SPARQL Quality Control

John Beverley

Assistant Professor, *University at Buffalo*
Co-Director, National Center for Ontological Research
Affiliate Faculty, *Institute of Artificial Intelligence and Data Science*

Outline

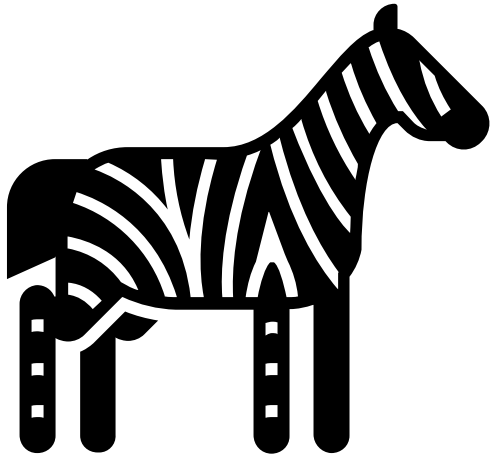
- Zebra Puzzle
- SPARQL Heuristics
- CCO Quality Control SPARQL Checks

Outline

- Zebra Puzzle
- SPARQL Heuristics
- CCO Quality Control SPARQL Checks

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.



WHO OWNS THE ZEBRA?

rdfs:comment [language: en]

Note 1: Each house is painted exactly one color; each house is painted a different color.
Note 2: Each house has exactly one human occupant of distinct nationality and exactly one distinct pet is owned by that human.
Note 3: Each human occupant drinks exactly one distinct beverage and smokes exactly one distinct brand of cigarettes.

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- Beverage
- Cigarette
- Color
- House
- Man
- Pet

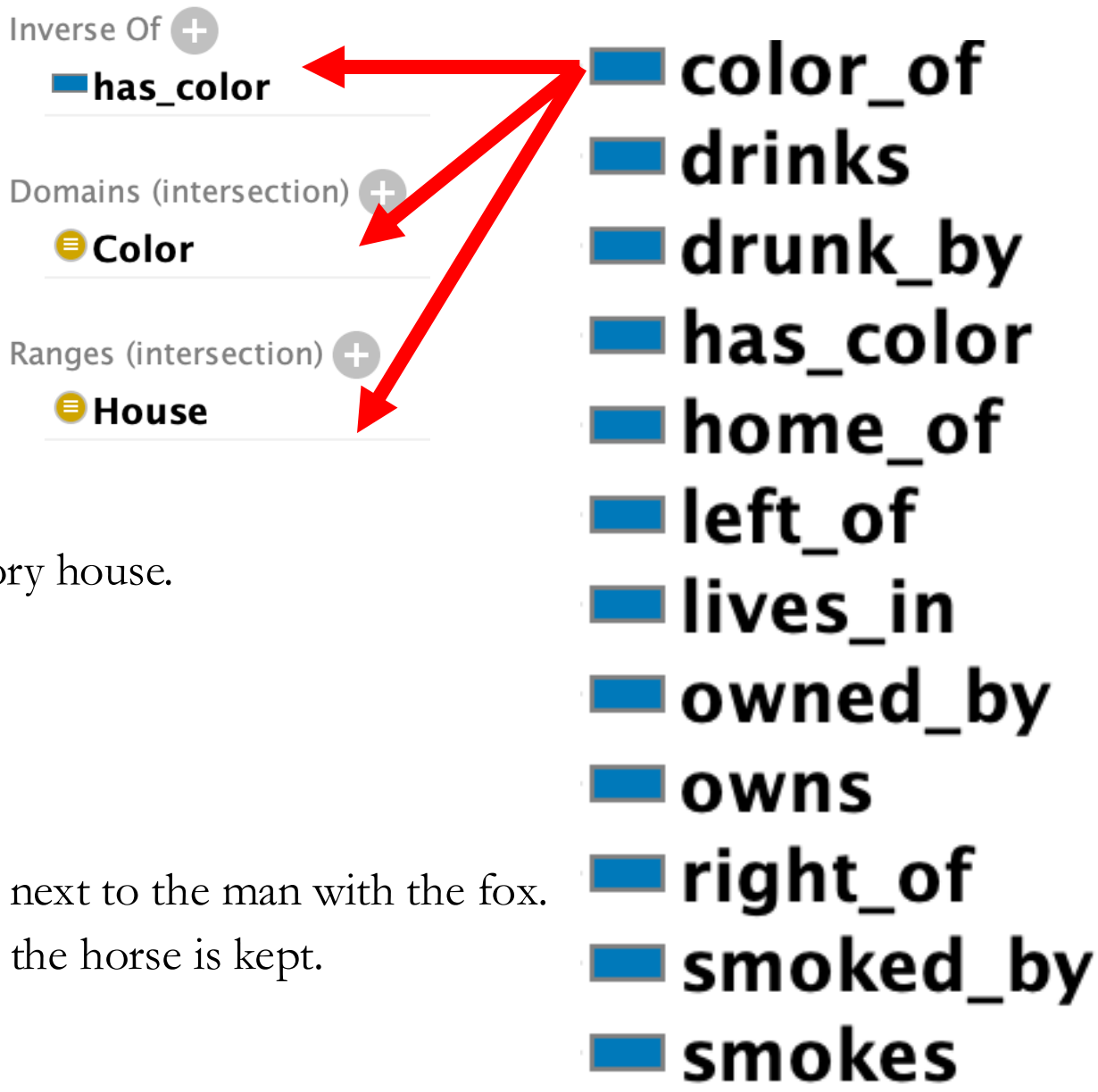
Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- color_of
- drinks
- drunk_by
- has_color
- home_of
- left_of
- lives_in
- owned_by
- owns
- right_of
- smoked_by
- smokes

