Thing has_quality Danger
    - plus properties: Danger has_reason
      has_risk
      has_avoidance_measure
  - Sometimes called “reification”
    - But “reification” is used differently in different communities

79

Re-representing the property has_danger as the class Risk



80

Lions are dangerous

- All lions pose a deadly risk of physical attack that can be avoided by physical separation
- All lions have the quality risk that is
  - of type some physical attack
  - of seriousness some deadly
  - has avoidance means some physical separation

81

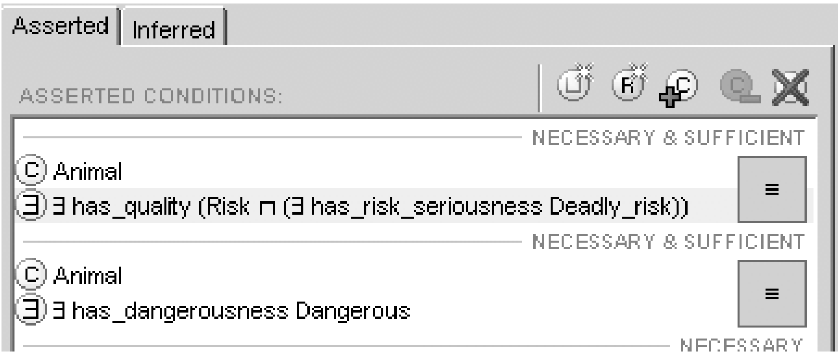
Can add a second definition of Dangerous Animal

- A dangerous animal is any animal that has the quality Risk that is Deadly
  - or
- Dangerous_animal =
  - Animal
    has_quality *some*
    (Risk AND has_seriousness *some* Deadly)
  - [NB: “that” paraphrases as “AND”]

82

In the tool

- Dangerous_animal =
  - Animal
    has_quality *some*
    (Risk AND has_seriousness *some* Deadly)



83

This says that

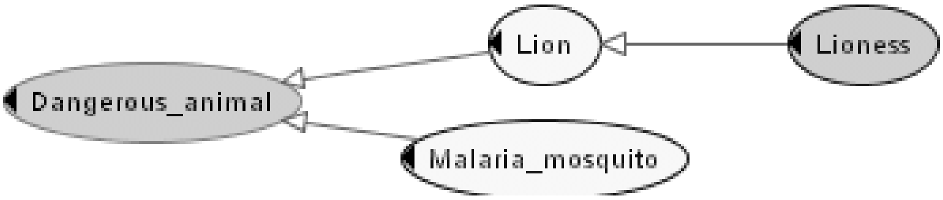
- Any animal that is Dangerous

  is also

  An animal that has the quality
  Dangerousness with the seriousness Deadly

84

Anopheles Mosquitos now count as dangerous



- Because they have a deadly risk of carrying disease

85

Multiple definitions are dangerous

- Better to use one way or the other
  - Otherwise keeping the two ways consistent is difficult
  - … but ontologies often evolve so that simple Properties are re-represented as Qualities

86

Often have to re-analyse

- What do we mean by “Dangerous”
  - How serious the danger?
  - How probable the danger?
  - Whether from individuals (Lions) or the presence or many (Mosquitos)?
- Moves to serious questions of “ontology”
  - The information we really want to convey
    - Often a sign that we have gone to far
      - So we will stop

