Project

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ
2019-2020

Ελευθεριάδης Πέτρος 6044

Ερώτημα 1

A

Η οντολογία που έφτιαξα αφορά το ηλεκτρονικό παιχνίδι “Hearthstone”, το οποίο είναι ένα digital collectible card game. Στο παιχνίδι αυτό υπάρχει μια τεράστια συλλογή από κάρτες, από τις οποίες ο παίκτης διαλέγει να φτιάξει μια τράπουλα από 30 κάρτες και μπαίνει σε μονομαχίες με άλλους παίκτες οι ποιοι έχουν τη δικιά τους τράπουλα.

Δε μας ενδιαφέρει πως γίνεται η μάχη αυτή, καθώς στην οντολογία αυτή θα ασχοληθούμε με την **κάρτα** ως οντότητα. Όπως προανέφερα στο παιχνίδι υπάρχουν πολλές κάρτες (2000+) και μπορούμε να τις χωρίσουμε σε πολλές κατηγορίες.

Η βασικότερη κατηγορία είναι αν η κάρτα είναι **minion** (τέρας) ή **spell** (ξόρκι).
Υπάρχουν και άλλα είδη καρτών αλλά δεν είναι σημαντικά και θα επικεντρωθούμε στα δυο αναφερόμενα.

Το **minion** ξεχωρίζει επειδή έχει σώμα και έχει 2 στατιστικά **attack** (επίθεση) και **health** (ζωή).
Παράδειγμα minion:

Το ”2” κάτω αριστερά δείχνει το **attack** και το ”3” το **health**

Το **spell** δεν έχει σώμα. Είναι απλά μια κάρτα με ένα **text** που κάνει κάτι.
Παράδειγμα spell:

Το spell αυτό κάνει 6 ζημιά σε ένα στόχο.

Κάθε κάρτα, είτε minion είτε spell έχει ένα κόστος "**cost**" για να παιχτεί
Αυτό το κόστος φαίνεται πάνω αριστερά στη κάρτα.

Ο δεύτερος τρόπος που θα διαχωρίσουμε τις κάρτες είναι με τους **heroes.** Στο παιχνίδι υπάρχουν 9 διαφορετικοί heroes. Κάθε hero έχει κάρτες μοναδικές για αυτόν που δε μπορούν να χρησιμοποιήσουν οι άλλοι. Επίσης υπάρχουν και οι Neutral cards που μπορούν να χρησιμοποιηθούν από ολα τα heroes.

Παράδειγμα.
Αν θέλω να φτιάξω μια τράπουλα για τον hero "Priest", μπορω να χρησιμοποιήσω μόνο κάρτες του Priest και Neutral cards.

Στην οντολογία έβαλα μόνο 3 heroes (Priest, Warrior, Hunter) για να μη γίνει αχρείαστα μεγάλη.
Επίσης έκανα τις τράπουλες να αποτελούνται από 5 κάρτες αντί για 30.

Β

- Θα καλύπτει τη συλλογή καρτών του παιχνιδιού "Hearthstone"
- Για να μπορεί ο οποιοσδήποτε να έχει μια βάση δεδομένων στο ίντερνετ και να μπορεί να ψάχνει τις διάφορες κάρτες που τον ενδιαφέρουν και να βρίσκει πληροφορίες για αυτές, χωρίς να χρειάζεται να μπει στο παιχνίδι. Επίσης μπορεί να βρίσκει ποιες κάρτες ταιριάζουν με άλλες ώστε να φτιάξει μια πιο "δυνατή" τράπουλα
- Ποια τέρατα έχουν 5 attack και πανω?, ποιο spell του τάδε hero κοστίζει λιγότερο από 3?, δείξε μου όλα τα minion του ταδε hero με τόσο Health. Ποια spells ταιριάζουν με αυτό το minion? Κλπ
- Αν μια κάρτα ταιριάζει με μια άλλη, και εκείνη με μια τρίτη θα μπορούσε ο χρήστης να βρει έναν ωραίο συνδυασμό καρτών. Αν μια κάρτα εχει μεγαλύτερη επίθεση από μια άλλη, και εκείνη μεγαλύτερη από άλλη κλπ.

