

Αναπαράσταση Γνώσης στον Παγκόσμιο Ιστό

ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ 2022-2023

Ερώτημα 1:

A. Αποφασίσαμε να κατασκευάσουμε μία οντολογία που θα αναπαριστά έναν ζωολογικό κήπο. Συγκεκριμένα, ορίζουμε τις οντότητες που αποτελούν έναν ζωολογικό κήπο (όπως τα ζώα που φιλοξενεί, τους εργαζόμενους, τους επισκέπτες και τους χώρους που περιέχει) και τις έννοιες που σχετίζονται με την οργάνωση και την λειτουργία του αλλά και με την κατηγοριοποίηση και τα χαρακτηριστικά των ζώων του.

B. Απαντήσεις στις ερωτήσεις που ζητούνται:

- Το αντικείμενο που θα καλύψει η οντολογία που επιλέξαμε να δημιουργήσουμε είναι ένας ζωολογικός κήπος.
- Η συγκεκριμένη οντολογία που αφορά ζωολογικό κήπο μπορεί να χρησιμοποιηθεί ώστε να γνωρίζει κάποιος ενδιαφερόμενος επισκέπτης πώς οργανώνεται και πως λειτουργεί ένας ζωολογικός κήπος, τα ζώα που μπορεί να φιλοξενεί αλλά και τις τιμές των εισιτηρίων μιας επίσκεψης.
- Η οντολογία που επιλέξαμε θα παρέχει απαντήσεις σε ερωτήσεις τύπου:
Τι είδους ζώα μπορεί να συναντήσει κάποιος σε έναν ζωολογικό κήπο;
Ποια άτομα εργάζονται σε έναν ζωολογικό κήπο;
Ποια είναι η τιμή του εισιτηρίου για τους επισκέπτες;
Ποιες είναι οι διαθέσιμες εγκαταστάσεις ενός ζωολογικού κήπου;
- Με τον μηχανισμό συμπερασμού ορίζονται και συνδέονται τα δεδομένα, οι ιδιότητες, οι κλάσεις και τα υπόλοιπα στοιχεία RDF/RDFS. Επομένως, μπορούμε να πούμε ότι μέσω του μηχανισμού συμπερασμού μπορούν να φανερωθούν κλάσεις ή σχέσεις που δεν είχαν γίνει γνώστες νωρίτερα. Με αυτόν τον τρόπο μπορεί να επεκταθεί η πληροφορία της οντολογίας.

Μια εφαρμογή που θα μπορούσε να υπάρξει η οποία θα αξιοποιεί την οντολογία μας είναι μια εφαρμογή μηχανοργάνωσης ενός ζωολογικού κήπου. Σε αυτή θα μπορούν να έχουν πρόσβαση τόσο οι εργαζόμενοι όσο και οι επισκέπτες του κήπου.

Συγκεκριμένα, οι εργαζόμενοι θα μπορούν μέσα από την εφαρμογή να ελέγχουν τις ανάγκες των ζώων και θα φροντίζουν να βρίσκονται εγκαίρως στα πόστα τους. Όσον αφορά τους επισκέπτες του κήπου, θα μπορούν με την σειρά τους να ενημερώνονται για τα ζώα που θα συναντήσουν και τις τιμές των εισιτηρίων.

Γ. Στην οντολογία μας έχουμε την κλάση Zoo που είναι η ανώτατη υπερκλάση όλων των υπολοίπων. Η κλάση Zoo αποτελείται από 4 κύριες υποκλάσεις: Animal, Worker, Visit, Zoo_Structure.

Συγκεκριμένα, η κλάση Animal περιέχει όλες τις κατηγορίες ζώων που φιλοξενεί ο ζωολογικός κήπος (Mammal, Bird, Reptile, Fish) και η κάθε μια από αυτές περιέχει και κάποια είδη ζώων (πχ Tiger). Επιπλέον, ο ζωολογικός κήπος φιλοξενεί και την κατηγορία Honorary Mammal στην οποία ανήκουν ζώα που έχουν χαρακτηριστικά θηλαστικών και πτηνών ταυτόχρονα (Kiwi).

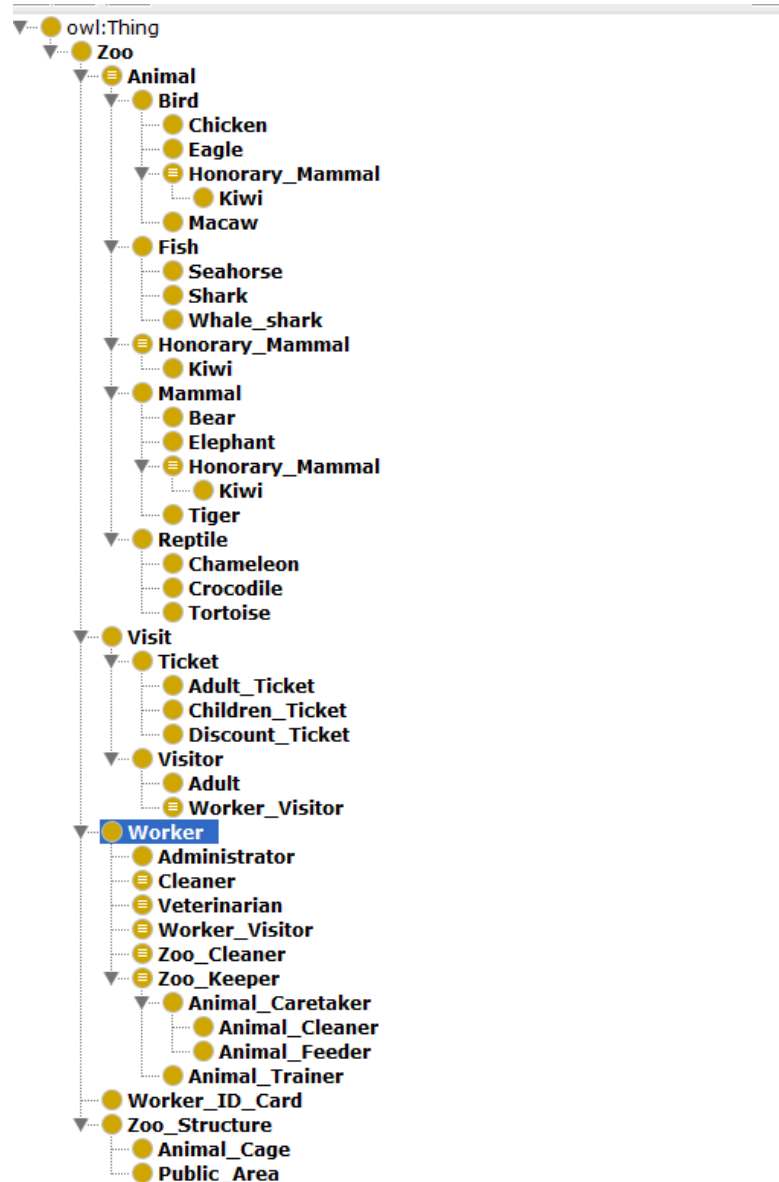
Στην συνέχεια, η κλάση Worker περιέχει όλους τους εργαζόμενους του ζωολογικού κήπου (πχ Veterinarian, Administrator, Zoo Keeper κλπ). Οι Zoo Keepers μπορεί να είναι Animal CareTakers και Animal Trainers ενώ στους Animal CareTakers ανήκουν οι καθαριστές και οι ταϊστές των ζώων.