87

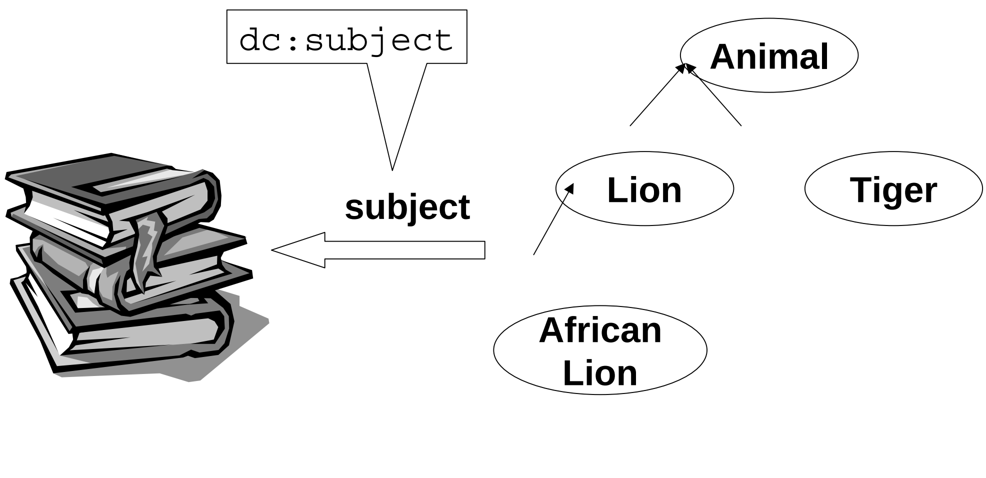
88

Part V – Patterns: Classes as values

- Upper ontologies & Domain ontologies
- Building from trees and untangling
- Using a classifier
- Closure axioms & Open World Reasoning
- Specifying Values
- n-ary relations
- *Classes as values – using the ontology*
- Part-whole relations

