



Knowledge-Based Systems

Laboratory activity

Ontology title: Hunting ontology

Students: Trif Gheorghe Andrei - Pintilei Ovidiu

Emails: trifandrei@yahoo.com - pintilei.ovidiu09@gmail.com

Assoc. Prof.dr. eng. Adrian Groza
Adrian.Groza@cs.utcluj.ro

Contents

1	Hunting ontology documentation	3
1.1	Competency questions	3
1.2	Use cases	3
1.3	Related ontologies	3
1.4	Tboxes	4
1.5	Aboxes	5
1.6	Rules	6
1.7	Queries	8
1.8	Ontology evaluation	9
2	DL-Learner	10
2.1	Configuration and execution	10
2.2	Results	11
3	Java API	12
3.1	Configuration and execution	12
3.2	Results	13
4	Fuzzy Description Logic	14
5	Ontology Design Patterns	15
5.1	Partition Pattern	15
5.2	N-ary relations	15
5.3	PartOf	15
6	Querying your ontology	17
6.1	Racer	17
6.2	NRQL	18
A	Racer code	19
B	DL-learner code	20
C	Fuzzy code	22
D	Java API code	23

Chapter 1

Hunting ontology documentation

1.1 Competency questions

1. How do we find an animal given its foot print?
2. Get all hunters from country X.
3. What are the calibers for each weapon?
4. Which caliber is required to hunt animal Y?
5. What are each animal's foot prints?
6. What is the personal information of the hunter X?
7. Check if X is a hunter.
8. What are the foot-prints of all animals?

1.2 Use cases

The fields in which the ontology can be applied:

- help hunters with information about weapons, animals and hunting experience
- help hunters with current legislature
- help people who try to buy a weapon and learn more about hunting

1.3 Related ontologies

1. <https://dbpedia.org/ontology/weapon>
2. <https://dbpedia.org/ontology/Animal>

The enumerated ontologies are not used in the project.

1.4 Tboxes

In this project, we defined the following concepts regarding hunting domain: weapon, caliber, animal, country, terrain, foot-print, hunter, food.

A weapon is the most important concept from the hunting domain because it fulfills the hunting activity. The weapon concept uses another concept called caliber which is specified in the following line:

(implies weapon (all has-caliber caliber)) - this concept implies that all weapons must have a caliber. Calibers can have different instances which will be defined in the next section. Also, different weapons can have different calibers.

Another important concept is animal which has as a sub-concept the foot-print concept.

(define-primitive-role has-foot-print :domain animal :range foot-print)
(implies animal(all has-foot-print foot-print)) - this implies that all animals must have a different foot-print model.

Also, 'animal' has 3 related concepts: caliber, terrain and food.

(define-primitive-role caliber-for-animal :domain caliber :range animal) - different animals require different calibers because some animals are smaller and they don't need a big sized caliber and other larger animals require caliber for their size. This role helps the hunter into choosing the optimal caliber for the animal which he is going to hunt.

(define-primitive-role terrain-for-animal :domain terrain :range animal :feature t) - because there are different types of animal species, these can not live in the same habitat (e.g. bear in the mountains, rabbit in plain). For this reason, we need this role to specify the habitat in which an animal can live and to help the hunter into where to go look for the wanted prey.

(define-concept Food (or grass honey)) - different animals eat different foods

The next concept is country which is important because we can find the nationality of a hunter.

(define-primitive-role has-address :domain Hunter :range Country) - this is a sub-concept of the hunter.

The hunter is the main actor of our ontology. This concept can have name, address and age.

(instance Trif Hunter)
(attribute-filler Trif "Andrei" has-name)
(attribute-filler Trif 23 has-age)

Here are all the roles that we use in our ontologies:

(define-primitive-role has-foot-print :domain animal :range foot-print)
(define-primitive-role caliber-for-animal :domain caliber :range animal)
(define-primitive-role caliber-for-foot-print :domain caliber :range foot-print)
(define-primitive-role terrain-for-animal :domain terrain :range animal :feature t)
(define-concrete-domain-attribute has-name :domain Hunter :type string)
(define-concrete-domain-attribute has-age :domain Hunter :type integer)

(define-primitive-role has-address :domain Hunter :range Country)
(define-primitive-role has-part :transitive t :inverse is-part-of :domain Food)

One role has a domain and a range, for example caliber-for-animal has domain caliber and range animal.