Μια ιστοσελίδα database από κάρτες που κάθε ιδιότητα της κάρτας θα ηταν ένα resource και θα μπορούσε κάποιος να βρει πολύ εύκολα αυτό που ψάχνει. Εύκολη ταξινόμηση καρτών σε διάφορα φίλτρα. Κάρτες που βρίσκονται πιο συχνά σε τράπουλες (το site θα ειχε ένα deck builder όπου χρήστες θα έφτιαχναν decks).

Γ

Card: Είναι η καρτα. Εχει παντα καποιο κοστος

Hero_Card: Η καρτα που ανήκει σε καποιο hero

Hero_Minion: Το minion που ανήκει σε καποιο hero

Hunter_Card: Καρτα που ανήκει στο hero Hunter

Hunter_Minion: Minion του Hunter

Hunter_Spell: Spell του Hunter

Priest_Card: Καρτα που ανήκει στο hero Priest

Priest_Minion: Minion του Priest

Priest_Spell: Spell του Priest

Warrior_Card: Καρτα που ανήκει στο hero Warrior

Warrior_Minion: Minion του Warrior

Warrior_Spell: Spell του Warrior

Minion: Καρτα και έχει attack και health

Neutral_Card: Καρτα που δεν είναι hero card

Neutral_Minion: Neutral Minion

Spell: Καρτα που δεν είναι Minion

Deck: περιέχει ακριβώς 5 κάρτες

Hunter_Deck: περιέχει τουλάχιστον μια κάρτα Hunter

Priest_Deck: περιέχει τουλάχιστον μια κάρτα Priest

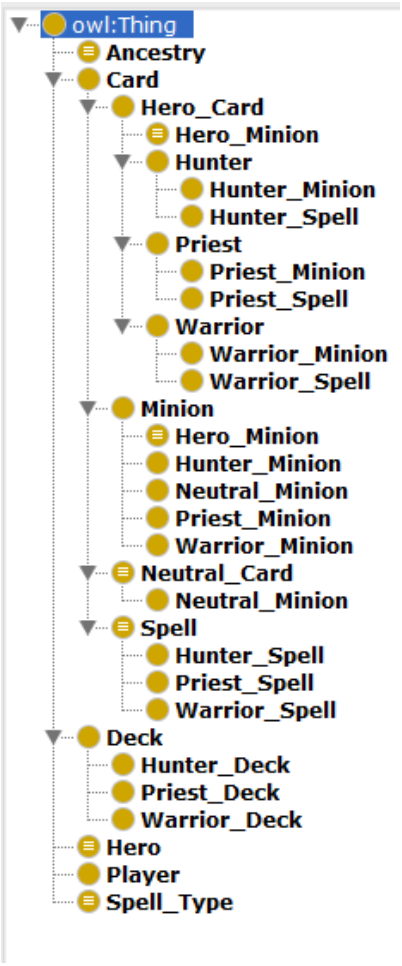
Warrior_Deck: περιέχει τουλάχιστον μια κάρτα Warrior

Hero: εχει 3 hero individuals και ένα neutral

Player: Αυτός που χρησιμοποιεί τα deck

Spell_Type: εχει individuals για τύπου spell

Ancestry: εχει individuals για τύπου minion



Δ

Data Properties

Cost: το κόστος να παιχτεί η κάρτα. Functional

Deck_name: Όνομα για deck

Minion_text: Αφορά το text των minion

Actual_text: Είναι το text που αναγράφεται στο minion. Functional

Keyword: Έχει να κάνει με το παιχνίδι. Καθορίζει το πότε θα γίνει αυτό που αναγράφεται στο actual_text ή κάποια άλλη ιδιαιτερότητα. Παίρνει τιμές “Battlecry, Deathrattle, Charge”

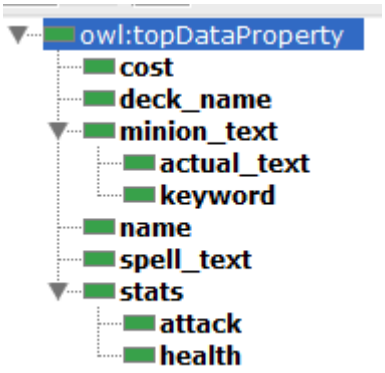
Name: Το όνομα της κάρτας. Functional

Spell_text: text για τα spells. Functional

Stats: περιέχει τα attack και health του minion

Attack: αριθμός της επίθεσης. Functional

Health: αριθμός της ζωής. Functional



Object Properties

Costs_more_than: Transitive

Costs_less_than: Transitive, Inverse of “costs_more_than”

Costs_same_as: Symmetric

Has-more_attack_than: Transitive

Has-less_attack_than: Inverse of “has-more_attack_than”

Has_ancestry: Beast, Mech, Pirate

Has_same_hp: Symmetric

Has_spell_type: Αφορά το τι κάνει το spell. Heal, Damage, Destroy.