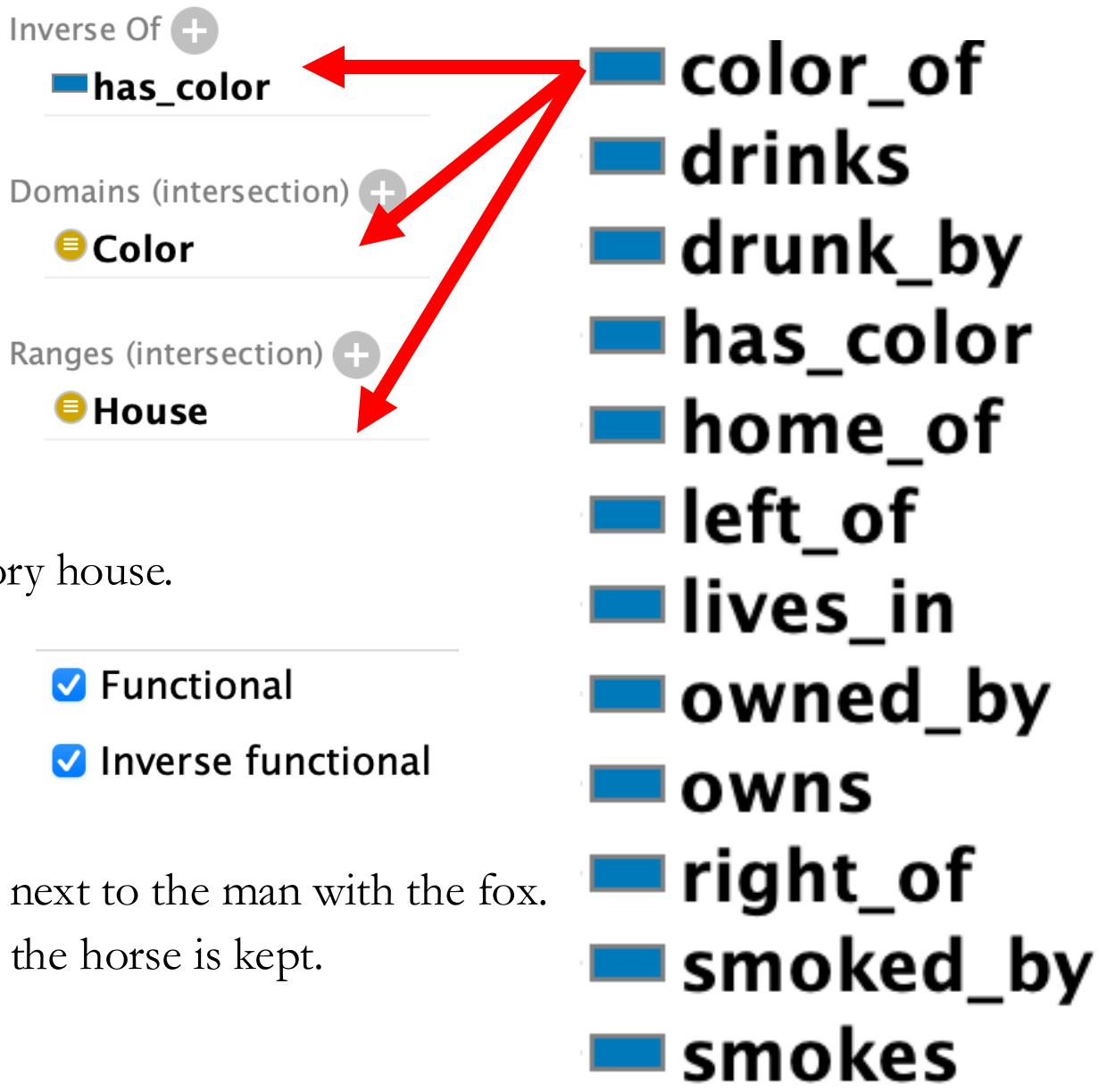
Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.



Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.



Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- color_of
- drinks
- drunk_by
- has_color
- home_of
- left_of
- lives_in
- owned_by
- owns
- right_of
- smoked_by
- smokes

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- color_of
- drinks
- drunk_by
- has_color
- home_of
- left_of
- lives_in
- owned_by
- owns
- right_of
- smoked_by
- smokes

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- color_of
- drinks
- drunk_by
- has_color
- home_of
- left_of
- lives_in
- owned_by
- owns
- right_of
- smoked_by
- smokes

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- color_of
- drinks
- drunk_by
- has_color
- home_of
- left_of
- lives_in
- owned_by
- owns
- right_of
- smoked_by
- smokes

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

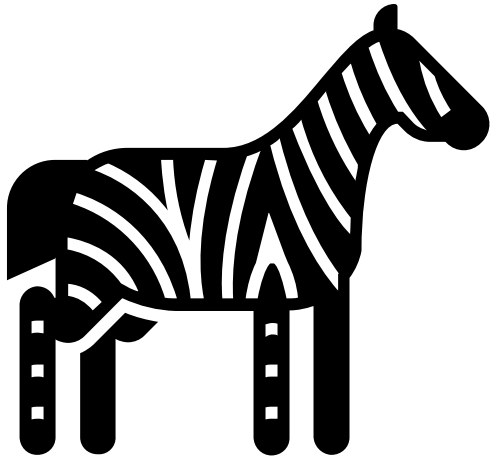
Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.



- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

Color

color_of some (left_of some (has_color value ivory))

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

green is the color of some x

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

...and since the domain of color_of is colors and the range is houses, it follows that x is a house

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

green is the color of some house

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

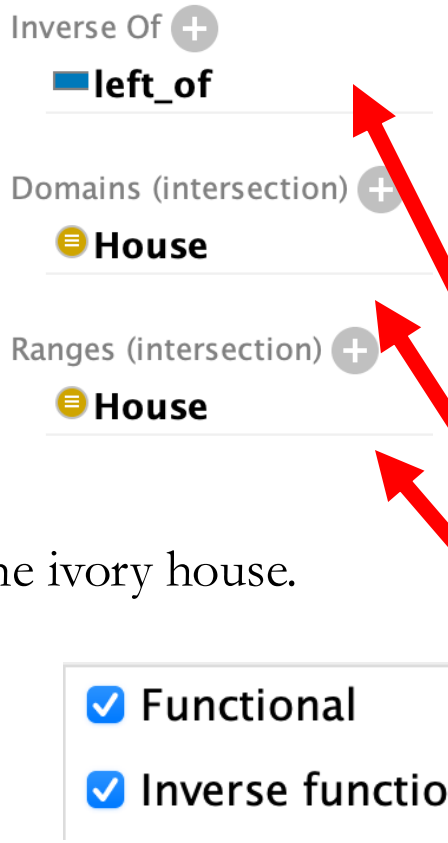
Where that house is to the right of some x

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.



Zebra Puzzle

color_of some (right_of some (has_color value ivory))

Where that house is to the right of some house

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

...that is ivory

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Zebra Puzzle

color_of some (right_of some (has_color value ivory))

Altogether, green is the color of some house x that is to the right of some house y that has color ivory

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in a house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese man smokes Parliaments.
15. The Norwegian lives next to the blue house.

- blue
- chesterfields
- coffee
- dog
- englishman
- fox
- green
- horse
- house_1
- house_2
- house_3
- house_4
- house_5
- ivory
- japanese
- kools
- lucky_strikes
- milk
- norwegian
- old_gold
- orange_juice
- parliaments
- red
- snail
- spaniard
- tea
- ukrainian
- water
- yellow
- zebra

Outline

- Zebra Puzzle
- SPARQL Heuristics
- CCO Quality Control SPARQL Checks

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

How many classes are in the artifact?

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

Return the variable ?classCount...

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

...as a string...

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

...that reflects the count of distinct resources...

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

...that are either of type rdf:Class or owl:Class...

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

...but aren't blank nodes...

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

Why string?

```
# Return a count of the number of classes, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
VALUES ?type {owl:Class rdf:Class}
  ?class a ?type.
FILTER(!isBlank(?class))
}
```

Because COUNT returns an integer and your results will accordingly have an xsd:integer tag applied to them, e.g. “24”xsd:integer

```
# Return a count of the number of object properties, excluding blank nodes

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (STR(COUNT(DISTINCT ?property)) AS ?propertyCount)
WHERE {
  ?property a owl:ObjectProperty.
  FILTER(!isBlank(?property))
}
```

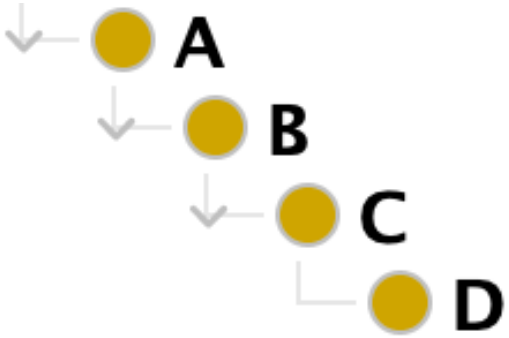
Similar query returns the number of object properties in the file

```
SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
  ?class a owl:Class.
  FILTER NOT EXISTS { ?instance a ?class. }
  FILTER NOT EXISTS { ?subclass rdfs:subClassOf ?class.
    ?instance a ?subclass.}
}
```

Return the number of classes that do not have instances.

```
SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
  ?class a owl:Class.
  FILTER NOT EXISTS { ?instance a ?class. }
  FILTER NOT EXISTS { ?subclass rdfs:subClassOf ?class.
    ?instance a ?subclass.}
}
```

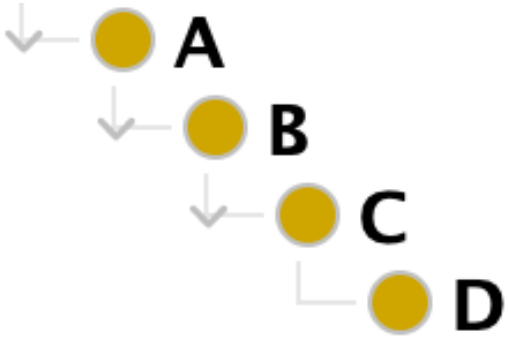
Be sure to check subclasses as well



If x is an instance of D then it's an instance of A

```
SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
  ?class a owl:Class.
  FILTER NOT EXISTS { ?instance a ?class. }
  FILTER NOT EXISTS { ?subclass rdfs:subClassOf ?class.
    ?instance a ?subclass.}
}
```

Be sure to check subclasses as well



SPARQL is targeted; it only checks what you tell it to check, i.e. unless you tell it to check whether x is an instance of A, it won't check

Useful Operators

- Suppose you want to match all proper inferred subclasses of a class

  rdfs:subClassOf+ matches one to n paths (excluding itself)