1.5 Aboxes

In this section it is presented the ABox part of the ontology. The ABox is the component which refers to the instances through which the ontology has been populated and the different connections between them. Knowing this, in our ontology there are described the relations between weapons, calibers, animals, foot-prints etc. In Racer these instances are declared using the 'instance' key-word and for relations it is used the 'related' key-word. Relations refer to the concept of more instances being tied together.

Next we will present all the instances for every concept described in the Tboxes section.

For the weapon concept we have the following instances and relation:

(instance sniper weapon)
(instance shotgun weapon)
(instance pistol weapon)

First we have 3 instances of weapon and these are sniper, pistol and shotgun. These 3 weapons each have a different caliber specified in the following 3 relations:

(related sniper cal365mm has-caliber)
(related shotgun cal66mm has-caliber)
(related pistol cal12mm has-caliber)

Also the cal365mm, cal66mm and cal12mm are instances of the caliber concept.

(instance cal365mm caliber)
(instance cal66mm caliber)
(instance cal12mm caliber)

The next 3 instances represent the animal imported into our ontology:

(instance bear animal)
(instance deer animal)
(instance rabbit animal)

Here we have 3 animals bear, deer and rabbit. These instances are related with the caliber domain because we try to create a method through which we can get the optimal caliber for each animal.

(related cal365mm bear caliber-for-animal)
(related cal66mm deer caliber-for-animal)
(related cal12mm rabbit caliber-for-animal)

They are also related to the terrain domain because each animal lives in a different area.

(related mountain bear terrain-for-animal)
(related plain rabbit terrain-for-animal)
(related forest deer terrain-for-animal)

Also deer-foot-print, bear-foot-print and rabbit-foot-print are all instances of foot-print concept.

```
(instance deer-foot-print foot-print)
(instance bear-foot-print foot-print)
(instance rabbit-foot-print foot-print)
```

All instances of animal have a sub-concept called foot-print. So naturally they all have different foot-prints.

```
(related deer deer-foot-print has-foot-print)
(related bear bear-foot-print has-foot-print)
(related rabbit rabbit-foot-print has-foot-print)
```

The next instances are for the Country domain. We defined 3 different country instances. These specify the locations of different hunters.

```
(instance Romania Country)
(instance Sweden Country)
(instance Russia Country)
```

The hunter instances can have 3 attributes called name, address and age. As you can see the in the following lines, we map a hunter named Trif who has the age of 23 and is from Romania.

```
(instance Trif Hunter)
(attribute-filler Trif "Andrei" has-name )
(attribute-filler Trif 23 has-age)
(related Trif Romania has-address)
```

Next we have the instances for the food concept. These are different from each another so we use the relation constraint called disjoint.

```
(disjoint grass honey)
```

1.6 Rules

```
;definim regula calibru pentru o urma
(define-rule (?x ?z caliber-for-foot-print)
  (and (?x ?y caliber-for-animal)
    (?y ?z has-foot-print )))
(run-all-rules)
```

```
;regula arma for terrain
(define-rule (?x ?z weapon-for-terrain)
  (and (?x ?y1 has-caliber)
    (?y1 ?y2 caliber-for-animal)
    (?z ?y2 terrain-for-animal)))
(run-all-rules)
```

```
;reguli pentru arma unui animal
(define-rule (?x animal-has-weapon )
  (and (?x weapon)
```

```

(rabbit animal)
(?y rabbit caliber-for-animal)
(?x ?y has-caliber)))
(run-all-rules)

(define-rule (?x animal-has-weapon )
  (and (?x weapon)
    (deer animal)
    (?y deer caliber-for-animal)
    (?x ?y has-caliber)))
(run-all-rules)

(define-rule (?x animal-has-weapon )
  (and (?x weapon)
    (bear animal)
    (?y bear caliber-for-animal)
    (?x ?y has-caliber)))
(run-all-rules)

(define-rule (?y ?x caliber-for-terrain)
  (and (?x terrain)
    (?z animal)
    (?y ?z caliber-for-animal)
    (?x ?z terrain-for-animal)))
(run-all-rules)

;Ce vaneaza trif?
(define-rule (Trif ?y animal-hunt )
  (and (Trif Hunter)
    (?y animal)
    (Trif ?z has-address)
    (?y ?z animal-has-country)))
(run-all-rules)

;Ce arma foloseste
(define-rule (Trif ?t use-weapon )
  (and (Trif Hunter)
    (?y animal)
    (Trif ?z has-address)
    (?y ?z animal-has-country)
    (?x ?y caliber-for-animal)
    (?t ?x has-caliber)))
(run-all-rules)

;Ce teren predomina in tara x
(define-rule (?x ?t dominant-terrain )
  (and (?x Country)
    (?t ?y terrain-for-animal)
    (?y ?x animal-has-country)))
(run-all-rules)