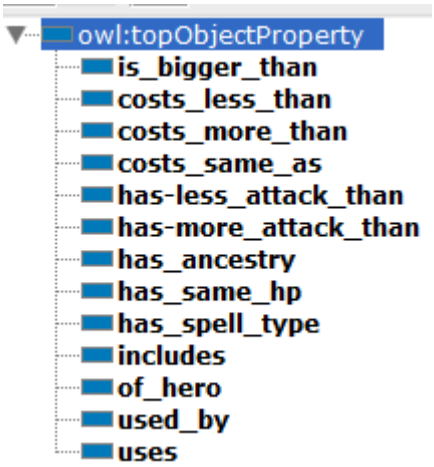
Includes: Με domain “Deck” και range “Card”, δείχνουμε ποιες κάρτες περιέχει ένα deck.

Of_hero: Domain=Card, range=Hero. Functional

Uses: Domain=Player, range=deck. Δείχνει ποιο deck χρησιμοποιεί ο layer. Functional

Used_by: Inverse of uses. Inverse Functional.

Is_bigger_than: Χρησιμοποίησα SWRL για να συνδυάσω το Has_same_hp και Has-more_attack_than και αν ισχύουν και τα δυο τότε ισχύει το is_bigger_than.



SWRL

- Minion(?a) ^ has_same_hp(?a, ?b) ^ has-more_attack_than(?a, ?b) -> is_bigger_than(?a, ?b)

Το εξηγησα πανω

- Card(?x) ^ costs_same_as(?x, ?y) ^ costs_more_than(?y, ?z) -> costs_more_than(?x, ?z)

Εδώ, περα από τη transitive ιδιοτητα της costs_more_than, ηθελα να εκμεταλλευτω και το costs_same_as για να εχω περισσοτερες inferred πληροφοριες

- Card(?x) ^ costs_same_as(?x, ?y) ^ costs_less_than(?y, ?z) -> costs_less_than(?x, ?z)

Το ίδιο κι εδώ απλά με costs_less_than.

Εννοείται ότι παρόμοιες ιδιότητες και SWRL θα μπορούσα να είχα κάνει για τις ιδιότητες attack και health. Δηλαδή has_more_hp, has_less_hp, has_same_attack αλλά η ιδέα δεν αλλάζει οπότε δεν τα έκανα.

Ε

Κάποια παραδείγματα αντικειμένων.

- Priest_Minion: Northshire_Cleric.

Ιδιότητες:

cost 1,

attack 1,

name "Northshire Cleric"^^rdfs:Literal,

actual_text "Whenever a minion is healed, draw a card."^^rdfs:Literal,

health 3

costs_less_than Hunter's_Mark

costs_same_as Whirlwind

costs_more_than Circle_of_Healing

- Warrior_Spell: Brawl

Ιδιότητες:

spell_text "Destroy all minions except one. (chosen randomly)"^^rdfs:Literal,

cost 5,

name "Brawl"^^rdfs:Literal

has_spell_type Destroy

costs_more_than Bash

costs_less_than Dr._Boom

- Hunter_Minion: Savannah_Highmane

Ιδιοτητες:

keyword "Deathrattle"

Name "Savannah Highmane"^^rdfs:Literal

actual_text "Summon two 2/2 Hyenas."^^rdfs:Literal

Health 5

Cost 6

Attack 6

has_same_hp Cabal_Shadow_Priest

costs_less_than Dr._Boom

has_ancestry Beast

has-more_attack_than Cabal_Shadow_Priest

costs_same_as Cabal_Shadow_Priest

costs_more_than Sylvanas_Windrunner

- Neutral_Minion: Injured_Blademaster

Ιδιοτητες:

actual_text "Deal 4 damage to HIMSELF."^^rdfs:Literal

Keyword "Battlecry"