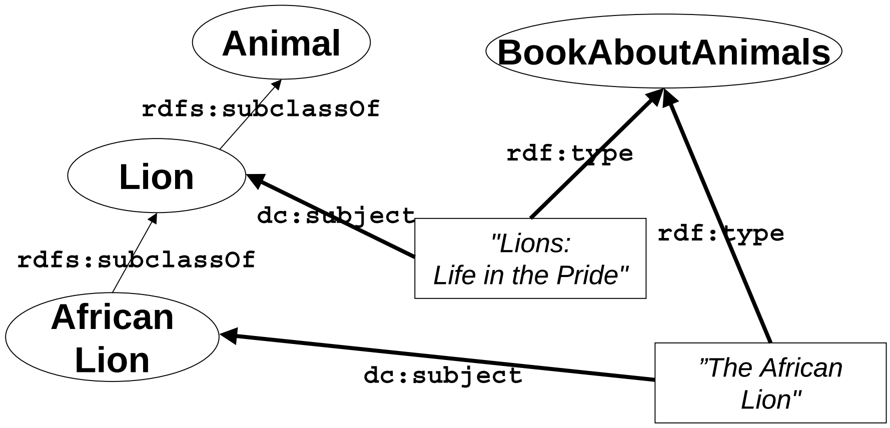
89

Using Classes as Property Values



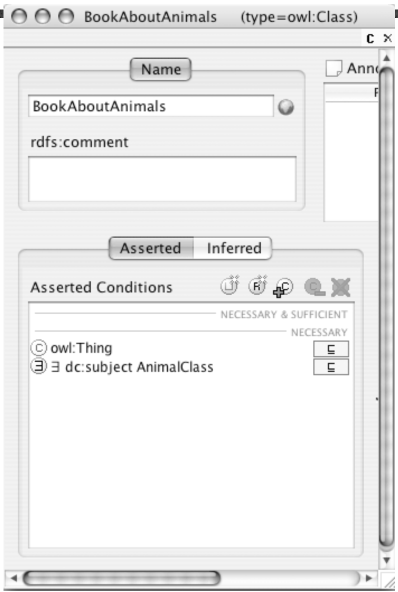
90

Using Classes Directly As Values



91

Representation in Protégé



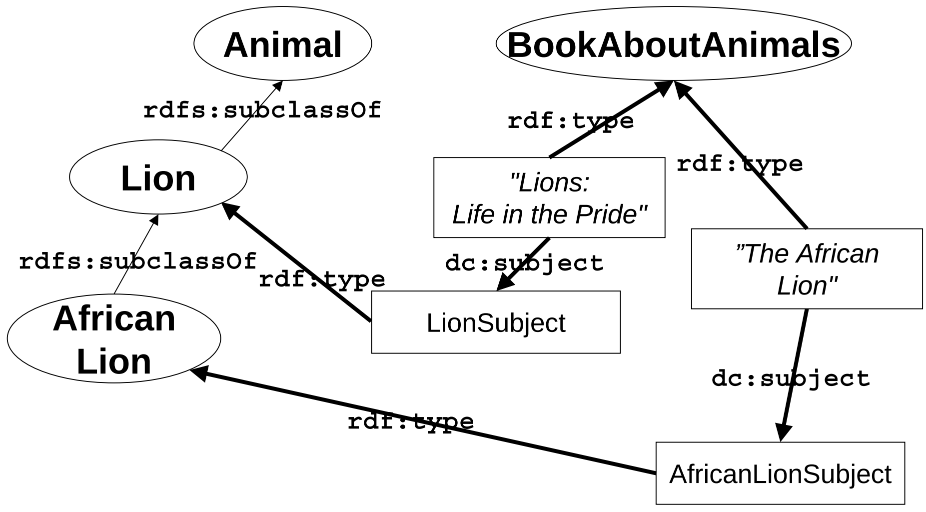
92

Approach 1: Considerations

- Compatible with OWL Full and RDF Schema
- Outside OWL DL

93

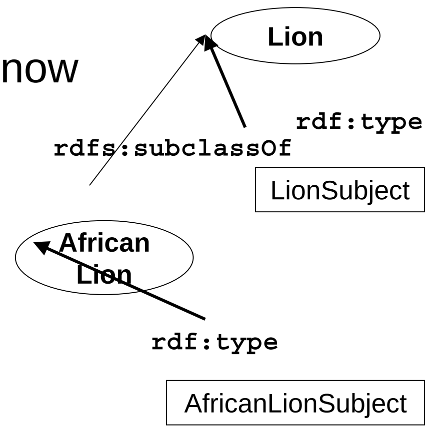
Approach 2: Hierarchy of Subjects



94

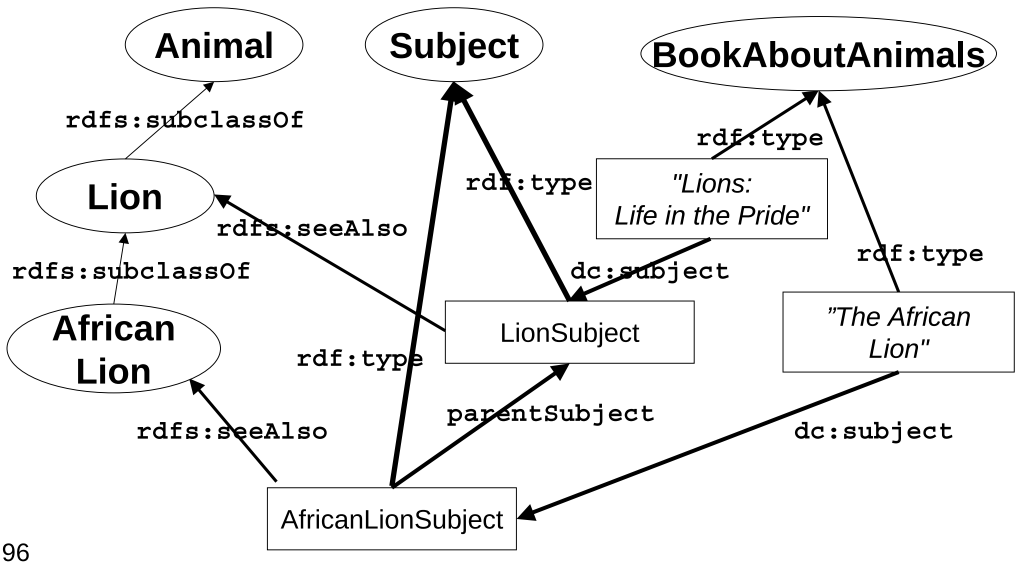
Hierarchy of Subjects: Considerations

- Compatible with OWL DL
- Instances of class Lion are now subjects
- No direct relation between LionSubject and AfricalLionSubject
- Maintenance penalty