```

```

;Unde este Trif acum?
(define-rule (Trif ?x is-where )
  (and (?x Country)
    (Trif ?y animal-hunt)
    (?y ?x animal-has-country)))
(run-all-rules)

```

1.7 Queries

1. How do we find an animal given its foot print?

```

(RETRIEVE (?X) (AND (?X ANIMAL) (DEER-FOOT-PRINT FOOT-PRINT) (?X DEER-FOOT-PRINT HAS-FOOT-PRINT))) --> (((?X DEER)))
(RETRIEVE (?X) (AND (?X ANIMAL) (BEAR-FOOT-PRINT FOOT-PRINT) (?X BEAR-FOOT-PRINT HAS-FOOT-PRINT))) --> (((?X BEAR)))
(RETRIEVE (?X) (AND (?X ANIMAL) (RABBIT-FOOT-PRINT FOOT-PRINT) (?X RABBIT-FOOT-PRINT HAS-FOOT-PRINT))) --> (((?X RABBIT)))

```

2. Get all hunters from Romania.

```

(RETRIEVE (?X) (AND (?X HUNTER) (ROMANIA COUNTRY) (?X ROMANIA HAS-ADDRESS))) --> (((?X TRIF)))

```

3. What are the calibers for each weapon?

```

(INDIVIDUAL-FILLERS SNIPER HAS-CALIBER) --> (CAL365MM)
(INDIVIDUAL-FILLERS PISTOL HAS-CALIBER) --> (CAL12MM)
(INDIVIDUAL-FILLERS SHOTGUN HAS-CALIBER) --> (CAL66MM)

```

4. Which caliber is required to hunt a bear?

```

(INDIVIDUAL-FILLERS BEAR (INV CALIBER-FOR-ANIMAL)) --> (CAL365MM)

```

5. What are each animal's foot prints?

```

(INDIVIDUAL-FILLERS DEER HAS-FOOT-PRINT) --> (DEER-FOOT-PRINT)
(INDIVIDUAL-FILLERS BEAR HAS-FOOT-PRINT) --> (BEAR-FOOT-PRINT)
(INDIVIDUAL-FILLERS RABBIT HAS-FOOT-PRINT) --> (RABBIT-FOOT-PRINT)

```

6. What is the personal information of the hunter Trif?

```

(INDIVIDUAL-FILLERS TRIF HAS-ADDRESS) --> (ROMANIA)
(INDIVIDUAL-TOLD-ATTRIBUTE-VALUE TRIF HAS-AGE) --> 23

```

7. Check if Trif is a hunter.

```

(INDIVIDUAL-INSTANCE? TRIF HUNTER) --> T

```

8. What are the foot-prints of all animals?

```

(INDIVIDUAL-FILLERS DEER HAS-FOOT-PRINT) --> (DEER-FOOT-PRINT)
(INDIVIDUAL-FILLERS BEAR HAS-FOOT-PRINT) --> (BEAR-FOOT-PRINT)
(INDIVIDUAL-FILLERS RABBIT HAS-FOOT-PRINT) --> (RABBIT-FOOT-PRINT)

```

9. What is the required caliber given a foot-print?

```

(((RELATED CAL365MM BEAR-FOOT-PRINT CALIBER-FOR-FOOT-PRINT))
 ((RELATED CAL66MM DEER-FOOT-PRINT CALIBER-FOR-FOOT-PRINT))
 ((RELATED CAL12MM RABBIT-FOOT-PRINT CALIBER-FOR-FOOT-PRINT)))

```

10. Which is the required weapon for a certain terrain?

```

(((RELATED SNIPER MOUNTAIN WEAPON-FOR-TERRAIN))
 ((RELATED SHOTGUN FOREST WEAPON-FOR-TERRAIN))
 ((RELATED PISTOL PLAIN WEAPON-FOR-TERRAIN)))

```


1.8 Ontology evaluation

You can evaluate your ontology against several metrics.