Health 7

Attack 4

Cost 3

Name "Injured Blademaster"^^rdfs:Literal

costs_less_than Chillwind_Yeti

has-more_attack_than Wild_Pyromancer

costs_same_as Animal_Companion

costs_more_than Armorsmith

- Priest_Deck: Priest1

Includes Wild_Pyromancer

Includes Lightbomb

Includes Dr._Boom

Includes Northshire_Cleric

Includes Injured_Blademaster

used_by Petros

- Player: Opponent

Uses hunter1

Ερώτημα 2

- **16 κλάσεις οργανωμένες σε τουλάχιστον τρία επίπεδα ιεραρχίας** (υποκλάσεων)

Τις ανεφερα πανω

- 6 κλάσεις να αποτελούν υποκλάσεις άλλων (Subsumption) -

Hunter_Minion
Hunter_Spell
Priest_Card
Priest_Minion
Priest_Spell
Warrior_Card
Warrior_Minion

6 κλάσεις να είναι ξένες μεταξύ τους (Disjointness) -

Hunter_Minion
Priest_Minion
Warrior_Minion
Neutral_Minion

6 κλάσεις να προκύπτουν από λογική σύνθεση άλλων: Χρησιμοποιήστε τουλάχιστον δύο φορές καθεμία από τις παρακάτω πράξεις:

τομή (Intersection)

Hunter_Minion
Hunter_Spell
Priest_Card
Priest_Minion
Priest_Spell
Warrior_Card
Warrior_Minion

ένωση (Union)

συμπλήρωμα (Complement)

Spell
Neutral_Card

existential restriction (someValuesFrom)

Hunter_Deck
Priest_Deck
Warrior_Deck

universal restriction (allValuesFrom)

Deck
Player

hasValue

Hunter_Card

Priest_Card
Warrior_Card
Neutral_Card

Minimum/Maximum Cardinality

Deck

- 2 κλάσεις να προκύπτουν από συνδυασμό λογικών πράξεων και περιορισμών σε σχέσεις.

- 16 ιδιότητες οργανωμένες σε τουλάχιστον δυο επίπεδα ιεραρχίας. Να υπάρχουν ιδιότητες και των δυο τύπων (datatypeκαι objectProperties) σε ποσοστό 30% τουλάχιστον από το καθένα.

Τις ανεφερα στο ερωτημα 1

Επίσης από αυτές θα πρέπει τουλάχιστον:

4 ιδιότητες να αποτελούν subproperties άλλων ιδιοτήτων
Attack
Health
Actual_text
keyword

4 ιδιότητες να οριστούν με τις αντίστοιχες αντίστροφες (inverse)
costs_less_than
has-less_attack_than
used_by

2 ιδιότητες να είναι συμμετρικές (symmetric)
has_same_hp
costs_same_as

2 ιδιότητες να είναι μεταβατικές(transitive)
costs_more_than
has-more_attack_than
is_bigger_than

2 ιδιότητες να είναι συναρτησιακές (functional)
Of_hero
Name
cost

2 ιδιότητες να είναι inverse functional
Uses
Used_by

Ερώτημα 3

Hunter_Spell: Animal_Companion

Animal_Companion type of_hero Hunter

Το spell αυτό είναι του hero Hunter.

Αυτό ισχύει επειδή το Hunter_Spell είναι subclass του class Hunter_Card και αυτό με τη σειρά του έχει συνθήκη “of_hero value Hunter”

Neutral_Minion: Sylvanas_Windrunner

Sylvanas_Windrunner costs_more_than Flash_Heal

Η πρώτη κάρτα κοστίζει παραπάνω από τη δεύτερη

Αυτό ισχύει λόγω της Transitive ιδιότητας costs_more_than καθώς η κάρτα αυτή κοστίζει περισσότερο από μια άλλη και εκείνη περισσότερο από τη Flash_Heal

Priest_Spell: Circle_of_Healing

Circle_of_Healing costs_less_than Northshire_Cleric

Το spell κοστιζει λιγοτερο από τη κάρτα Northshire_Cleric

Αυτό ισχύει επειδή υπάρχει η γνώση πως η Northshire_Cleric κοστίζει περισσότερο από το Circle_of_Healing και η costs_less_than είναι inverse of costs_more_than

Priest_Spell: Flash_Heal

Arcane_Shot costs_same_as Flash_Heal

Το πρώτο spell κοστίζει το ίδιο με το δεύτερο

Αυτό ισχύει λόγω της symmetric ιδιοτητας costs_same_as και υπάρχει γνώση πως το δεύτερο κοστίζει το ίδιο με το πρωτο.