95

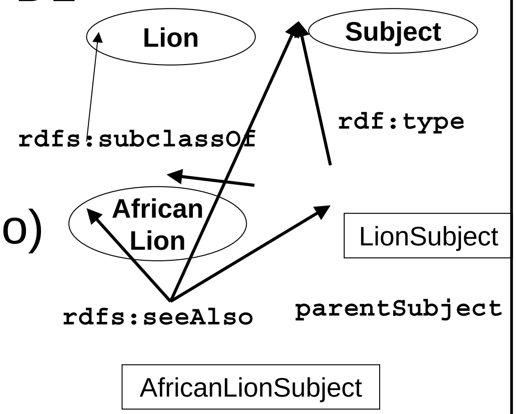
Hierarchy of Subjects



96

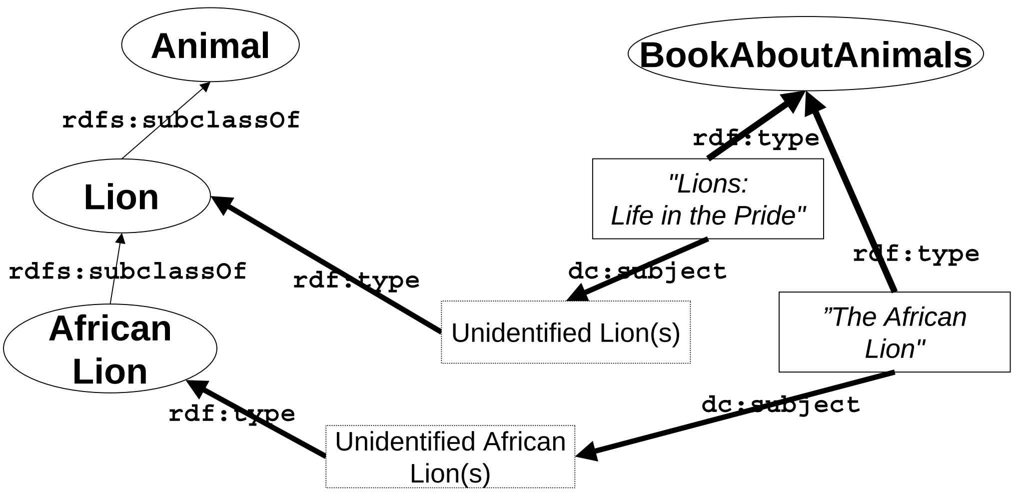
Hierarchy of Subjects: Considerations

- Compatible with OWL DL
- Subject hierarchy (terminology) is independent of class hierarchy (rdfs:seeAlso)
- Maintenance penalty



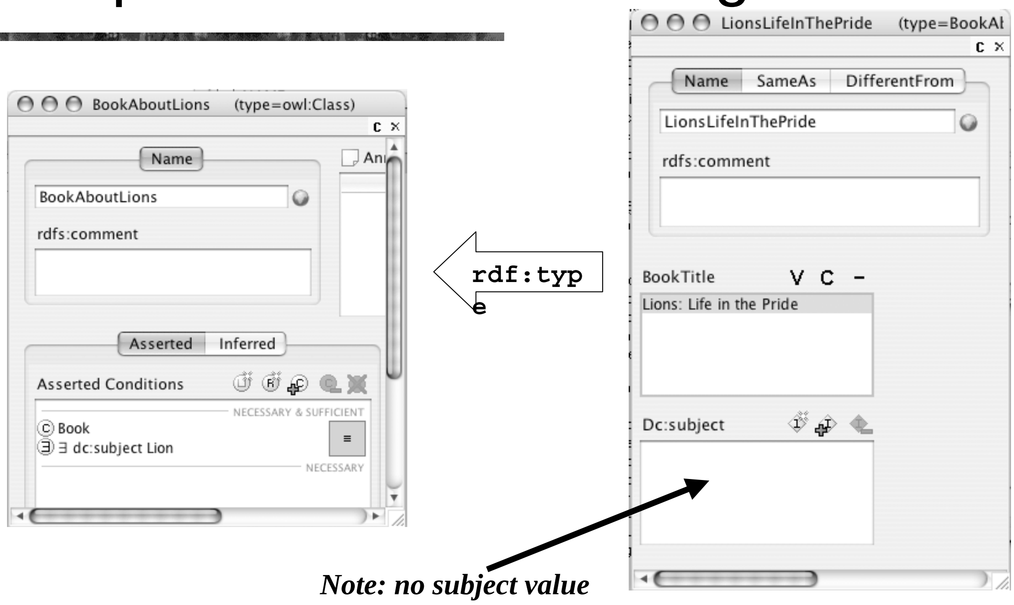
97

Using members of a class as values



98

Representation in Protege



99

Considerations

- Compatible with OWL DL
- Interpretation: the subject is one or more specific lions, rather than the Lion class
- Can use a DL reasoner to classify specific books

100

Part VI – Patterns: Part-whole relations

- Upper ontologies & Domain ontologies
- Building from trees and untangling
- Using a classifier
- Closure axioms & Open World Reasoning
- Specifying Values
- n-ary relations
- Classes as values – using the ontology
- *Part-whole relations*

101

Part-whole relations *One method: NOT a SWBP draft*

- How to represent part-whole relations in OWL is a commonly asked question
- SWBP will put out a draft.
- This is one approach that will be proposed
  - It has been used with classes
  - It has no official standing
  - It is presented for information only

102

Part Whole relations

- OWL has no special constructs
  - But provides the building blocks
- Transitive relations
  - Finger is_part_of Hand
    Hand is_part_of Arm
    Arm is_part_of Body
    - →
  - Finger is_part_of Body