Table 1.1: Ontology evaluation metrics in Racer

Number of concepts	(evaluate(length (all-atomic-concepts)))	20
Number of roles	(evaluate(length (all-roles)))	28
Number of individuals	(evaluate(length (all-individuals)))	19
Number of rules	(evaluate(length (all-rules)))	12
DL expressivity	(get-tbox-language)(get-abox-language)	SHf

After we checked the ontology's consistency and coherence, the results are:

(TBOX-COHERENT?) --> T

(TBOX-CYCLIC?) --> NIL

Chapter 2

DL-Learner

2.1 Configuration and execution

We converted our hunting.racer file into an owl file called hunt.owl and it looks like in the next photo.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns="http://example.com/father#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://example.com/hunt">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="not-weapon"/>
  <owl:Class rdf:ID="weapon">
    <owl:equivalentClass>
      <owl:Class>
        <owl:complementOf rdf:resource="#not-weapon"/>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="hasCaliber"/>
  <weapon rdf:ID="sniper">
    <hasCaliber>
      <not-weapon rdf:ID="cal365mm">
      </not-weapon>
    </hasCaliber>
  </weapon>
  <weapon rdf:ID="pistol">
    <hasCaliber rdf:resource="#cal12mm"/>
  </weapon>
  <weapon rdf:ID="shotgun">
    <hasCaliber rdf:resource="#cal66mm"/>
  </weapon>
  <not-weapon rdf:ID="knife"/>
  <not-weapon rdf:ID="bow"/>
  <not-weapon rdf:ID="prastie"/>
</rdf:RDF>
```

Next, we downloaded the DLELearner 1.4.0 software from <https://dl-learner.org>. In the *examples* directory from DL-learner, we created a new file with the extension .conf and in this file we modified the prefix to match the .owl file generated in the previous step. To execute the .conf file, we ran the following command in the command prompt:
bin/cli.bat examples/hunt.conf

```

/**
 * Hunt Example
 *
 * possible solution:
 *   male AND EXISTS hasChild.TOP
 *   weapon and exists caliber.top
 * Copyright (C) 2007, Jens Lehmann
 */

// declare some prefixes to use as abbreviations
prefixes = [ ("ex","http://example.com/hunt#") ]

// knowledge source definition
ks.type = "OWL File"
ks.fileName = "hunt.owl"

// reasoner
reasoner.type = "closed world reasoner"
reasoner.sources = { ks }

// learning problem
lp.type = "posNegStandard"
lp.positiveExamples = { "ex:sniper", "ex:shotgun", "ex:pistol" }
lp.negativeExamples = { "ex:prastie", "ex:bow", "ex:knife" }

// create learning algorithm to run
alg.type = "celoe"
alg.maxExecutionTimeInSeconds = 1

alg.writeSearchTree = true

```

2.2 Results

The software learns from examples and it generates a set of axioms. For positive examples, we used types of weapons which can be used in hunting sniper, shotgun and pistol and for negative examples, we used sling, bow and knife. The obtained result axioms are as following:

```

solutions:
1: weapon (pred. acc.: 100.00%, F-measure: 100.00%)
2: not (not-weapon) (pred. acc.: 100.00%, F-measure: 100.00%)
3: weapon or weapon (pred. acc.: 100.00%, F-measure: 100.00%)
4: weapon or not-weapon (pred. acc.: 100.00%, F-measure: 100.00%)
5: weapon or (not (not-weapon)) (pred. acc.: 100.00%, F-measure: 100.00%)
6: not-weapon or (not (not-weapon)) (pred. acc.: 100.00%, F-measure: 100.00%)
7: weapon or (not (not-weapon)) (pred. acc.: 100.00%, F-measure: 100.00%)
8: weapon and (not (weapon)) (pred. acc.: 100.00%, F-measure: 100.00%)
9: weapon and (not (not-weapon)) (pred. acc.: 100.00%, F-measure: 100.00%)
10: weapon or (hasCaliber some Thing) (pred. acc.: 100.00%, F-measure: 100.00%)

```

Chapter 3

Java API

3.1 Configuration and execution

In order to populate the ontology with instances, we also used the JRacer API. This connects to the Racer server's ip address and to the port which can read the hunt.racer file to access the ontology.