Hunter_Minion: Savannah_Highmane

Savannah_Highmane has-more_attack_than Cruel_Taskmaster

Το πρώτο minion εχει περισσότερη επίθεση από το δεύτερο.

Ισχύει λόγω της transitive ιδιότητας has-more_attack_than καθώς υπάρχει γνώση πως το Savannah_Highmane εχει περισσότερη επίθεση από ένα άλλο το οποίο έχει περισσότερη επίθεση από το Cruel_Taskmaster

Priest_Minion: Cabal_Shadow_Priest

Cabal_Shadow_Priest has-less_attack_than Savannah_Highmane

Το πρώτο minion έχει λιγότερη επίθεση από το δεύτερο

Ισχύει επειδή η has-less_attack_than είναι inverse of has-more_attack_than και υπάρχει γνώση πως το δεύτερο έχει περισσότερη επίθεση απο το πρώτο.

Neutral_Minion: Sylvanas_Windrunner

Sylvanas_Windrunner has_same_hp Chillwind_Yeti

Το πρώτο minion έχει ίδια ζωή με το δεύτερο.

Ισχύει λόγω της symmetric has_same_hp και υπάρχει γνώση πως το δεύτερο έχει ίδια ζωή με το πρώτο.

Neutral_Minion: Sylvanas_Windrunner

Sylvanas_Windrunner costs_more_than Flash_Heal

Η πρώτη κάρτα κοστίζει περισσότερο από τη δεύτερη.

Αυτό ισχύει λογω του κανόνα SWRL: Card(?x) ^ costs_same_as(?x, ?y) ^ costs_more_than(?y, ?z) -> costs_more_than(?x, ?z), καθώς η πρώτη κάρτα κοστίζει το ίδιο με μια άλλη, η οποία κοστίζει περισσότερο απο τη Flash_Heal.

Priest_Minion: Cabal_Shadow_Priest

Cabal_Shadow_Priest costs_less_than King_Krush

Η πρώτη κάρτα κοστίζει λιγότερο από τη δευτερη.

Αυτό ισχύει λόγω του κανόνα SWRL: Card(?x) ^ costs_same_as(?x, ?y) ^ costs_less_than(?y, ?z) -> costs_less_than(?x, ?z), καθώς η πρώτη κάρτα κοστίζει το ίδιο με μια άλλη, η οποία κοστίζει λιγότερο απο τη King_Krush.

Hunter_Minion: Savannah_Highmane

Savannah_Highmane is_bigger_than Cabal_Shadow_Priest

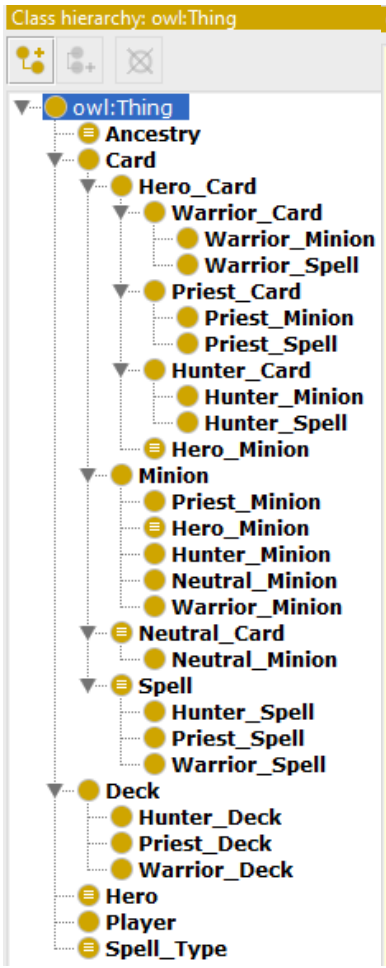
Το πρώτο minion είναι μεγαλύτερο από το δεύτερό.