- Suppose you want to find all the subclasses of A, including itself

  rdfs:subClassOf* matches zero to n paths (including itself)

```
SELECT (STR(COUNT(DISTINCT ?class)) AS ?classCount)
WHERE {
  ?class a owl:Class.
  FILTER NOT EXISTS { ?instance a ?class. }
  FILTER NOT EXISTS { ?subclass rdfs:subClassOf+ ?class.
    ?instance a ?subclass.}
}
```

Note the "+" in this version of the query

SPARQL provides two operators that can be appended to rdfs:subClassOf and rdfs:subPropertyOf that check property chains

"+" connects the subject and object of the path by one or more matches
"*" connects the subject and object of the path by zero or more matches

```
# Returns object properties that are linked to instances

SELECT DISTINCT ?subject ?property ?object
WHERE {
  ?property a owl:ObjectProperty.
  ?subject ?property ?object .
  FILTER(!isBlank(?subject) && !isBlank(?object))
}
```

Returns oject properties linked to some instance and returns them in a table such that x R y

```
# Return list of instances associated with the nth class with highest number of instances

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?instance ?label
WHERE {
  {
    SELECT ?type
    WHERE {
      ?instance rdf:type ?type.
    }
    GROUP BY ?type
    ORDER BY DESC(COUNT(?instance))
    OFFSET 2                        # Currently second highest, replace as needed
    LIMIT 1
  }
  ?instance rdf:type ?type.
  OPTIONAL { ?instance rdfs:label ?label. }
}
```

Say you return a count of resources ordered by the number of instances each has, smallest to largest

```
# Return list of instances associated with the nth class with highest number of instances

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?instance ?label
WHERE {
  {
    SELECT ?type
    WHERE {
      ?instance rdf:type ?type.
    }
    GROUP BY ?type
    ORDER BY DESC(COUNT(?instance))
    OFFSET 2                        # Currently second highest, replace as needed
    LIMIT 1
  }
  ?instance rdf:type ?type.
  OPTIONAL { ?instance rdfs:label ?label. }
}
```

Suppose it's too long to really be manageable, when in truth you only want the list of instances associated with the class that has the 2nd highest number of instances

```
# Return list of instances associated with the nth class with highest number of instances

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?instance ?label
WHERE {
  {
    SELECT ?type
    WHERE {
      ?instance rdf:type ?type.
    }
    GROUP BY ?type
    ORDER BY DESC(COUNT(?instance))
    OFFSET 2                    # Currently second highest, replace as needed
    LIMIT 1
  }
  ?instance rdf:type ?type.
  OPTIONAL { ?instance rdfs:label ?label. }
}
```

OFFSET by 2 and LIMIT 1

Change “2” to any number you like

```
# Return count of instances of resource, ordered largest to smallest

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?resource (STR(COUNT(?instance)) AS ?count)
WHERE {
  ?instance a ?resource.
  FILTER(!isBlank(?resource))
}
GROUP BY ?resource
ORDER BY DESC(COUNT(?instance))
```

```
# Return count of instances of resource, ordered largest to smallest

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?resource (STR(COUNT(?instance)) AS ?count)
WHERE {
  ?instance a ?resource.
  FILTER(!isBlank(?resource))
}
GROUP BY ?resource
ORDER BY DESC(COUNT(?instance))
```

resource	count
owl:AnnotationProperty	"206"
owl:DatatypeProperty	"21"
owl:Class	"11"
rdfs:Datatype	"9"
owl:ObjectProperty	"8"
owl:NamedIndividual	"6"
D	"1"

Useful Operators

- Suppose you want to check match the parent class of a given class, but only have in SPARQL owl:subClassOf

^owl:subClassOf is the inverse of owl:subClassOf

- Suppose you want to match any x that is rdfs:type of owl:subClassOf A

rdfs:type/owl:subClassOf is a property path

- Suppose you want to match all x such that it is either rdf:type or owl:Class or rdfs:Class

rdfs:Class | owl:Class | rdf:type explores every path

Outline

- Zebra Puzzle
- SPARQL Heuristics
- CCO Quality Control SPARQL Checks

Quality Control

- As you investigate the CCO repository, you'll note there are not many SPARQL queries being run against builds

You will be providing new quality control SPARQL queries to be added to the CCO repository