```
String ip = "127.0.0.1";
int port = 8088;
// String filename="/Applications/RacerPro 2.0 preview/examples/owl/people-pets.owl";
String filename="\\C:/Users/trian/Desktop/Sbc/test.racer\\";

RacerClient racer = new RacerClient(ip,port);
try {
    racer.openConnection();
```

After the connection setup has been made we can create new instances and send them to our ontology. In the next image we create instances for our most important concept and using the method *sendRaw* we send them to the ontology.

```
racer.sendRaw("(racer-read-file " + filename + ")");

String name="wolf";
racer.sendRaw("(instance " + name+ " animal)");

name="moose";
racer.sendRaw("(instance " + name+ " animal)");
name="elk";
racer.sendRaw("(instance " + name+ " animal)");

name="desert_eagle";
racer.sendRaw("(instance " + name+ " weapon)");
name="bow";
racer.sendRaw("(instance " + name+ " weapon)");
name="ak47";
racer.sendRaw("(instance " + name+ " weapon)");
name="caliber150mm";
racer.sendRaw("(instance " + name+ " caliber)");
name="caliber88mm";
racer.sendRaw("(instance " + name+ " caliber)");
name="caliber35mm";
racer.sendRaw("(instance " + name+ " caliber)");
```

After all our instances are sent to the ontology we can now execute a query to show that all are according to plan. The comment part of our code is used when we want to modify our

racer file, but in testing this is not desired because it can rip-off our file.

```
System.out.println(racer.sendRaw("(all-atomic-concepts)+"+"\n"));
System.out.println(racer.sendRaw("(concept-instances animal)+"+"\n"));
System.out.println(racer.sendRaw("(concept-instances caliber)+"+"\n"));
System.out.println(racer.sendRaw("(concept-instances weapon)+"+"\n"));
System.out.println(racer.sendRaw("(concept-instances terrain)+"+"\n"));
System.out.println(racer.sendRaw("(individual-fillers sniper has-caliber) "+"+"\n"));
System.out.println(racer.sendRaw("(concept-instances foot-print)+"+"\n"));
System.out.println(racer.sendRaw("(concept-instances Country)+"+"\n"));

/*
FileWriter fileWriter = new FileWriter("C:/Users/trian/Desktop/Sbc/test.racer", true); //Set true for append mode
racer.out = new PrintWriter(fileWriter);
racer.out.println("\n"+data);
racer.out.close();
*/
racer.closeConnection();
}
catch (Exception e) {
    e.printStackTrace();
}
```

3.2 Results


Here are all the results from the previous query. First are all concepts and after that are all instances of every important concept. As you can see at the sixth line, we can make a query to find the caliber of a specific weapon.

```
(TOP BOTTOM HAS-AGE HAS-NAME WEAPON TERRAIN ANIMAL MINIFISH CALIBER BAIT STORE HONY HUNTER GRASS FOOD COUNTRY FOOT-PRINT HAS-ADDRESS)
(ELK MOOSE WOLF BEAR DEER RABBIT)
(CALIBER35MM CALIBER88MM CALIBER150MM CAL365MM CAL66MM CAL12MM)
(AK47 BOW DESERT_EAGLE SNIPER SHOTGUN PISTOL)
(MOUNTAIN FOREST PLAIN)
(CAL365MM)
(BEAR-FOOT-PRINT DEER-FOOT-PRINT RABBIT-FOOT-PRINT)
(RUSIA SUEIDIA ROMANIA)
```

Chapter 4

Fuzzy Description Logic

We define four concepts which refer to a gun loader. *LowAmmo*, *AllMostFullAmmo*, *ammo10*, *LessThan15Bullets* and *incarcator* are our concepts. They are defined in the image below. Our goal was to determine if a gun loader is in one of the the 3 states, and for a loader with 12 bullets and being part of *LessThan15bullets* it is empty (0.2).

 hunt - Notepad