103

Many kinds of part-whole relations

- Physical parts
  - hand-arm
- Geographic regions
  - Hiroshima - Japan
- Functional parts
  - cpu – computer

- See Winston & Odell
  Artale
  Rosse

104

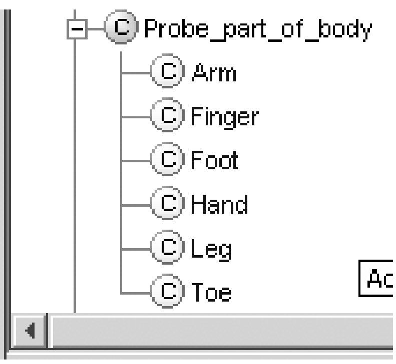
Simple version

- One property *is_part_of*
  - transitive
    - finger is_part_of *some* Hand
      Hand is_part_of *some* Arm
      Arm is_part_of some Body

105

Get a simple list

- Probe_part_of_body =
  Domain_category
  is_part_of some Body



- Logically correct
  - But may not be what we want to see
    - The finger is not a kind of Hand
      - It is a part of the hand

106

Injuries, Faults, Diseases, Etc.

- A hand is not a *kind of* a body
  - … but an injury to a hand is a kind of injury to a body
- A motor is not a *kind of* automobile
  - … but a fault in the motor is a kind of fault in the automobile
- And people often expect to see partonomy hierarchies

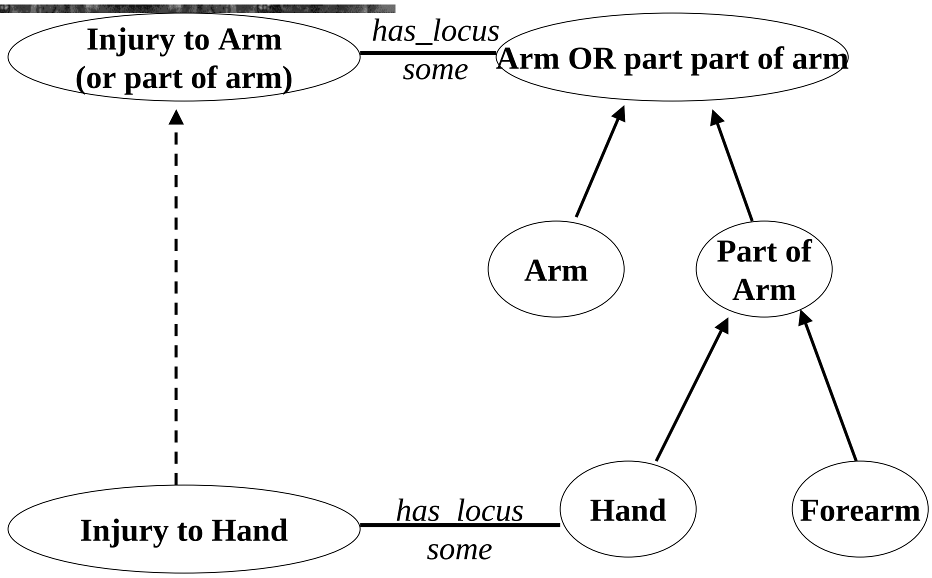
107

Being more precise: “Adapted SEP Triples”

- Body (‘as a whole’)
  - Body
- The Body’s parts
  - is_part_of *some* Body
- The Body and it’s parts
  - Body OR is_part_of *some* body
- Repeat for all parts
  - Use ‘Clone class’ or
  - NB: ‘JOT’ Python plugin is good for this