Build and test ontology release

```
1   ▶ Run make all
9   mkdir -p build/lib src/ .github/deployment/sparql build/artifacts src/ .github/deployment/
10  curl -L -o build/lib/robot.jar https://github.com/ontodev/robot/releases/download/v1.8.4/r
11    % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
12                                   Dload  Upload   Total   Spent    Left  Speed
13
14    0     0    0     0    0     0      0
15    0     0    0     0    0     0      0
16
17    7 78.3M    7 5932k    0     0  23.1M
18  100 78.3M  100 78.3M    0     0   158M
19  chmod +x build/lib/robot.jar
20  for file in src/cco-modules/AgentOntology.ttl src/cco-modules/EventOntology.ttl src/cco-modules/QualityOntology.ttl src/cco-modules/InformationEntityOntology.ttl
21          echo "Reasoning on $file...";
22          java -jar build/lib/robot.jar
23  done
24  Reasoning on src/cco-modules/AgentOnto
25  Reasoning on src/cco-modules/ArtifactO
26  Reasoning on src/cco-modules/CurrencyI
27  Reasoning on src/cco-modules/EventOnto
28  Reasoning on src/cco-modules/ExtendedI
29  Reasoning on src/cco-modules/FacilityO
30  Reasoning on src/cco-modules/Geospatia
31  Reasoning on src/cco-modules/QualityOn
32  Reasoning on src/cco-modules/UnitsOfMe
33  Reasoning on src/cco-modules/TimeOnto
```

```
44  element,definition,error
45  http://www.ontologyrepository.com/CommonCoreOntologies/has_affiliate,"x is_affiliated_with y iff x and y are
    any kind of social or business relationship.",WARNING: The following ontology elements have the same cco:defi
    http://www.ontologyrepository.com/CommonCoreOntologies/has_affiliate and http://www.ontologyrepository.com/Co
46  http://www.ontologyrepository.com/CommonCoreOntologies/is_affiliated_with,"x is_affiliated_with y iff x and y
    have any kind of social or business relationship.",WARNING: The following ontology elements have the same cco
    http://www.ontologyrepository.com/CommonCoreOntologies/is_affiliated_with and http://www.ontologyrepository.c
47  PASS Rule .github/deployment/sparql/duplicate_label.sparql: 0 violation(s)
48  PASS Rule .github/deployment/sparql/exactly_1_prefLabel_per_lang.sparql: 0 violation(s)
49  FAIL Rule .github/deployment/sparql/min_1_eng_def.sparql: 86 violation(s)
50  resource,label,error
51  http://www.ontologyrepository.com/CommonCoreOntologies/has_aunt,,WARNING: Missing definition for
    http://www.ontologyrepository.com/CommonCoreOntologies/has_aunt
52  http://www.ontologyrepository.com/CommonCoreOntologies/has_brother,,WARNING: Missing definition for
    http://www.ontologyrepository.com/CommonCoreOntologies/has_brother
53  http://www.ontologyrepository.com/CommonCoreOntologies/has_brother_in_law,,WARNING: Missing definition for
    http://www.ontologyrepository.com/CommonCoreOntologies/has_brother_in_law
54  http://www.ontologyrepository.com/CommonCoreOntologies/has_daughter,,WARNING: Missing definition for
    http://www.ontologyrepository.com/CommonCoreOntologies/has_daughter
55  http://www.ontologyrepository.com/CommonCoreOntologies/has_daughter_in_law,,WARNING: Missing definition for
    http://www.ontologyrepository.com/CommonCoreOntologies/has_daughter_in_law
56  http://www.ontologyrepository.com/CommonCoreOntologies/has_father,,WARNING: Missing definition for
    http://www.ontologyrepository.com/CommonCoreOntologies/has_father
57  http://www.ontologyrepository.com/CommonCoreOntologies/has_father_in_law,,WARNING: Missing definition for
```

The SPARQL Library of Common Core Ontologies

The goal of this project is to develop a suite of SPARQL queries that will serve as quality control (QC) checks against the Common Core Ontologies suite. These queries will be designed to identify and flag potential issues, ensuring the ontology's integrity, consistency, and adherence to predefined standards.

Assignment Details

> Your task is to construct SPARQL queries to be included in the CCO QC workflow. Ideally, your queries will be added to the CCO repository here.

Your queries will be ranked in terms of difficulty. The lowest - 8 - indicates a rather easy query, while the highest - 1 - will indicate a very sophisticated query.

For our purposes, the more sophisticated queries will be worth more points than less sophisticated, and you are required to submit enough queries to acquire 100 points according to the following point system:

Query Sophistication	Points
1	35
2	25
3	20
4	10
5	5
6	3
7	2
8	0

The SPARQL Library of Common Core Ontologies

The goal of this project is to develop a suite of SPARQL queries that will serve as quality control (QC) checks against the Common Core Ontologies suite. These queries will be designed to identify and flag potential issues, ensuring the ontology's integrity, consistency, and adherence to predefined standards.

Assignment Details

> Your task is to construct SPARQL queries to be included in the CCO QC workflow. Ideally, your queries will be added to the CCO repository here.

Your queries will be ranked in terms of difficulty. The lowest - 8 - indicates a rather easy query, while the highest - 1 - will indicate a very sophisticated query.

For our purposes, the more sophisticated queries will be worth more points than less sophisticated, and you are required to submit enough queries to acquire 100 points according to the following point system:

Query Sophistication	Points
1	35
2	25
3	20
4	10
5	5
6	3
7	2
8	0

The SPARQL Library of Common Core Ontologies

The goal of this project is to develop a suite of SPARQL queries that will serve as quality control (QC) checks against the Common Core Ontologies suite. These queries will be designed to identify and flag potential issues, ensuring the ontology's integrity, consistency, and adherence to predefined standards.

Assignment Details

> Your task is to construct SPARQL queries to be included in the CCO QC workflow. Ideally, your queries will be added to the CCO repository here.

Your queries will be ranked in terms of difficulty. The lowest - 8 - indicates a rather easy query, while the highest - 1 - will indicate a very sophisticated query.

For our purposes, the more sophisticated queries will be worth more points than less sophisticated, and you are required to submit enough queries to acquire 100 points according to the following point system:

Query Sophistication	Points
1	35
2	25
3	20
4	10
5	5
6	3
7	2
8	0

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

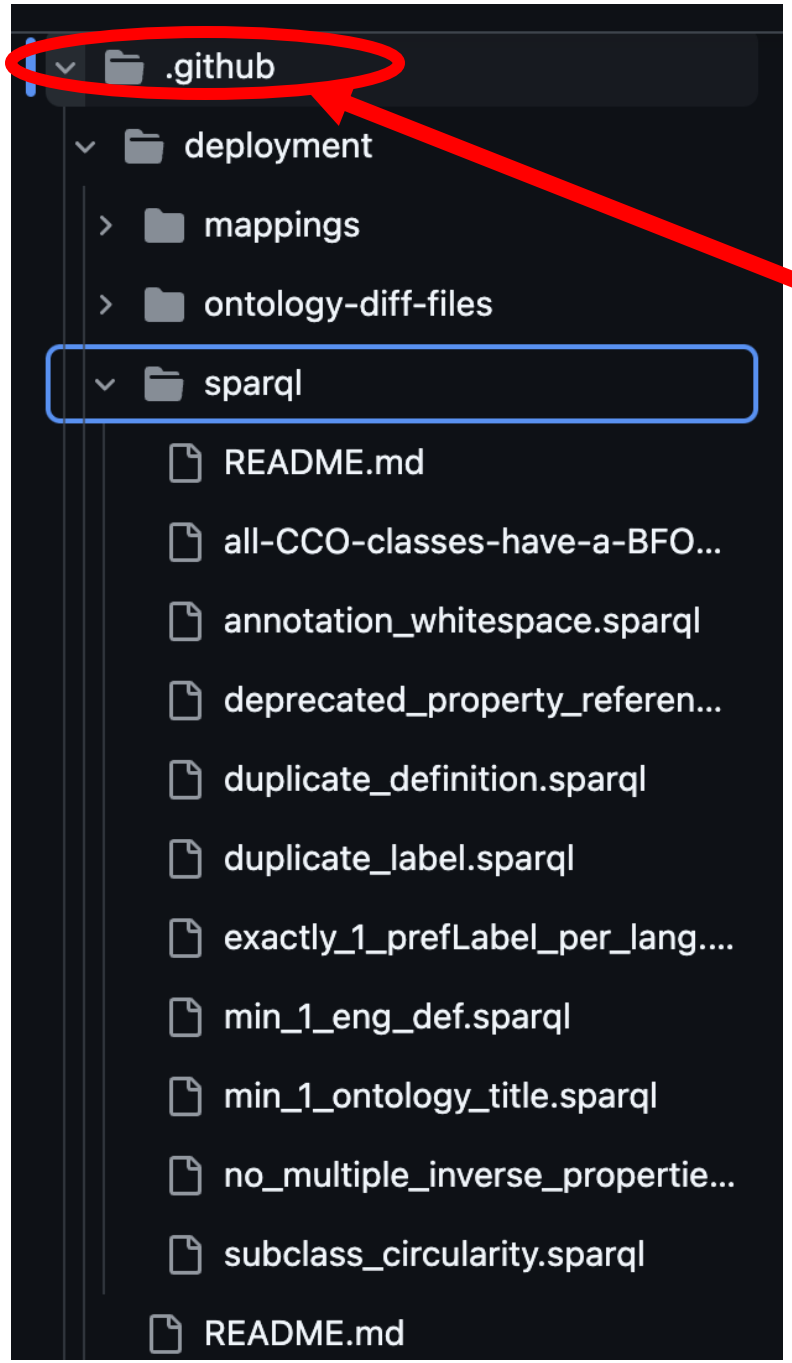
Guidance

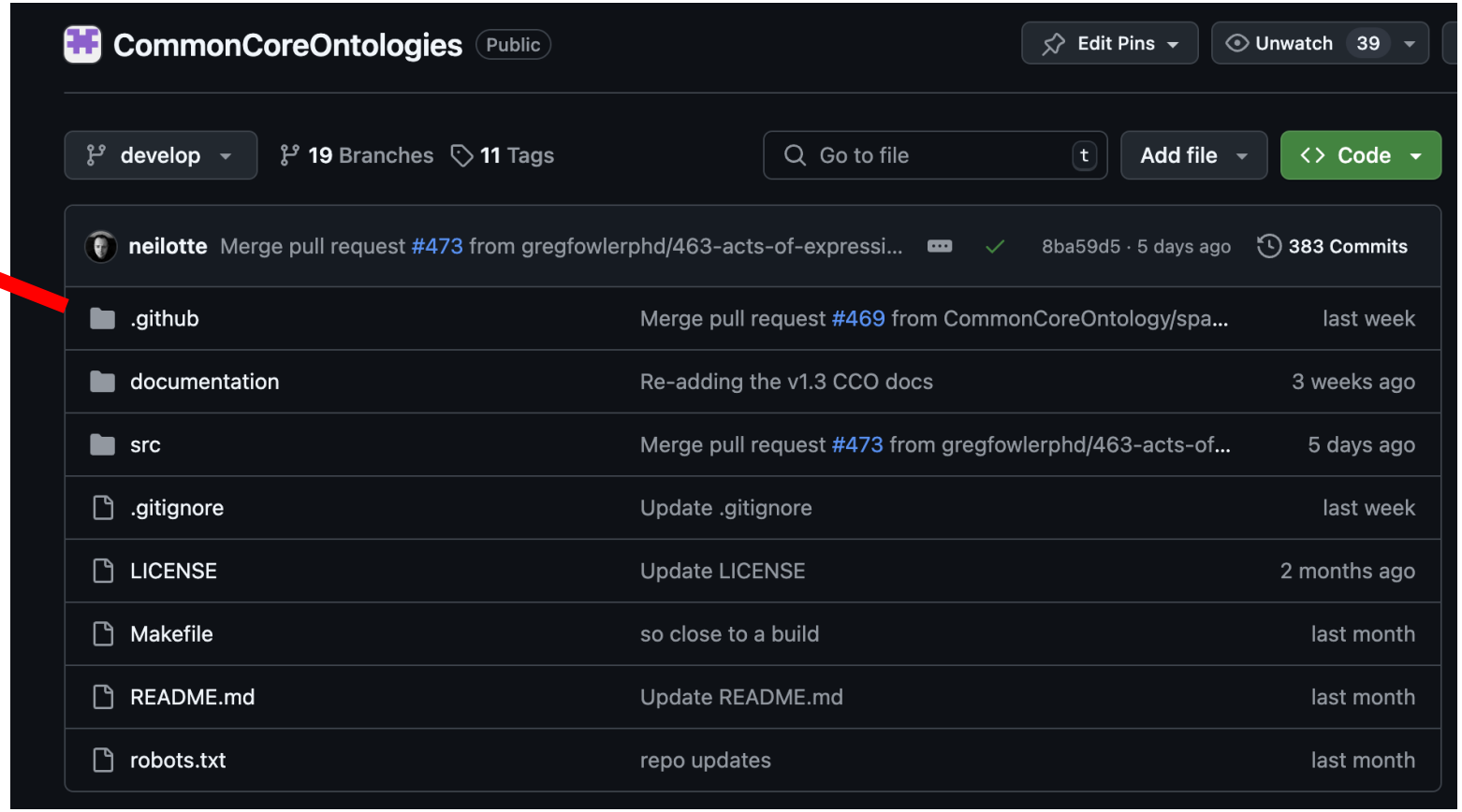
A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