```
File Edit Format View Help
(define-fuzzy-concept LessThan15Bullets left-shoulder(0,30,0,15))

(define-fuzzy-concept LowAmmo crisp(0,30,0,5))

(define-fuzzy-concept AlMostFullAmmo crisp(0,30,25,30))

(define-fuzzy-concept ammo10 left-shoulder(0,30,12,12))

(functional ammo)

(instance ak47 (and Weapon (some ammo ammo10)))

(implies sniper Weapon)

(define-concept incarcator (and Weapon (some ammo LessThan15Bullets)))

#determin cat la suta din incarcator e gol
(min-instance? ak47 incarcator)
```

For testing our fuzzy logic we open a cmd and in there we run our file, like in the image below.

```
C:\Users\trian\Desktop\Sbc\FuzzyDLWindows\FuzzyDL>java -jar FuzzyDL.jar hunt.txt
Is ak47 instance of incarcator ? >= 0.2
```

Chapter 5

Ontology Design Patterns

5.1 Partition Pattern

Consider whether some set of subconcepts fully covers a concept.

```
;;;;;;Partition pattern;;;;;;  
(define-concept Food (or grass hony))  
  
(disjoint grass hony)  
  
;(concept-disjoint? iarba miere)  
;(concept-parents iarba)
```

5.2 N-ary relations

The aim is to model n-ary relations in an ontology, given that DL was designed to express binary relations.

```
;;;;;;N-ary relation pattern;;;;;;  
  
( implies Hunter ( and ( some propety-1 has-name)  
                      ( some propety-2 has-age)  
                      ( some propety-3 has-address)))  
  
(define-concrete-domain-attribute has-name :domain Hunter :type string)  
(define-concrete-domain-attribute has-age :domain Hunter :type integer)  
(define-primitive-role has-address :domain Hunter :range Country )  
  
(instance Trif Hunter)  
(attribute-filler Trif "Andrei" has-name )  
(attribute-filler Trif 23 has-age)  
(related Trif Romania has-address)  
  
;(concept-instances Hunter)  
;(individual-fillers Trif has-address)  
;(individual-told-attribute-value Trif has-age)
```

5.3 PartOf

This ODP aims to represent entities and their parts. The pattern exploits reasoning on transitive role has-part.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;PartOf pattern;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  
( define-primitive-role has-part :transitive t :inverse is-part-of :domain Food)  
( implies Store ( some has-part Bait ))  
( implies Minifish ( some is-part-of Bait ))  
( implies Minifish Food )  
;(concept-subsumes? ( some is-part-of Bait ) Minifish )
```


Chapter 6

Querying your ontology

6.1 Racer

```
(concept-instances caliber)
(concept-instances weapon)
(concept-instances animal)
(concept-instances foot-print)

;comanda pentru urma unui animal specific
(individual-fillers deer has-foot-print)
(individual-fillers bear has-foot-print)
(individual-fillers rabbit has-foot-print)

;determina animalul dupa calibru si invers
(individual-fillers cal365mm caliber-for-animal)
(individual-fillers bear (inv caliber-for-animal))

;determina vanatori
(concept-instances Hunter)
;determina adresa vanatorului trif
(individual-fillers Trif has-address)
;determina varsta vanatorului
(individual-told-attribute-value Trif has-age)

;verifica daca mancare este diferita
(concept-disjoint? iarba miere)
(concept-parents iarba)

;how to see sniper's caliber
(individual-fillers sniper has-caliber)
(individual-fillers pistol has-caliber)
(individual-fillers shotgun has-caliber)

;how to see instances of caliber
;(concept-instances caliber)
;(concept-instances animal)
;(concept-instances weapon)

(individual-instance? Trif Hunter)
```