Η κλάση Visit περιέχει τα εισιτήρια και τους επισκέπτες του ζωολογικού κήπου. Τα εισιτήρια χωρίζονται σε 3 κατηγορίες: Παιδικά, Ενηλίκων και εκπαιδευτικά εισιτήρια. Θεωρούμε ότι οι επισκέπτες είναι ενήλικες καθώς εκείνοι πληρώνουν για τα εισιτήρια.

Ένας εργαζόμενος που δουλεύει στον ζωολογικό κήπο και θέλει να τον επισκεφτεί θα πληρώνει εκπαιδευτικό εισιτήριο και θα ανήκει στην κλάση Worker_Visitor (τομή Worker και Visitor).

Η κλάση Zoo_Structure περιέχει τις εγκαταστάσεις του ζωολογικού κήπου που χωρίζονται στα κλουβιά των ζώων και στις κοινόχρηστες περιοχές.

Τέλος, η κλάση Worker_ID_Card είναι υποκλάση του Zoo και δηλώνει τις κάρτες ταυτοποίησης που έχουν όλοι οι εργαζόμενοι του ζωολογικού κήπου.



Δ. Στην οντολογία μας έχουμε Object και Data Properties.

Συγκεκριμένα οι **Object Properties** που δημιουργήσαμε είναι οι εξής:

BelongsTo: η ιδιότητα είναι Functional και δηλώνει ότι μια ID Card ανήκει σε έναν Worker.

CageHosts: η ιδιότητα είναι Inverse Functional και δηλώνει ότι ένα κλουβί φιλοξενεί ένα έως 3 ζώα (Cardinality).

Cares: η ιδιότητα αυτή δηλώνει ότι ένας Animal Caretaker φροντίζει ένα ζώο. Έχει δύο υπο-ιδιότητες: CleansCage και Feeds.

CleansCage: δηλώνει ότι ένας Animal Cleaner καθαρίζει ένα κλουβί ζώου.

Feeds: δηλώνει ότι ένας Animal Feeder ταΐζει ένα ζώο.

Cleans: η ιδιότητα αυτή δηλώνει ότι ένας Zoo Cleaner καθαρίζει έναν κοινόχρηστο χώρο.

HasIdCard: η ιδιότητα είναι Inverse Functional και δηλώνει ότι ένας Worker κατέχει μία Worker_ID_Card.

HasSameHeight: η ιδιότητα αυτή είναι συμμετρική και δηλώνει ότι δύο ζώα έχουν το ίδιο ύψος.

HasSameSalary: η ιδιότητα αυτή είναι συμμετρική και δηλώνει ότι δύο εργαζόμενοι έχουν τον ίδιο μισθό.

IsBiggerThan: η ιδιότητα είναι μεταβατική και δηλώνει ότι ένα ζώο είναι μεγαλύτερο σε μέγεθος από κάποιο άλλο.

IsCaredBy: δηλώνει ότι ένα ζώο δέχεται φροντίδα από κάποιον Animal CareTaker και έχει δυο υπο-ιδιότητες: CageCleanedBy, IsFedBy.

CageCleanedBy: δηλώνει ότι ένα κλουβί ζώου καθαρίζεται από έναν Animal Cleaner.

IsFedBy: δηλώνει ότι ένα ζώο ταϊζεται από έναν Animal Feeder.

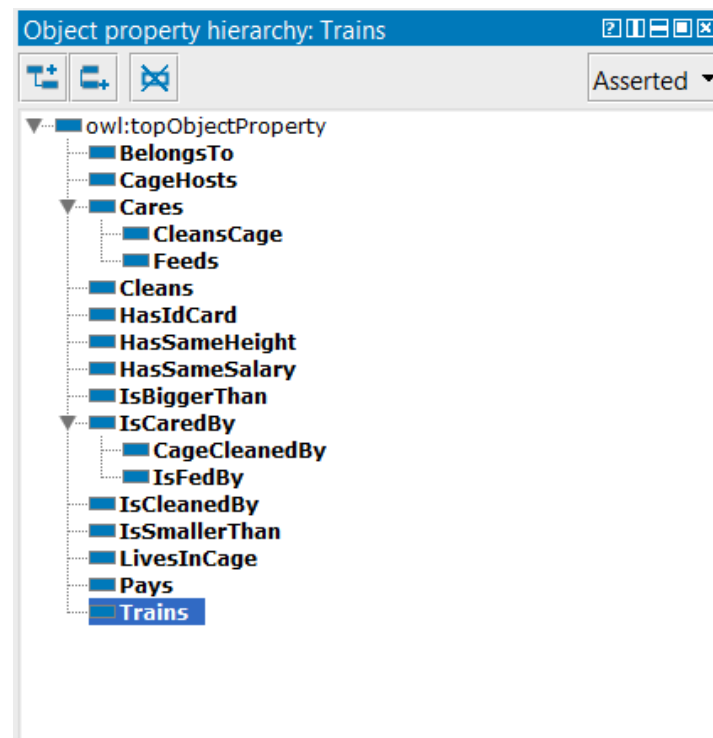
IsCleanedBy: δηλώνει ότι ένα Public Area καθαρίζεται από έναν Zoo Cleaner.

IsSmallerThan: η ιδιότητα είναι μεταβατική και δηλώνει ότι ένα ζώο είναι μικρότερο σε μέγεθος από κάποιο άλλο.

LivesInCage: η ιδιότητα είναι Functional και δηλώνει ότι ένα ζώο ζει σε ένα κλουβί.

Pays: Δηλώνει ότι ένας επισκέπτης πληρώνει εισιτήριο.

Trains: Δηλώνει ότι ένας Animal Trainer εκπαιδεύει ένα πτηνό.



Για τα **Data Properties** έχουμε τα εξής:

HasFeathers: δηλώνει την παρουσία πούπουλων σε ένα ζώο.

Age: Δηλώνει την ηλικία ενός ζώου.

Animal_ID: δηλώνει το μοναδικό ID του κάθε ζώου.

CageNum: Δηλώνει τον αριθμό που έχει το κάθε κλουβί.

Eats: Δηλώνει το είδος της τροφής που καταναλώνει το ζώο. Έχει 2 υπο-ιδιότητες: **EatsMeat** και **EatsPlants**.

EatsMeat: Δηλώνει ότι ένα ζώο τρώει κρέας.

EatsPlants: Δηλώνει ότι ένα ζώο τρώει φυτά.

HasBaby: Δηλώνει ότι ένα ζώο έχει μωρά και το πλήθος τους.

HasBroom: Δηλώνει την χρήση σκούπας για έναν **Zoo_Cleaner**.

HasDegreeIn: Δηλώνει το πτυχίο που κατέχει ένας κτηνίατρος.

HasName: Δηλώνει το μικρό όνομα ενός **Worker**.