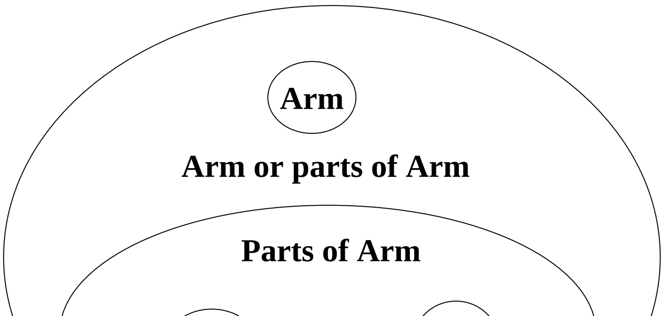
108

Adapted SEP triples: UML like view



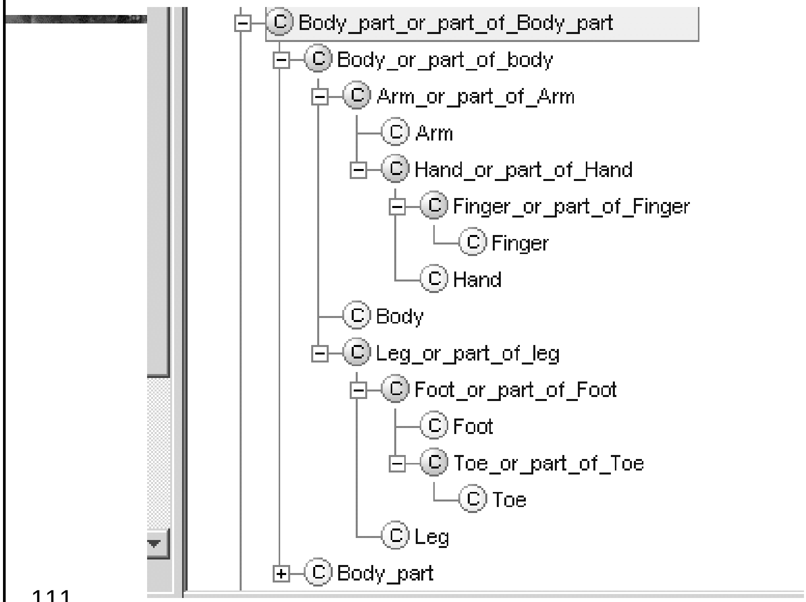
109

Adapted SEP triples: Venn style view



110

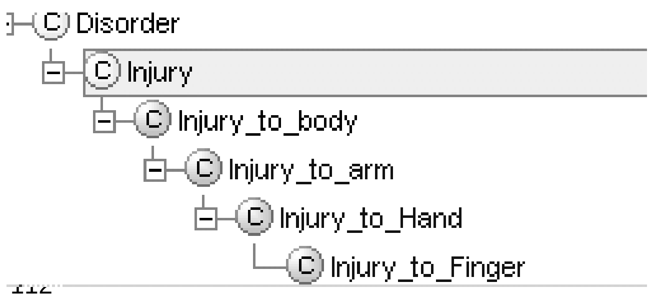
Resulting classification: Ugly to look at, but correct



111

Using part-whole relations: Defining injuries or faults

- Injury_to_Hand =
  Injury has_locus *some* Hand_or_part_of_hand
- Injury_to_Arm =
  Injury has_locus *some* Arm_or_part_of_Arm
- Injury_to_Body =
  Injury has_locus *some* Body_or_part_of_Body



- The expected hierarchy from point of view of anatomy

112

Geographical regions and individuals

- Similar representation possible for individuals but more difficult
  - and less well explored

113

Simplified view: Geographical_regions

- Class: Geographical_region
  - Include countries, cities, provinces, …
    - A detailed ontology would break them down
- Geographical features
  - Include Hotels, Mountains, Islands, etc.
- Properties:
  - Geographical_region *is_subregion_of* Geographical_Region
  - Geogrpahical_feature *has_location* Geographical_Region
  - *is_subregion_of* is transitive
  - Features located in subregions are located in the region.

114

Geographical regions & features are represented as individuals

- Japan, Honshu, Hiroshima, Hiroshima-ken,…
- Mt_Fuji, Hiroshima_Prince_Hotel, …

115

Facts*

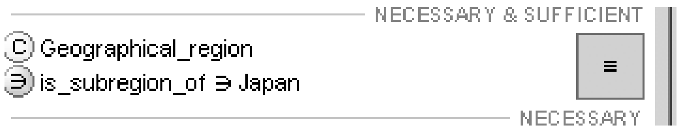
- Honshu is_subregion_of *hasValue* Japan
  Hiroshima-ken is_subregion_of *hasValue* Honshu
  Hiroshima is_subregion_of hasValue Hiroshima-ken
- Mt_Fuji has_location *hasValue* Honsh
  Hiroshima_prince_hotel has_location *hasValue* Hiroshima-ken

*with apologies for any errors in Japanese geography

116

Definitions

- Region_of_Japan =
  Geographical_region *AND*
  is_subregion_of *hasValue* Japan