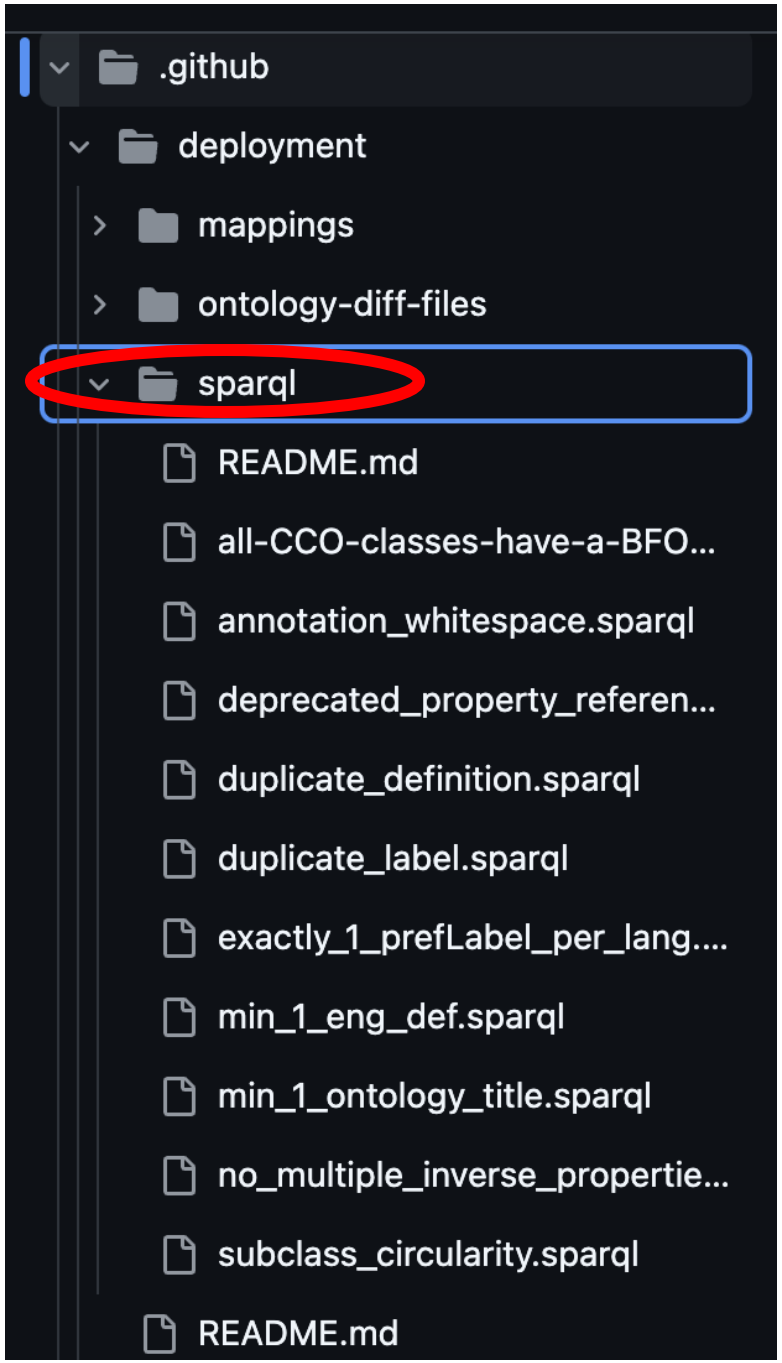
- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