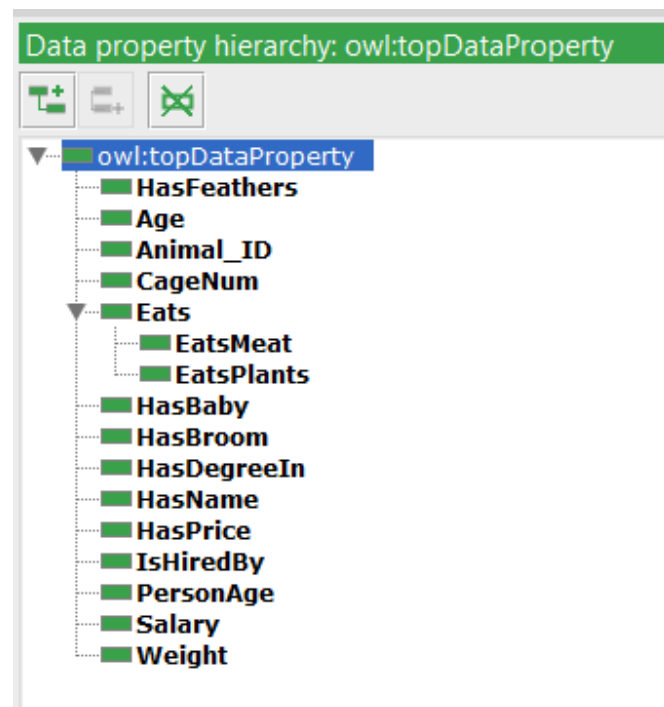
HasPrice: Δηλώνει την τιμή ενός εισιτηρίου.

IsHiredBy: Δηλώνει ότι ένας **Worker** έχει προσληφθεί από ένα συγκεκριμένο άτομο.

PersonAge: Δηλώνει την ηλικία ενός εργαζόμενου.

Salary: Δηλώνει τον μισθό ενός εργαζόμενου.

Weight: Δηλώνει το βάρος ενός ζώου.



Ε. Μερικά ενδεικτικά στιγμιότυπα για τις κλάσεις της οντολογίας:

Tiger1: αποτελεί στιγμιότυπο της κλάσης Tiger και αποτελεί μια τίγρη.

Michael_Scott: είναι Worker και αποτελεί στιγμιότυπο της κλάσης Administrator.

AdultTicket_001: στιγμιότυπο της κλάσης Adult_Ticket.

Shark_Tank: στιγμιότυπο της κλάσης Animal_Cage.

Gift_Shop: στιγμιότυπο της κλάσης Public_Area.

Ερώτημα 2:

Στο συγκεκριμένο ερώτημα αναπτύξαμε την οντολογία μας στο εργαλείο Protégé-OWL.

Οι κλάσεις και οι ιδιότητες που δημιουργήσαμε φαίνονται στο προηγούμενο ερώτημα μαζί με τις ιεραρχίες τους.

Ερώτημα 3:

1^η περίπτωση: *Alexandra_Zwidou Cares Tiger_Cage*

Η Αλεξάνδρα Ζωίδου φροντίζει το κλουβί της τίγρης.

Η τριπλέτα αυτή προκύπτει αφού η ιδιότητα *CleansCage* είναι υπό-ιδιότητα της *Cares*.

The screenshot shows the Protégé OWL editor interface. The top bar indicates the description is 'Alexandra_Zwidou' and the property assertions are 'Alexandra_Zwidou'. The 'Types' tab shows 'Animal_Cleaner' as the type for 'Alexandra_Zwidou'. The 'Object property assertions' tab shows 'CleansCage Tiger_Cage' and 'Cares Tiger_Cage'. The main area displays the explanation for the assertion 'Alexandra_Zwidou Cares Tiger_Cage'. It lists four explanations, each showing a sequence of logical steps and the number of other justifications for each step. The explanations are labeled 'Explanation 1' through 'Explanation 4'. At the bottom, there is an 'OK' button.

2^η περίπτωση: *Bear1 IsSmallerThan Whale_Shark1*

Η αρκούδα1 είναι μικρότερη από τον φαλινοκαρχαρία1. Η τριπλέτα αυτή προκύπτει λόγω της μεταβατικής ιδιότητας IsSmallerThan.

The screenshot shows the Protege GUI with the description 'Bear1' and property assertions including 'IsSmallerThan Crocodile1', 'IsSmallerThan Whale_Shark1', 'IsSmallerThan Elephant1', 'IsBiggerThan Eagle1', and 'IsBiggerThan Tiger4'. The explanation window is open, showing two explanations for the triple 'Bear1 IsSmallerThan Whale_Shark1'. Explanation 1 shows a chain of reasoning: 1) Elephant1 IsSmallerThan Whale_Shark1 (in ALL other justifications), 2) Transitive: IsSmallerThan (in NO other justifications), 3) Crocodile1 IsSmallerThan Elephant1 (in ALL other justifications), and 4) Bear1 IsSmallerThan Crocodile1 (in ALL other justifications). Explanation 2 shows a different chain: 1) Elephant1 IsSmallerThan Whale_Shark1 (in ALL other justifications), 2) Transitive: IsBiggerThan (in NO other justifications), 3) Crocodile1 IsSmallerThan Elephant1 (in ALL other justifications), 4) IsBiggerThan InverseOf IsSmallerThan (in NO other justifications), and 5) Bear1 IsSmallerThan Crocodile1 (in ALL other justifications).

3^η περίπτωση: *Crocodile1 HasSameHeight Tortoise1*

Ο κροκόδειλος1 έχει ίδιο ύψος με την χελώνα1. Η τριπλέτα αυτή προκύπτει λόγω της συμμετρικής ιδιότητας HasSameHeight.

The screenshot shows the Protege GUI with the description 'Crocodile1' and property assertions including 'IsSmallerThan Elephant1', 'IsSmallerThan Whale_Shark1', and 'HasSameHeight Tortoise1'. The explanation window is open, showing two explanations for the triple 'Crocodile1 HasSameHeight Tortoise1'. Explanation 1 shows a chain of reasoning: 1) Tortoise1 HasSameHeight Crocodile1 (in ALL other justifications) and 2) HasSameHeight InverseOf HasSameHeight (in NO other justifications). Explanation 2 shows a different chain: 1) Symmetric: HasSameHeight (in NO other justifications) and 2) Tortoise1 HasSameHeight Crocodile1 (in ALL other justifications).

4^η περίπτωση: *Shark2 LivesInCage Shark_Tank*

Ο καρχαρίας2 ζει στο ενυδρείο για τους καρχαρίες. Η τριπλέτα αυτή προκύπτει λόγω της ιδιότητας CageHosts που είναι η αντίστροφη της LivesInCage.

The screenshot shows the Protege GUI with the description 'Shark2' and property assertions including 'IsSmallerThan Elephant1', 'IsSmallerThan Whale_Shark1', 'IsBiggerThan Eagle1', 'IsBiggerThan Tiger4', and 'LivesInCage Shark_Tank'. The explanation window is open, showing two explanations for the triple 'Shark2 LivesInCage Shark_Tank'. Explanation 1 shows a chain of reasoning: 1) Shark_Tank CageHosts Shark2 (in ALL other justifications) and 2) CageHosts InverseOf LivesInCage (in NO other justifications).