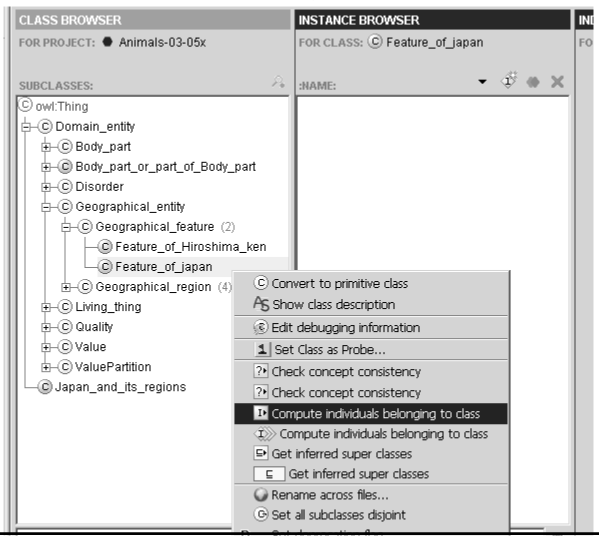


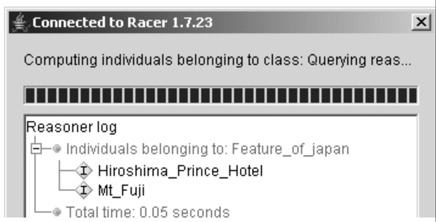
- Feature_of_Japan =
  Geographical_feature *AND*
  (hasLocation *hasValue* Japan *OR*
  hasLocation *hasValue* Region_of_Japan)

117

In tools at this time

- Must ask from right mouse button menu in Individuals tab





- better integration under development

Warning:
Individuals and reasoners

- Individuals only partly implemented in reasoners
  - If results do not work, ask
    - Open World reasoning with individuals is very difficult to implement
  - If it doesn't work, try simulating individuals by classes
  - Large sets of individuals better in "Instance Stores", RDF triple stores, databases, etc that are restricted or closed world
- ***Ontologies are mainly about classes***
  - ***Ontologies are NOT databases***

119

Qualified cardinality constraints

- Use with partonomy
- Use with n-ary relations

120

Cardinality Restrictions

- “All mammals have four limbs”
  - “All Persons have two legs and two arms”
  - “(All mammals have two forelimbs and two hind limbs)”

121

What we would like to say: Qualified cardinality constraints

- Mammal
  has_part cardinality=4 Limb
- Mammal
  has_part cardinality = 2 Forelimb
  has_part cardinality = 2 Hindlimb
- Arm = Forelimb AND is_part_of some Person

122

What we have to say in OWL

- The property *has_part* has subproperties:
  - *has_limb*
    - *has_leg*
    - *has_arm*
    - *has_wing*

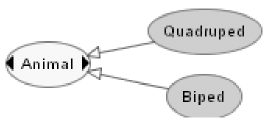
- Mammal, Reptile, Bird *has_limb* cardinality=4
  Person *has_leg* cardinality=2
  Cow, Dog, Pig… *has_leg* cardinality=4
  Bird *has_leg* cardinality=2