Ισχύει λόγω του κανόνα SWRL: Minion(?a) ^ has_same_hp(?a, ?b) ^ has-more_attack_than(?a, ?b) -> is_bigger_than(?a, ?b), καθώς το πρώτο έχει ίδια ζωή με το δεύτερο αλλά και μεγαλύτερη επίθεση.

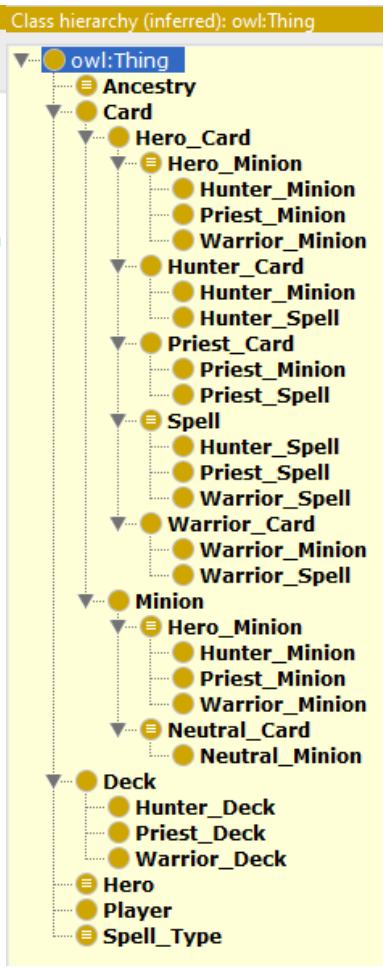
Ερωτημα 4

A

Asserted Hierarchy



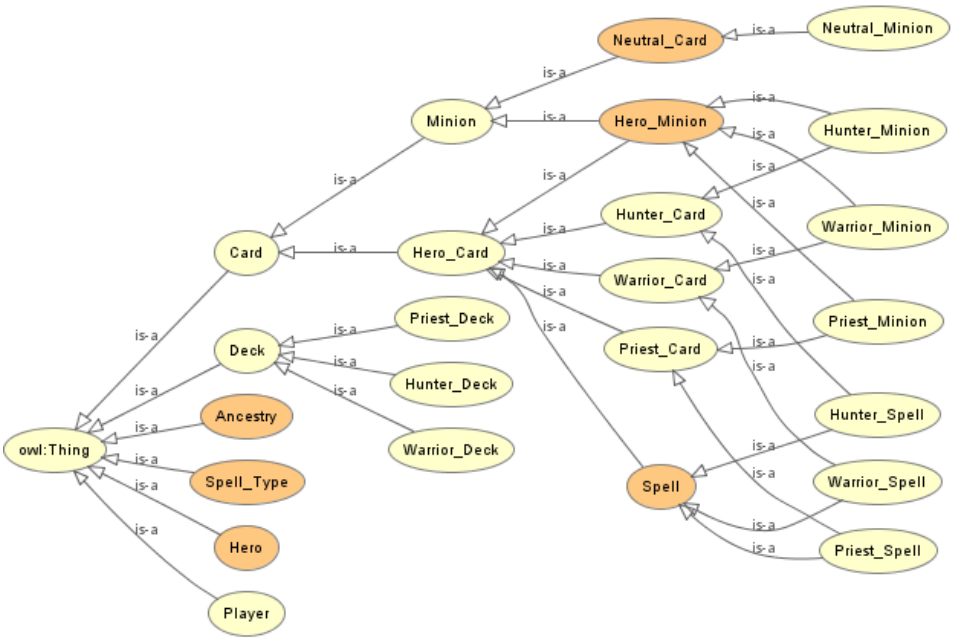
Inferred Hierarchy



B

Asserted Graph

Inferred Graph



C

Μια διάφορα είναι πως η κλάση Hero_Minion στο inferred έχει υποκλασεις τα Priest_ Minion Warrior_ Minion και Hunter_ Minion το οποίο είναι σωστό.