5^η περίπτωση: *Tiger1 Same Individual As Tiger4.*

Η Τίγρης1 είναι ίδια με την Τίγρη4.

Η τριπλέτα αυτή προκύπτει επειδή η Τίγρης 1 ζει στο κλουβί Tiger_Cage και η Τίγρης4 ζει στο κλουβί Tiger_Cage και το κλουβί είναι γεμάτο λόγω Cardinality της CageHosts. Η CageHosts είναι Inverse Functional και έτσι πρέπει αναγκαστικά τα δύο individual να είναι το ίδιο.

Description: Tiger4

Types **+**

Tiger

Same Individual As **+**

Tiger1

Different Individuals **+**

Tiger2, Tiger3

Property assertions: Tiger_Cage

Object property assertions **+**

CageCleanedBy Alexandra_Zwidou

CageHosts Tiger4

CageHosts Tiger3

CageHosts Tiger2

CageHosts Tiger1

IsCaredBy Alexandra_Zwidou

Description: Animal_Cage

Equivalent To **+**

SubClass Of **+**

CageHosts max 3 Animal

CageHosts min 1 Animal

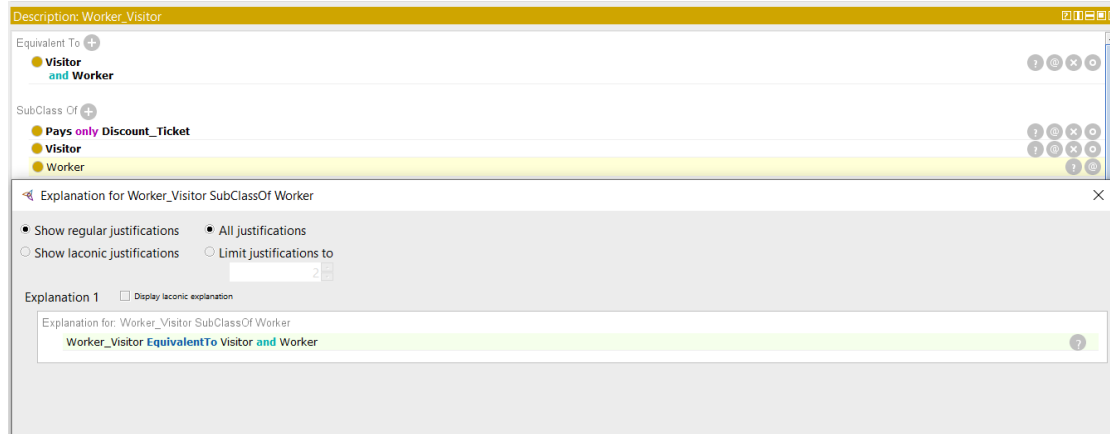
not (Public_Area)

Zoo_Structure

General class axioms **+**

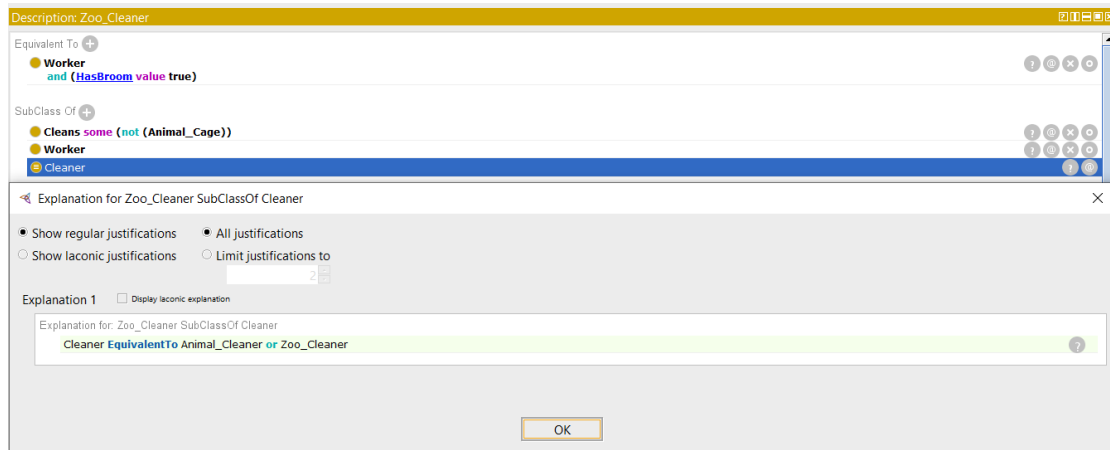
6^η περίπτωση: *Worker_Visitor SubClassOf Worker*.

Ο εργαζόμενος που επισκέπτεται τον ζωολογικό κήπο είναι υποκλάση των εργαζόμενων. Η τριπλέτα αυτή προκύπτει αφού η κλάση *Worker_Visitor* είναι ίση με την τομή των κλάσεων *Worker* και *Visitor*.



7^η περίπτωση: *Zoo_Cleaner SubClassOf Cleaner*.

Η κλάση *Zoo_Cleaner* (καθαριστές του ζωολογικού κήπου) είναι υποκλάση της κλάσης *Cleaner* (καθαριστές). Αυτό προκύπτει καθώς η κλάση *Cleaner* ισούται με την ένωση των καθαριστών του ζωολογικού κήπου (*Zoo_Cleaner*) και των καθαριστών των ζώων (*Animal_Cleaner*).



8^η περίπτωση: *Martin_Iglesias Same Individual As Martin_Iglesias_2*

Ο *Martin_Iglesias* είναι το ίδιο άτομο με τον *Martin_Iglesias_2*. Αυτό προκύπτει επειδή στα δύο αυτά άτομα ανήκει το ίδιο ID Card και η ιδιότητα *BelongsTo* είναι *Functional*.

Description: Martín_Iglesias

Types

+

Zoo_Cleaner

Same Individual As

+

Martin_Iglesias_2

Different Individuals

+

Description: Martín_Iglesias_ID

Types

+

Worker_ID_Card

Same Individual As

+

Different Individuals

+

Property assertions: Martín_Iglesias_ID

+

Object property assertions

+

BelongsTo Martín_Iglesias

BelongsTo Martín_Iglesias_2

Data property assertions

+

Negative object property assertions

+

Negative data property assertions

+

9^η περίπτωση: *Shark2 Same Individual As Shark3.*

Ο καρχαρίας2 είναι το ίδιο άτομο με τον καρχαρία3.

Αυτή η τριπλέτα προκύπτει αφού τα δύο individual έχουν τις ίδιες τιμές για τα Data Properties που διαθέτει το καθένα. Η ιδιότητα Animal_ID είναι Functional.