- Biped = Animal AND
  has_leg cardinality=2

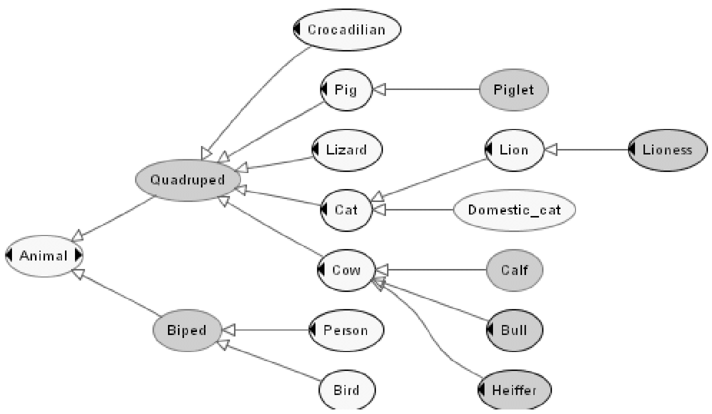
123

Classification of bipeds and quadrupeds

- Before classification



- After classificaiton



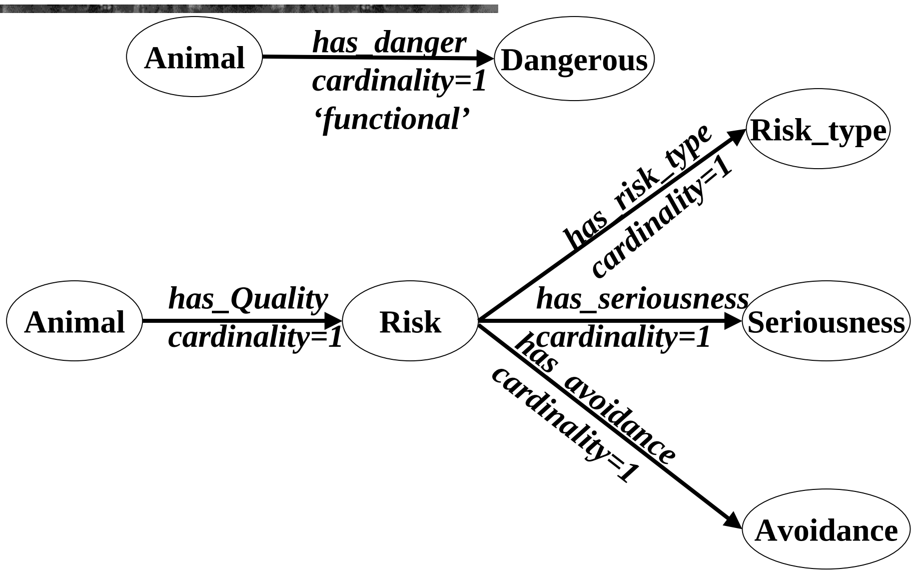
124

Cardinality and n-ary relations

- Need to control cardinality of relations represented as classes
  - An animal can have just 1 "dangerousness"
    - Requires a special subproperty of quality:
      - has_dangerousness_quality cardinality=1

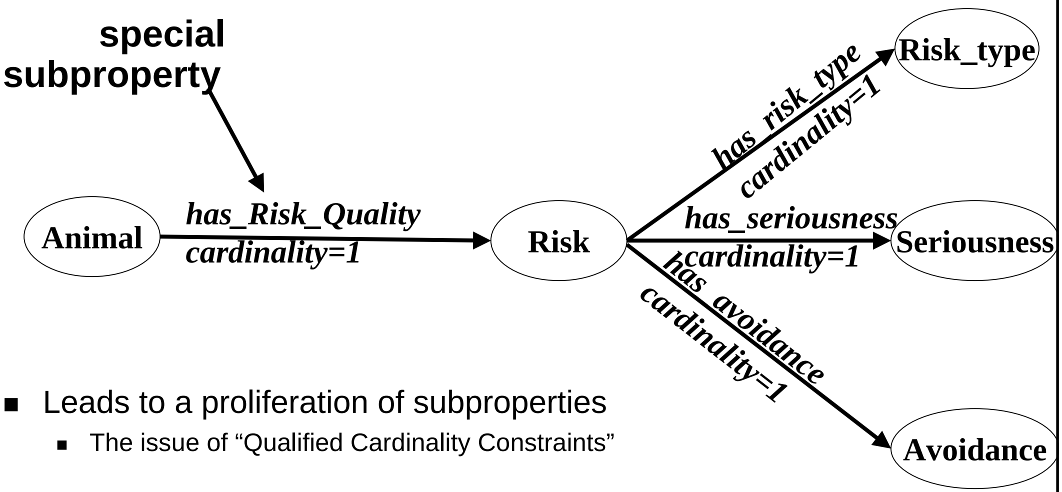
125

Re-representing the property has_danger as the class Risk



126

In OWL must add subproperty for each quality to control cardinality, e.g. *has_risk_quality*



- Leads to a proliferation of subproperties
  - The issue of "Qualified Cardinality Constraints"

127

128

Part VII – Summary

- Upper ontologies & Domain ontologies
- Building from trees and untangling
- Using a classifier
- Closure axioms & Open World Reasoning
- Specifying Values
- n-ary relations
- Classes as values – using the ontology
- Part-whole relations
  - Transitive properties
  - Qualified cardinality restrictions

129

End

- To find out more:
  - **http://www.co-ode.org**
    - Comprehensive tutorial and sample ontologiesxz
  - **http://protege.stanford.org**
    - Subscribe to mailing lists; participate in forums
  - On the SW in general: **semanticweb@yahoogroups.com**
  - For specific feedback to SWBP
    - Home & Mail Archive: **http://www.w3.org/2001/sw/BestPractices/** **public-swbp-wg@w3.org**

130

Part VI – Hands On supplement

- Open Animals-tutorial-step-2

131

Exercise 3: (Advanced supplement)

- Define a new kind of Limb – Wing
- Describe birds as having 2 wings
- Define a Two-Winged_animal
- Does bird classify under Two-Winged_animal?

132