Letting the classifier check your intuitions

Existentials, Universals, & other logical variants
Some, Only, Not, And, Or, etc.

Lab exercise - 3b

Alan Rector & colleagues

To Submit (With Lab 4)

- A text report explaining the results
  - In six part as described here. Each clearly headed and with cross links to the ontology
- An ontology with branches demonstrating each of the issues
  - Or several ontologies clearly labelled to correspond to the report.
- All zipped with the results of the University Untangling exercise with Lab 4.

Closure Restrictions
A standard pattern

- aProperty *SOME* $Class_1$ and
  aProperty *SOME* $Class_2$ and
  aProperty *SOME* $Class_3$ and
  …
  aProperty *SOME* $Class_n$ and
  aProperty *ONLY*
  ($Class_1$ or $Class_2$ or $Class_3$ or … or $Class_n$) } **closure restriction**
- disjoint ($Class_1$ $Class_2$ $Class_3$ … $Class_n$

Part 1: Create an new ontology with the following definitions

- Primitive Concepts
  - Family
    Child
      Girl
      Boy
- Properties
  - has_children
- Defined concepts
  - All_girl_family ≡
    Family & (*restriction* **has_children** *ONLY* Girl)
  - No_child_family ≡
    Family & not (restriction **has_children** *SOME* Top)
  - Zero_child_family ≡
    Family & (restriction **has_children** *max-cardinality=0*)

Classify it and examine the result

- What does this tell you about the use of *ONLY* (Universal qualification)
- Why do we say that most ontologies are based on SOME (Existential qualification)
- Repair the definition of "All_girl_family" so that it behaves as expected
- Make this "Result 1" in your report

5

Paraphrases

- Note: in what follows 'THAT' transforms to logical 'AND', i.e.:
  - Dog THAT *has_owner* SOME Person

    is the same as
  - DOG AND *has_owner* SOME Person

6

Part 2: Functional Properties with SOME & ONLY

- Create the classes Dog, Cat, and Person and make them disjoint
- Create the property 'has_owner' with the inverse 'owns'
  - Make has_owner functional -(tick functional box on properties tab)
- Define the four classes
  - Dog THAT *has_owner* SOME Person
  - Dog THAT *has_owner* ONLY Person
  - Person THAT *owns* SOME Dog
  - Person THAT *owns* ONLY Dog
- Run the classifier and explain the result.
  - Why is the pattern the subsumptions of Person and Dog different?
    - What does this tell you about the use of SOME and ONLY with functional properties.
      - Why is this true?
- Submit this as "Result 2" in your report

7

Part 3: Conjunction, disjunction & negation

- (You should already have created the classes Dog, Cat, and Person and made them disjoint)
- Define - using necessary & sufficient conditions - the four classes
  - Person THAT *owns* SOME (Dog AND Cat)
    Person THAT *owns* ONLY (Dog AND Cat)
    Person THAT *owns* SOME (Dog OR Cat)
    Person THAT *owns* ONLY (Dog OR Cat)
- Before you run the classifier, predict which will be satisfiable and how the inferred subsumption hierarchy will be arranged. Run the classifier and check your understanding. Write up the explanation for the result
- Set a restriction on Person that every person *owns* at least one thing (*owns* min 1)
  - Predict how will this affect the classification.
  - Run the classifier. Explain the result. - it may be easier to see in OWLViz)

8

Negation

- Create examples with the negation of the previous restrictions - be sure to use necessary and sufficient conditions and leave the restriction that a person must own at least one thing.
  - Person THAT NOT *owns* SOME (Dog AND Cat)
    Person THAT NOT *owns* ONLY (Dog AND Cat)
    Person THAT NOT *owns* SOME (Dog OR Cat)
    Person THAT NOT *owns* ONLY (Dog OR Cat)
  - Before you run the classifier, predict which will be satisfiable and how the inferred subsumption hierarchy will be arranged. Run the classifier and check your understanding. Write up the explanation for the result. (some of these are hard)
  - Set the range of *owns* to (Dog OR Cat). How will this change the above results? Run the classifier, and explain.
  - Remove then restriction that a Person owns at least one thing. Explain the result
  - (It may be easier to see complex hierarchies in OWLViz). Also you might want to have a Person_1 and Person_2, owns_1, and owns_2 for different demonstrations.

9

Negation (cont)

- Add
- Person THAT (owns SOME Dog) AND (owns SOME Cat)
  Person THAT (owns SOME Dog)
  Person THAT (owns SOME Cat)
  Person THAT (owns min 2)
- Predict how these will classify with respect to the previous definitions. Classify and explain.
- Explain the relationship between:
  - owns SOME (Dog OR Cat)
    (owns SOME DOG) AND (owns SOME Cat)
    owns SOME (Dog AND Cat)
- Label the discussion for this section Results 3 in your report.

10

Part 4: Domain and Range Constraints

- If you have not already done so, create the class Animal and make it a superclass of both Cat and Dog.
- Create a sibling class Manufactured_thing so the hierarchy looks like
  - …
    Animal
    Manufactured_thing
  - Do NOT make Manufactured_thing and Animal disjoint yet.
    Make Car a primitive subclas of Manufactured_thing
- Make the domain of the property owns to be Person and the range to the Animal
- Define Personal_car as a car owned by a person and a Car_owner as a person that owns a car.
- Predict what will happen. Use the classifier to check
- Now make Animal and Manufactured_thing disjoint
- Predict what will happen and verify with the classifier
- Explain in write up under Results 4.

11

Part 5: General Inclusion Axioms

- Create defined class “Animals that have claws and meow” and create a necessary statement that it implies Cat.
- Add a restriction to Dog that all dogs have claws
- Create a subclass of dog that meows
- Predict the result; classify it as check.
  How would you debug this case? How might you make the errors easier to find?
- Submit as Results 5.

12

Part 6: Parts and Wholes

- Add to mammals that they all have four limbs
- Create a common parent for cat and dog - call it Cat_or_Dog
- Add that the limbs of Cat_or_Dog are of type leg.
- Add that all ‘digits’ (fingers and toes) are parts of legs of Cat_or_Dog
- Add that claws are parts of the digits of Cat_or_Dog.
- Create the ‘Adapted SEP Triples’ to show the parts hierarchy for Cat_of_Dog. Classify it and check.
- Explain why this is done with is_part_of rather than has_part. (You might want to demonstrate by experiment)
- Label as Results 6.

13