Types

+

Shark

Same Individual As

+

Shark3

Different Individuals

+

Shark2

Object property assertions

+

IsBiggerThan Eagle1

IsSmallerThan Whale_Shark1

LivesInCage Shark_Tank

IsBiggerThan Chameleon1

IsBiggerThan Seahorse1

IsBiggerThan Kiwi1

IsBiggerThan Chicken1

IsBiggerThan Tortoise1

IsBiggerThan Macaw1

Data property assertions

+

Age 27

Weight 19000

Animal_ID 112

Negative object property assertions

+

Negative data property assertions

+

Types

+

Shark

Same Individual As

+

Shark1

Different Individuals

+

Shark2

Object property assertions

+

IsBiggerThan Eagle1

IsSmallerThan Whale_Shark1

LivesInCage Shark_Tank

IsBiggerThan Chameleon1

IsBiggerThan Seahorse1

IsBiggerThan Kiwi1

IsBiggerThan Chicken1

IsBiggerThan Tortoise1

IsBiggerThan Macaw1

Data property assertions

+

Age 27

Animal_ID 112

Weight 19000

Negative object property assertions

+

Negative data property assertions

+

10^η περίπτωση: *Swtiris_Papadopoulos Type Zoo_Cleaner*.

Ο Σωτήρης Παπαδόπουλος ανήκει στην κλάση Zoo_Cleaner. Η τριπλέτα αυτή προκύπτει αφού το individual αυτό έχει την ιδιότητα HasBroom ίση με True και η κλάση Zoo_Cleaner ορίζεται ως οι εργαζόμενοι που έχουν σκούπα.

Description: Swtiris_papadopoulos

Property assertions: Swtiris_papadopoulos

Object property assertions: +

Data property assertions: +

HasBroom true

Types

- Worker
- Zoo_Cleaner

Explanation for Swtiris_papadopoulos Type Zoo_Cleaner

Show regular justifications | All justifications

Show laconic justifications | Limit justifications to

Explanation 1

Explanation for: Swtiris_papadopoulos Type Zoo_Cleaner

1) Swtiris_papadopoulos HasBroom true (in ALL other justifications)

2) HasBroom Domain Zoo_Cleaner (in NO other justifications)

Explanation 2

Explanation for: Swtiris_papadopoulos Type Zoo_Cleaner

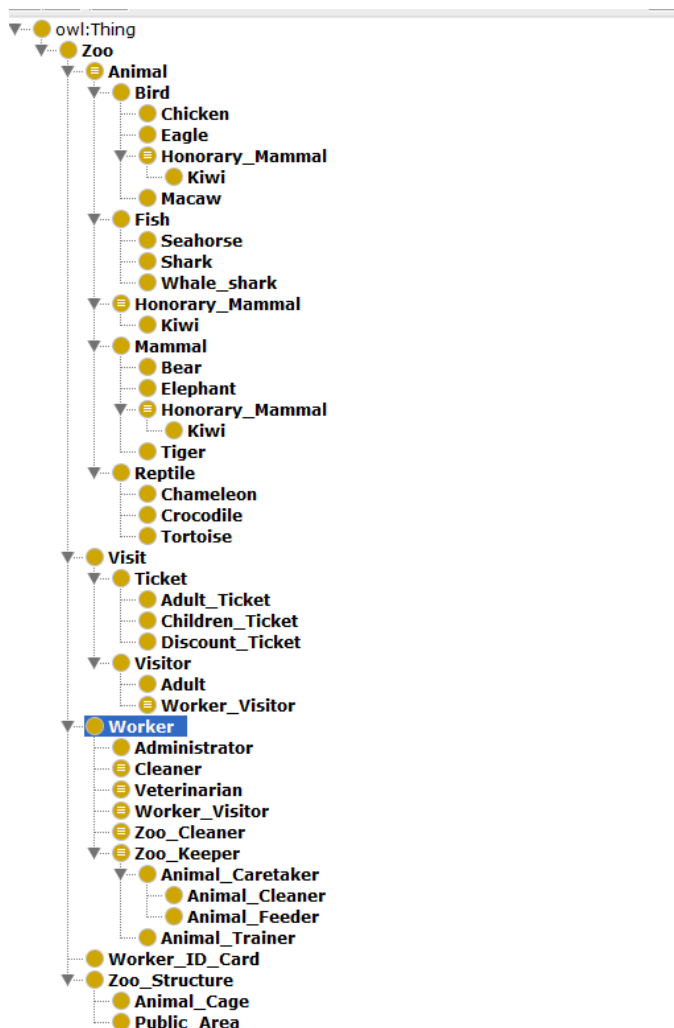
1) Swtiris_papadopoulos Type Worker (in NO other justifications)

2) Swtiris_papadopoulos HasBroom true (in ALL other justifications)

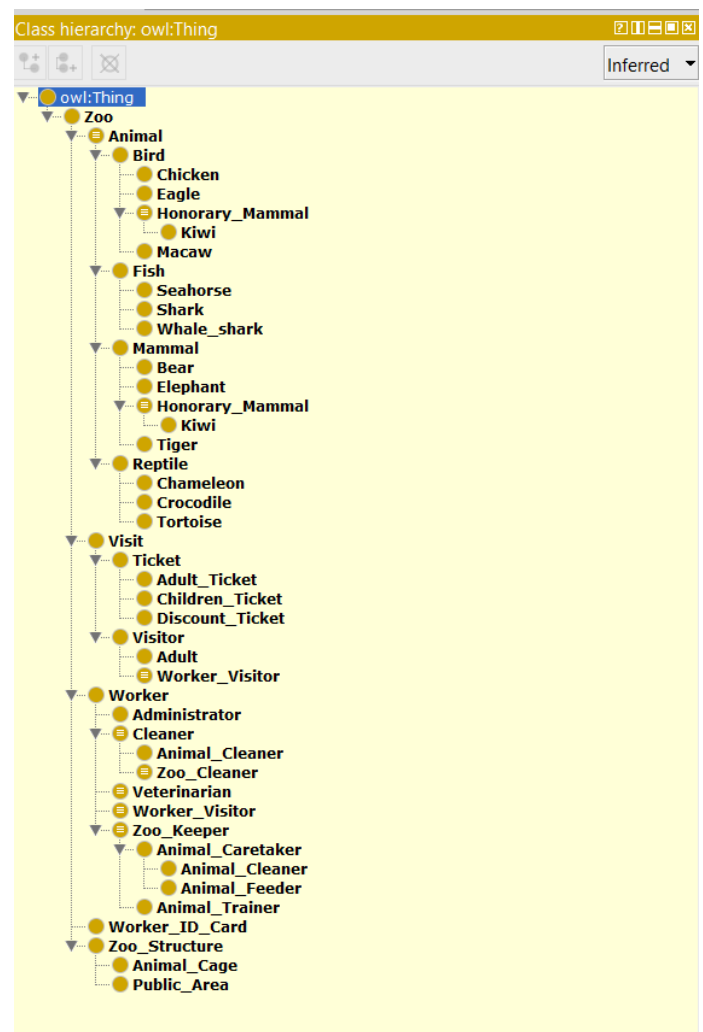
3) Zoo_Cleaner EquivalentTo Worker and (HasBroom value true) (in NO other justifications)

Ερώτημα 4:

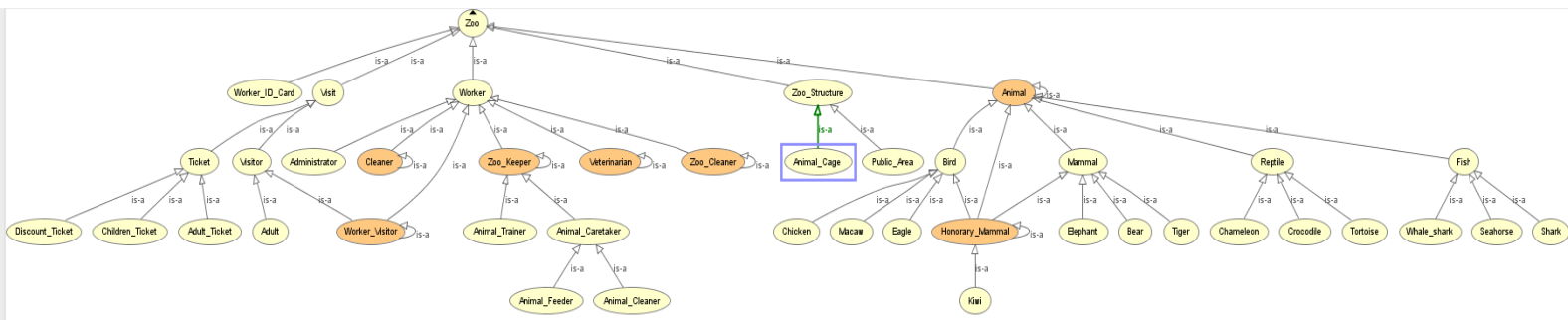
a. Asserted Class hierarchy:



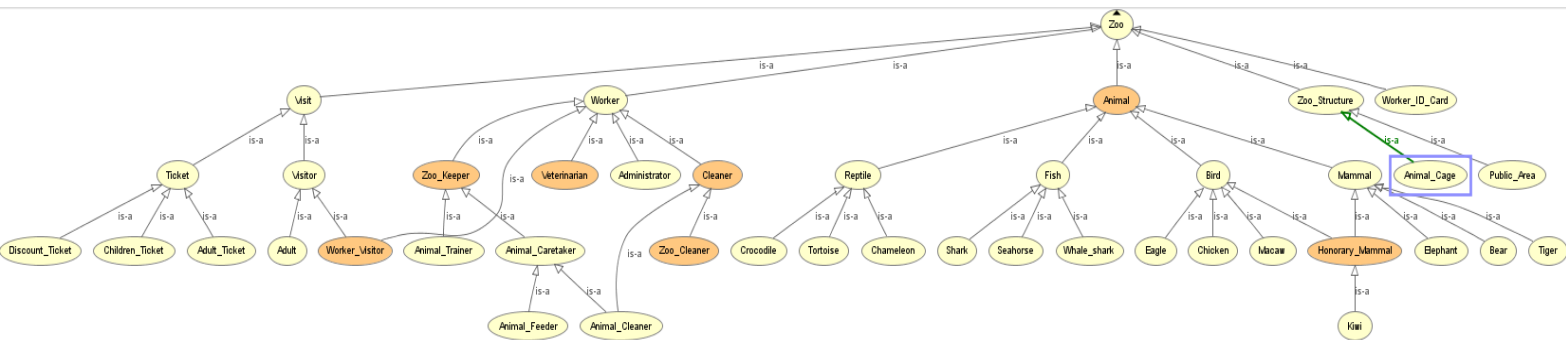
Inferred Class hierarchy:



b. Asserted Model:



Inferred Model:



c. Οι διαφορές που εντοπίζονται στις δύο ιεραρχίες και στους αντίστοιχους γράφους είναι οι εξής:

Όσον αφορά τις ιεραρχίες παρατηρούμε ότι η κλάση Cleaner στην asserted ιεραρχία δεν έχει υποκλάσεις ενώ στην inferred παρατηρούμε ότι η κλάση Cleaner αποτελείται από τις δύο υποκλάσεις Zoo_Cleaner και Animal_Cleaner (και οι δύο είναι Cleaners).

Η ίδια διαφορά παρατηρείται και στους δύο γράφους. Στο inferred model υπάρχουν βελάκια από τις κλάσεις Zoo_Cleaner και Animal_Cleaner που οδηγούν στην κλάση Cleaner.

Ερώτημα 5:

a. Παρακάτω έχουμε παραθέσει 5 queries με τους κώδικες SPARQL και τα αντίστοιχα αποτελέσματα που επιστρέφει το καθένα από αυτά.

1. Επιστρέφει τον μέγιστο μισθό από όλους τους μισθούς των ατόμων (εργαζόμενοι) που είναι ανήκουν στην κλάση Worker.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>
```

```
SELECT (MAX(?Salary) as ?maxSalary)
WHERE { ?Worker sch:Salary ?Salary.
}
```

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>

SELECT (MAX(?Salary) as ?maxSalary)
WHERE { ?Worker sch:Salary ?Salary.
}
```

Execute

	?maxSalary
69000.0	

2. Επιστρέφει όλους τους εργαζόμενους (Worker) που έχουν μισθό μεγαλύτερο των 1000 ευρώ μαζί με την ηλικία τους, ταξινομημένους σύμφωνα με την ηλικία.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>
```

```
SELECT ?Worker ?Age
WHERE{
  ?Worker sch:Salary ?Salary.
  ?Worker sch:PersonAge ?Age.
FILTER(?Salary >1000)
}
```

ORDER BY ?Age

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>

SELECT ?Worker ?Age
WHERE{
  ?Worker sch:Salary ?Salary.
  ?Worker sch:PersonAge ?Age.
FILTER(?Salary >1000)
}

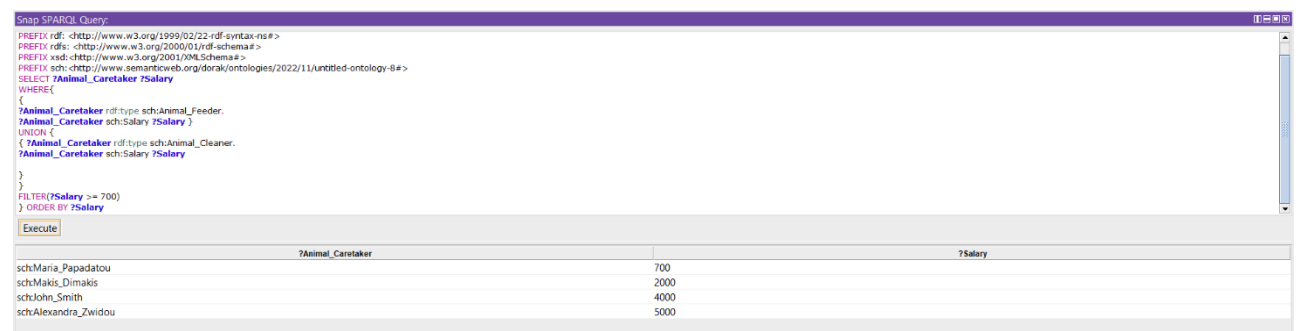
ORDER BY ?Age
```

Execute

	?Worker	?Age
	sch.Alexandra_Zwidou	19
	sch.John_Smith	25
	sch.Richardo_Pablo	32
	sch.Makis_Dimakis	36
	sch.Michael_Scott	44
	sch.Rita_Papa	58