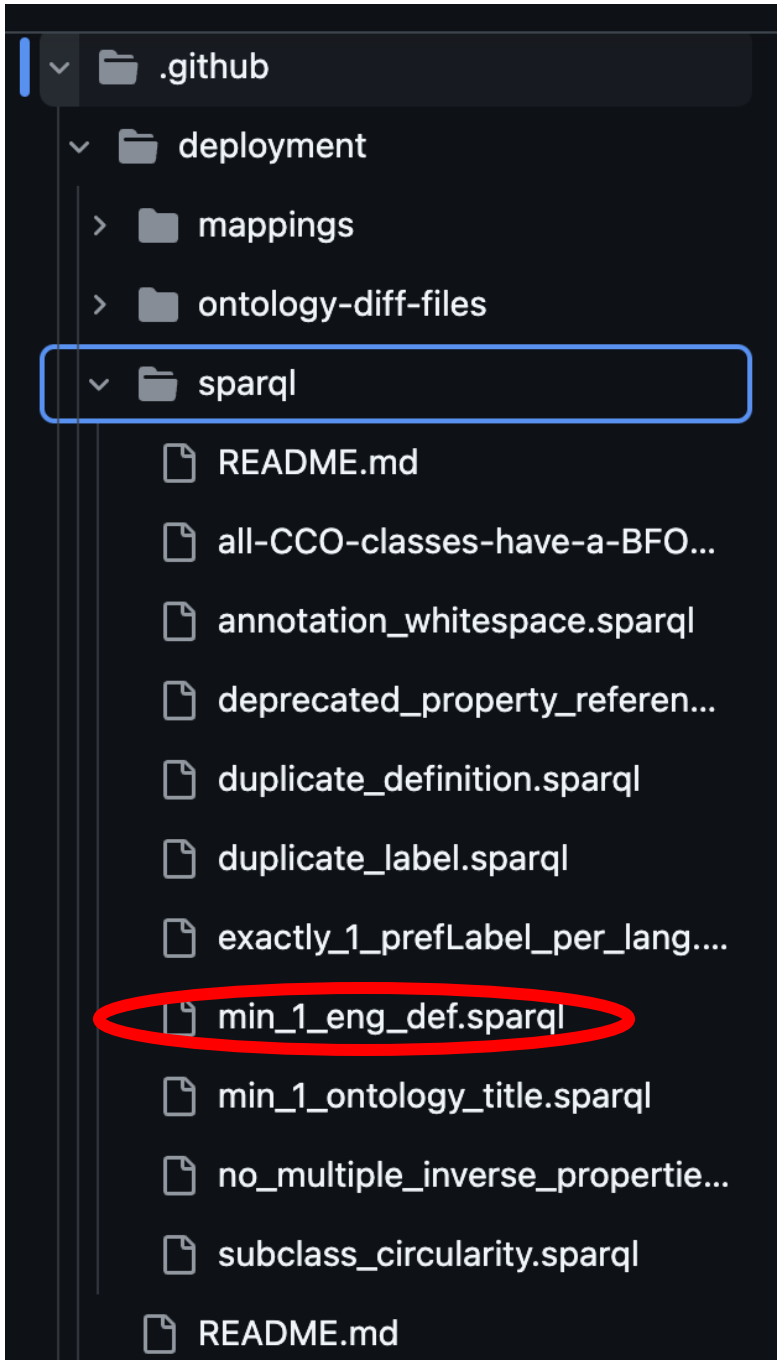
When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.


.github
deployment
mappings
ontology-diff-files
sparql
README.md
all-CCO-classes-have-a-BFO...
annotation_whitespace.sparql
deprecated_property_referen...
duplicate_definition.sparql
duplicate_label.sparql
exactly_1_prefLabel_per_lang....
min_1_eng_def.sparql
min_1_ontology_title.sparql
no_multiple_inverse_propertie...
subclass_circularity.sparql
README.md


CommonCoreOntologies
Public
Edit Pins
Unwatch 39
develop
19 Branches
11 Tags
Go to file
t
Add file
Code
neilotte Merge pull request #473 from gregfowlerphd/463-acts-of-expressi...
8ba59d5 · 5 days ago
383 Commits
.github
Merge pull request #469 from CommonCoreOntology/spa...
last week
documentation
Re-adding the v1.3 CCO docs
3 weeks ago
src
Merge pull request #473 from gregfowlerphd/463-acts-of...
5 days ago
.gitignore
Update .gitignore
last week
LICENSE
Update LICENSE
2 months ago
Makefile
so close to a build
last month
README.md
Update README.md
last month
robots.txt
repo updates
last month


.github
deployment
mappings
ontology-diff-files
sparql
README.md
all-CCO-classes-have-a-BFO...
annotation_whitespace.sparql
deprecated_property_referen...
duplicate_definition.sparql
duplicate_label.sparql
exactly_1_prefLabel_per_lang....
min_1_eng_def.sparql
min_1_ontology_title.sparql
no_multiple_inverse_propertie...
subclass_circularity.sparql
README.md




```
# Title:
#    Definition Required
# Constraint Description:
#    Any class or object property must have a non-empty definition with an English language tag.
# Severity:
#     Warning

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cco: <http://www.ontologyrepository.com/CommonCoreOntologies/>

SELECT DISTINCT ?resource ?label ?error
WHERE {
VALUES ?type {owl:Class owl:ObjectProperty}
    ?resource a ?type .
    OPTIONAL {
        ?resource cco:definition ?englishDefinition .
        FILTER (langMatches(lang(?englishDefinition), "en"))
        }
    FILTER(!bound(?englishDefinition))
    FILTER(!isBlank(?resource))
    BIND (concat("WARNING: Missing definition for ", str(?resource)) AS ?error)
}
ORDER BY ?resource
```

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies: explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.

Template

The SPARQL queries should have the template: Title (descriptive title of the query) Constraint Description: (description of the query functionality) Severity: (select "Warning" or "Error")

Your query should end with a BIND clause and an associated ?error in the SELECT. For example:

- BIND (concat("WARNING: The following ontology elements have the same rdfs:label ", str(?element), " and ", str(?element2)) AS ?error)

Guidance

A few tips for developing effective SPARQL queries for the Common Core Ontologies (CCO):

- Review the existing SPARQL queries so as not to duplicate work
- Review documentation and design patterns to understand stucture of CCO
- Understand common issues in ontologies; explore the OOPS! list here for inspiration
- Observe annotation conventions, e.g. use of labels, comments, etc. must be present and accurate

When creating queries, start with simple quality control checks and build complexity through practice. Feel free to leverage generative AI for this project. Also, feel free to collaborate with peers.

Be sure to test your queries. You may do this in Protege or in the SPARQL playground.