Μια άλλη είναι πως στο inferred εχει μπει η κλαση spell ως υποκλαση του Hero_card. Το οποίο είναι επίσης σωστο. Δεν υπάρχουν neutral spells. Κάθε spell πρέπει να ανήκει σε καποιο hero

Η κλάση Minion έχει χωριστεί σε 2 υποκλασεις, hero_minion και neutral_card και πλέον το hero_minion έχει μέσα τα minion του κάθε hero. Επίσης κατάλαβε ότι η κλάση neutral_card είναι πάντα minion και την έβαλε ως υποκλαση του minion.

Ερωτημα 5

Q1

Το πρώτο query ρωτάει να βρει το minion με επίθεση μεγαλύτερη του 5. Επίσης ζητάει διάφορες optional πληροφορίες για το minion. Ταξινομεί με βάση την επίθεση.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX hs: <http://www.semanticweb.org/petros/ontologies/2020/0/untitled-ontology-9#>
PREFIX foaf: <http://xmlns.com/foaf/spec/>

SELECT ?minion ?attack ?ancestry ?hero ?keyword ?text

WHERE{ ?minion rdf:type hs:Minion.
?minion hs:attack ?attack;
hs:of_hero ?hero.
FILTER (?attack > 5)
OPTIONAL {?minion hs:has_ancestry ?ancestry}
OPTIONAL {?minion hs:keyword ?keyword}
OPTIONAL {?minion hs:actual_text ?text}
}
ORDER BY ?attack
```

?minion	?attack	?ancestry
hs:Savannah_Highmane	6	hs:Beast
hs:Dr._Boom	7	
hs:King_Krush	8	hs:Beast

3 results

?hero	?keyword	?text
hs:Hunter	Deathrattle^^xsd:string	Summon two 2/2 Hyenas.^^rdfs:Literal
hs:Neutral	Battlecry^^xsd:string	Summon two 1/1 Boom Bots. WARNING: ...
hs:Hunter	Charge^^xsd:string	

Q2

Αυτό το query ψάχνει να βρει τα spells που κοστίζουν κάτω από 4. Επίσης ζηταει το text και, αν έχει, τον τύπο του spell.

```
18. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
19. PREFIX owl: <http://www.w3.org/2002/07/owl#>
20. PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
21. PREFIX hs: <http://www.semanticweb.org/petros/ontologies/2020/0/untitled-ontology-9#>
22. PREFIX foaf: <http://xmlns.com/foaf/spec/>
23.
24. SELECT ?spell ?cost ?type ?text
25.
26. WHERE{ ?spell rdf:type hs:Spell.
27. ?spell hs:cost ?cost;
28. hs:spell_text ?text.
29. FILTER (?cost < 4)
30. OPTIONAL {?spell hs:has_spell_type ?type}
31.
32. }
33. ORDER BY ?cost
```

?spell	?cost
hs:Circle_of_Healing	0
hs:Whirlwind	1
hs:Flash_Heal	1
hs:Arcane_Shot	1
hs:Hunter's_Mark	2
hs:Animal_Companion	3
hs:Bash	3

7 results

?type	?text
hs:Heal	Restore 4 Health to ALL minions.^^rdfs:Literal
hs:Damage	Deal 1 damage to ALL minions.^^rdfs:Literal
hs:Heal	Restore 5 Health.^^rdfs:Literal
hs:Damage	Deal 2 damage.^^rdfs:Literal
	Change a minion's Health to 1.^^rdfs:Literal
	Summon a random Beast Companion^^rdfs:Literal
hs:Damage	Deal 3 damage. Gain 3 Armor.^^rdfs:Literal

Q3

Το query αυτό ψάχνει να βρει ολες τις κάρτες που ανήκουν η στο Hero “Priest” η στο hero “Warrior”, και ζητάει επίσης το κόστος. Ταξινόμηση βάση κόστους.