3. Επιστρέφει όλους τους Animal Caretakers που αποτελούν UNION (ένωση) των Cleaners και των Feeders που έχουν μισθό μεγαλύτερο των 700 ευρώ, ταξινομημένους σύμφωνα με τον μισθό.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>
SELECT ?Animal_Caretaker ?Salary
WHERE{
{
?Animal_Caretaker rdf:type sch:Animal_Feeder.
?Animal_Caretaker sch:Salary ?Salary }
UNION {
{ ?Animal_Caretaker rdf:type sch:Animal_Cleaner.
?Animal_Caretaker sch:Salary ?Salary
}
}
}
FILTER(?Salary >= 700)
} ORDER BY ?Salary
```



The screenshot shows a SPARQL query execution interface. The query is the same as the one above. The results table has two columns: ?Animal_Caretaker and ?Salary. The results are as follows:

?Animal_Caretaker	?Salary
schMaria_Papadatou	700
schMakis_Dimakis	2000
schJohn_Smith	4000
schAlexandra_Zwidou	5000

4. Επιστρέφει όλα τα ζώα μαζί με το ID τους και το πλήθος των μωρών που μπορεί να έχουν, ταξινομημένα σύμφωνα με το ID τους.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>
```

```
SELECT ?Animal ?Animal_ID ?Baby
WHERE{ ?Animal sch:Animal_ID ?Animal_ID.
OPTIONAL{?Animal sch:HasBaby ?Baby}
}
ORDER BY ?Animal_ID
```

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>

SELECT ?Animal ?Animal_ID ?Baby
WHERE {
  ?Animal sch:Animal_ID ?Animal_ID.
  OPTIONAL { ?Animal sch:HasBaby ?Baby }
} ORDER BY ?Animal_ID
```

Execute

?Animal	?Animal_ID	?Baby
schChicken1	1	7
schEagle1	2	
schKiwi1	003	1
schMacaw1	004	
schSeahorse1	101	
schWhale_Shark1	103	
schShark3	112	
schShark1	112	
schShark2	122	
schBear1	201	5
schElephant1	202	
schTiger4	213	2
schTiger1	213	2
schTiger2	223	
schTiger3	233	
schChameleon1	301	
schCrocodile1	302	
schTortoise1	303	

5. Επιστρέφει το Animal ID και τα βάρη των ζώων που είναι μεγαλύτερα από έναν Αετό (για τα οποία έχει καταγραφεί βάρος) ταξινομημένα σύμφωνα με το βάρος τους. Με την χρήση του DISTINCT αποφεύγουμε τις διπλότυπες εγγραφές δηλαδή ζώα που μπορεί να έχουν ίδιο ID και ίδιο βάρος.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>

SELECT DISTINCT ?Animal_ID ?Weight
WHERE {
  ?Animal sch:IsBiggerThan ?Animal1.
  ?Animal1 rdf:type sch:Eagle.
  ?Animal sch:Animal_ID ?Animal_ID.
```

```
OPTIONAL { ?Animal sch:Weight ?Weight }
}
ORDER BY ?Weight
```

Snap SPARQL Query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sch: <http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8#>