```
34. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
35. PREFIX owl: <http://www.w3.org/2002/07/owl#>
36. PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
37. PREFIX hs: <http://www.semanticweb.org/petros/ontologies/2020/0/untitled-ontology-9#>
38.
39. SELECT ?card ?hero ?cost
40.
41. WHERE{
42. {
43. ?card rdf:type hs:Priest_Card;
44. hs:cost ?cost;
45. hs:of_hero ?hero.
46. }
47. UNION
48. {
49. ?card rdf:type hs:Warrior_Card;
50. hs:cost ?cost;
51. hs:of_hero ?hero.
52. }
53. }
54. ORDER BY ?cost
```

?card	?hero	?cost
hs:Circle_of_Healing	hs:Priest	0
hs:Northshire_Cleric	hs:Priest	1
hs:Flash_Heal	hs:Priest	1
hs:Whirlwind	hs:Warrior	1
hs:Cruel_Taskmaster	hs:Warrior	2
hs:Armorsmith	hs:Warrior	2
hs:Bash	hs:Warrior	3
hs:Brawl	hs:Warrior	5
hs:Lightbomb	hs:Priest	6
hs:Cabal_Shadow_Priest	hs:Priest	6

10 results

Q4

Ζητάει να βρει τη μεγαλύτερη ζωή σε minion.

```
55. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
56. PREFIX owl: <http://www.w3.org/2002/07/owl#>
57. PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
58. PREFIX hs: <http://www.semanticweb.org/petros/ontologies/2020/0/untitled-ontology-9#>
59.
60. SELECT  (max(?health) as ?maxhealth)
```

```
61.
62. WHERE{ ?minion rdf:type hs:Minion;
63. hs:health ?health.
64.
65. }
```

?maxhealth
8.0

Q5

Ζητάει να βρει ποιες κάρτες συμπεριλαμβάνονται σε κάποιο deck. Το distinct βοηθάει σε περιπτώσεις που μια κάρτα άνηκε σε 2 decks, θα εμφανιζόταν 2 φορές.

Στο συγκεκριμένο παράδειγμα χωρίς το DISTINCT, βγάζει 15 αποτελέσματα. Ενώ με το DISTINCT βγάζει 12

```
66. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
67. PREFIX owl: <http://www.w3.org/2002/07/owl#>
68. PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
69. PREFIX hs: <http://www.semanticweb.org/petros/ontologies/2020/0/untitled-ontology-9#>
70. PREFIX foaf: <http://xmlns.com/foaf/spec/>
71.
72. SELECT  DISTINCT ?card
73.
74. WHERE{ ?deck rdf:type hs:Deck.
75. ?deck hs:includes ?card
76. }
```

?card
hs:Dr._Boom
hs:Chillwind_Yeti
hs:Whirlwind
hs:Sylvanas_Windrunner
hs:Injured_Blademaster
hs:Hunter's_Mark
hs:Animal_Companion
hs:King_Krush
hs:Arcane_Shot
hs:Wild_Pyromancer
hs:Lightbomb
hs:Northshire_Cleric

12 results

Ερώτημα 6

Open World Assumption σημαίνει πως δε μπορούμε να θεωρούμε ότι κάτι δεν ισχύει χωρίς να αναφέρεται ξεκάθαρα ότι δεν ισχύει. Δε μπορούμε να βγάλουμε συμπέρασμα για την αλήθεια μιας πρότασης επειδή υπάρχει έλλειψη γνώσης.

Ένα παράδειγμα στην οντολογία που έφτιαξα θα ήταν αν έφτιαχνα ένα individual για τη κλαση Hero_Card και της έδινα ένα cost και of_hero Priest. Επειδή δεν της έχω δώσει τιμές στα attack και health δε σημαίνει πως αυτή η κάρτα δεν είναι minion, δηλαδή είναι spell. Αυτό φαίνεται κιολας στα sparql queries. Δεν εμφανίζεται στις αναζητήσεις για minion ούτε για spell. Μόνο αν ψάξω για κάρτα θα το βρει, επειδή έχω αναφέρει ότι είναι κάρτα αλλά δεν έχω αναφέρει αν είναι spell ή minion.

Non Unique Name Assumption. Δυο διαφορετικά ονόματα δε σημαίνει πως είναι διαφορετικές οντότητες. Δυο ονόματα μπορούν να αναφέρονται στο ίδιο αντικείμενο.

Για παράδειγμα, στην οντολογία μου έχω το αντικείμενο “Petros” για τη κλάση Player. Ο Petros χρησιμοποιεί το deck Priest1 με το property “uses” που είναι inverse functional. Αν φτιάξω έναν Test_Player και τον βάλω να χρησιμοποίει το deck που χρησιμοποιεί ο Petros, τότε το σύστημα συμπεραίνει ότι ο Test_Player είναι ο Petros, παρόλο που έχουν διαφορετικά ονόματα.