SELECT DISTINCT ?Animal_ID ?Weight
WHERE {
  ?Animal sch:IsBiggerThan ?Animal1.
  ?Animal1 rdf:type sch:Eagle.
  ?Animal sch:Animal_ID ?Animal_ID.
  OPTIONAL { ?Animal sch:Weight ?Weight }
}
ORDER BY ?Weight
```

Execute

?Animal_ID	?Weight
202	4000
112	19000
103	
213	
302	
201	

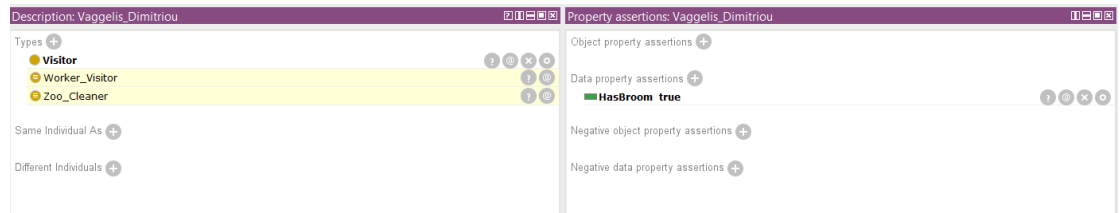
β. Οι κανόνες σε SWRL που δημιουργήσαμε φαίνονται παρακάτω:

untitled-ontology-8 (http://www.semanticweb.org/dorak/ontologies/2022/11/untitled-ontology-8)		
Active Ontology × Entities × Classes × Object Properties × Data Properties × Individuals by class × OWLViz × DL Query × SWRLTab × SQWRLTab × SPARQL Query ×		
	Name	Body
<input checked="" type="checkbox"/>	S1	Visitor(?untitled-ontology-8untitled-ontology-8x) ^ HasBroom(?untitled-ontology-8untitled-ontology-8x, true) -> Worker_Visitor(?untitled-ontology-8untitled-ontology-8x)
<input checked="" type="checkbox"/>	S2	Children_Ticket(?untitled-ontology-8x) -> HasPrice(?untitled-ontology-8x, 5)
<input checked="" type="checkbox"/>	S3	Mammal(?untitled-ontology-8untitled-ontology-8untitled-ontology-8x) ^ HasFeathers(?untitled-ontology-8untitled-ontology-8untitled-ontology-8x, true) -> Kiwi(?untitled-ontology-8untitled-ontology-8x)
<input checked="" type="checkbox"/>	S4	Visitor(?untitled-ontology-8untitled-ontology-8untitled-ontology-8x) ^ HasIdCard(?untitled-ontology-8untitled-ontology-8x, ?untitled-ontology-8untitled-ontology-8untitled-ontology-8x) -> Public_Area(?untitled-ontology-8untitled-ontology-8x)
<input checked="" type="checkbox"/>	S5	Worker(?untitled-ontology-8x) ^ Cleans(?untitled-ontology-8x, ?untitled-ontology-8y) -> Zoo_Cleaner(?untitled-ontology-8x)

1. Visitor(?x) ^ HasBroom(?x, true) -> Worker_Visitor(?x)

Αν ένας επισκέπτης έχει σκούπα (δηλαδή η ιδιότητα δεδομένων HasBroom έχει τιμή True) τότε ο επισκέπτης αυτός είναι και εργαζόμενος, δηλαδή ανήκει στην κλάση Worker_Visitor.


Το παράδειγμα για τον κανόνα αυτόν είναι το ακόλουθο:



2. Children_Ticket(?x) -> HasPrice(?x, "5"^^xsd:int)

Τα άτομα της κλάσης Children_Ticket δηλαδή τα παιδικά εισιτήρια θα μπορούν να πάρουν μόνο την τιμή 5.

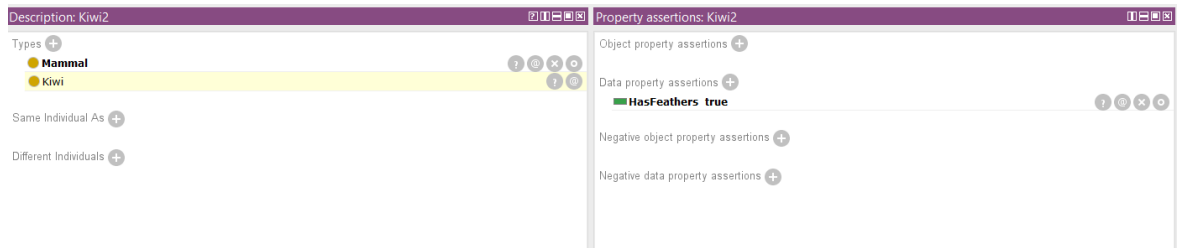
Το παράδειγμα για τον κανόνα αυτόν είναι το ακόλουθο:



3. Mammal(?x) ^ HasFeathers(?x, true) -> Kiwi(?x)

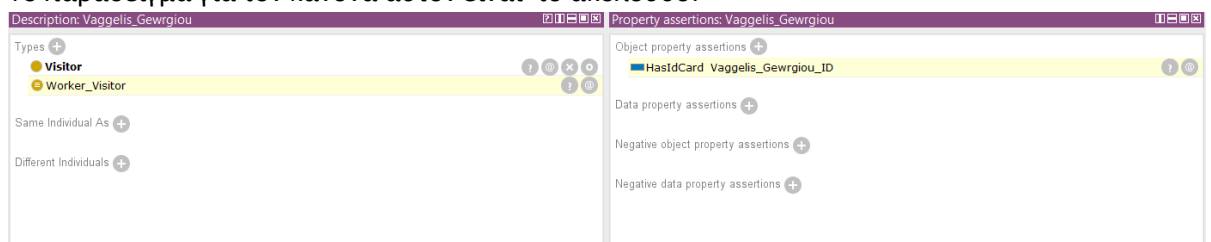
Τα άτομα της κλάσης Mammal, δηλαδή τα θηλαστικά που έχουν πούπουλα (η ιδιότητα δεδομένων HasFeathers έχει τιμή True) θα είναι Kiwi.

Το παράδειγμα για τον κανόνα αυτόν είναι το ακόλουθο:



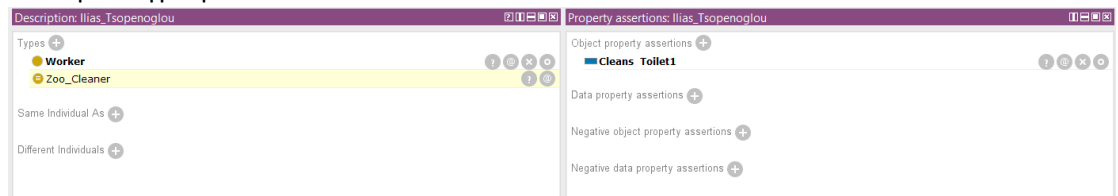
4. $\text{Visitor}(?x) \wedge \text{HasIdCard}(?x,?y) \rightarrow \text{Worker_Visitor}(?x)$
Ο επισκέπτης ο οποίος έχει ID card θα ανήκει στην κλάση των Worker_Visitor, δηλαδή των επισκεπτών που δουλεύουν στον ζωολογικό κήπο.

Το παράδειγμα για τον κανόνα αυτόν είναι το ακόλουθο:



5. $\text{Worker}(?x) \wedge \text{Cleans}(?x,?y) \wedge \text{Public_Area}(?y) \rightarrow \text{Zoo_Cleaner}(?x)$
Ο εργαζόμενος ο οποίος καθαρίζει έναν κοινόχρηστο χώρο είναι καθαριστής του ζωολογικού κήπου, δηλαδή Zoo_Cleaner.

Το παράδειγμα για τον κανόνα αυτόν είναι το ακόλουθο:



Ερώτημα 6:

Παρακάτω φαίνονται οι εξηγήσεις στους όρους που ζητούνται:

- open-world assumption: Ο συγκεκριμένος όρος σημαίνει πως η απουσία πληροφορίας δεν πρέπει να ταυτίζεται με την άρνηση της πληροφορίας. Πιο συγκεκριμένα, ένα statement μπορεί να είναι αληθές χωρίς να το γνωρίζουμε με σιγουριά, είναι δηλαδή unknown.

Ένα παράδειγμα από την οντολογία μας είναι το ακόλουθο:

Πχ. Ο Animal Feeder John Smith Feeds Tiger 1 αλλά όπως βλέπουμε και από τα screenshot αυτό δεν σημαίνει ότι δεν μπορεί και κάποιος άλλος Feeder να ταΐζει την Tiger 1.

The image displays two screenshots of the Protégé ontology editor interface. The top screenshot shows the 'Property assertions: John_Smith' panel. Under 'Object property assertions', there are six entries: 'Cares Tiger4', 'Cares Tiger3', 'Cares Tiger2', 'Cares Tiger1', 'Feeds Tiger4', and 'Feeds Tiger3'. Under 'Data property assertions', there are three entries: 'PersonAge 25', 'Salary 4000', and 'HasName "John"'. The bottom screenshot shows the 'Property assertions: Maria Ioannou' panel. It has the same structure, with 'Cares Tiger4', 'Cares Tiger3', 'Cares Tiger2', 'Cares Tiger1', 'Feeds Tiger4', and 'Feeds Tiger3' under object properties, and 'PersonAge 46', 'Salary 650', and 'HasName "Maria"' under data properties.

- non-unique-name assumption: Σύμφωνα με αυτή την υπόθεση, η διαφορετικότητα του κάθε individual πρέπει να δηλώνεται ξεκάθαρα ειδάλλως δυο individual μπορεί να θεωρηθεί ότι είναι το ίδιο.

Ένα παράδειγμα από την οντολογία μας είναι το ακόλουθο:

Έχουμε τους δύο Animal_Trainer που φαίνονται στις εικόνες οι οποίοι έχουν τις ίδιες Data Properties.

Για αυτό τον λόγο αν δεν δηλώσουμε ότι αυτά τα individual είναι διαφορετικά το ένα από το άλλο μπορεί να θεωρηθούν το ίδιο.

The image shows a screenshot of the Protégé ontology editor interface for the individual 'Giwrgos_Giannikopoulos'. The 'Property assertions' panel shows 'Object property assertions' with 'Cares Tiger4', 'Cares Tiger3', 'Cares Tiger2', 'Cares Tiger1', 'Feeds Tiger4', and 'Feeds Tiger3'. Under 'Data property assertions', there are three entries: 'PersonAge 45', 'Salary 800', and 'HasName "Giwrgos"'. The interface also shows the 'Types' panel with 'Animal_Trainer' and 'Giwrgos_Giannikopoulos'.

ΚΡΕΜΑΝΤΑΛΑ ΘΕΟΔΩΡΑ ΑΜ: 1067445
ΛΥΚΟΥΔΗΣ ΧΡΗΣΤΟΣ ΑΜ: 1067405

Description: Giwrgos_Giannakopoulos

Types

Animal_Trainer

Same Individual As

Different Individuals

Giwrgos_Giannikopoulos

Property assertions: Giwrgos_Giannakopoulos

Object property assertions

Data property assertions

PersonAge 45

HasName "Giwrgos"

Salary 800

Negative object property assertions

Negative data property assertions

Reasoner state out of sync with active ontology. ☒ Show Inferences