6.2 NRQL

```
;;;;;;;;;;;;;NRQL;;;;;;;;;;;;;
; CQ18: Which are all relations that were connected to weapon ?
(retrieve (?x) (?x weapon))

; CQ4 : Care este animalului x dupa urma sa?
(retrieve (?x)
  (and (?x animal)
    (deer-foot-print foot-print)
    (?x deer-foot-print has-foot-print)))

(retrieve (?x)
  (and (?x animal)
    (bear-foot-print foot-print)
    (?x bear-foot-print has-foot-print)))

(retrieve (?x)
  (and (?x animal)
    (rabbit-foot-print foot-print)
    (?x rabbit-foot-print has-foot-print)))

;Determinam toti vanatorii din Romania
(retrieve (?x )
  (and (?x Hunter)
    (Romania Country)
    (?x Romania has-address)))
```

Appendix A

Racer code

```
/**
 * Hunt Example
 *
 * possible solution:
 *   male AND EXISTS hasChild.TOP
 *   weapon and exists caliber.top
 * Copyright (C) 2007, Jens Lehmann
 */

// declare some prefixes to use as abbreviations
prefixes = [ ("ex","http://example.com/hunt#") ]

// knowledge source definition
ks.type = "OWL File"
ks.fileName = "hunt.owl"

// reasoner
reasoner.type = "closed world reasoner"
reasoner.sources = { ks }

// learning problem
lp.type = "posNegStandard"
lp.positiveExamples = { "ex:sniper", "ex:shotgun", "ex:pistol" }
lp.negativeExamples = { "ex:prastie", "ex:bow", "ex:knife" }

// create learning algorithm to run
alg.type = "celoe"
alg.maxExecutionTimeInSeconds = 1

alg.writeSearchTree = true
```

Appendix B

DL-learner code

```
    hunt.conf

/**
 * Hunt Example
 *
 * possible solution:
 *   male AND EXISTS hasChild.TOP
 *   weapon and exists caliber.top
 * Copyright (C) 2007, Jens Lehmann
 */

// declare some prefixes to use as abbreviations
prefixes = [ ("ex","http://example.com/hunt#") ]

// knowledge source definition
ks.type = "OWL File"
ks.fileName = "hunt.owl"

// reasoner
reasoner.type = "closed world reasoner"
reasoner.sources = { ks }

// learning problem
lp.type = "posNegStandard"
lp.positiveExamples = { "ex:sniper", "ex:shotgun", "ex:pistol" }
lp.negativeExamples = { "ex:prastie", "ex:bow", "ex:knife" }

// create learning algorithm to run
alg.type = "celoe"
alg.maxExecutionTimeInSeconds = 1

alg.writeSearchTree = true
```

hunt.owl

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns="http://example.com/father#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://example.com/hunt">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="not-weapon"/>
  <owl:Class rdf:ID="weapon">
    <owl:equivalentClass>
      <owl:Class>
        <owl:complementOf rdf:resource="#not-weapon"/>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="hasCaliber"/>
  <weapon rdf:ID="sniper">
    <hasCaliber>
      <not-weapon rdf:ID="cal365mm">
        </not-weapon>
      </hasCaliber>
    </weapon>
  <weapon rdf:ID="pistol">
    <hasCaliber rdf:resource="#cal12mm"/>
  </weapon>
  <weapon rdf:ID="shotgun">
    <hasCaliber rdf:resource="#cal66mm"/>
  </weapon>
  <not-weapon rdf:ID="knife"/>
  <not-weapon rdf:ID="bow"/>
  <not-weapon rdf:ID="prastie"/>
</rdf:RDF>
```

Appendix C

Fuzzy code

```
(define-fuzzy-concept LessThan15Bullets left-shoulder(0,30,0,15))

(define-fuzzy-concept LowAmmo crisp(0,30,0,5))

(define-fuzzy-concept AlmostFullAmmo crisp(0,30,25,30))

(define-fuzzy-concept ammo10 left-shoulder(0,30,12,12))

(functional ammo)

(instance ak47 (and Weapon (some ammo ammo10)))

(implies sniper Weapon)

(define-concept incarcator (and Weapon (some ammo LessThan15Bullets)))

#determin cat la suta din incarcator e gol
(min-instance? ak47 incarcator)
```

Appendix D

Java API code

```
package sbc12;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.PrintWriter;

import com.racersystems.jracer.RacerClient;

public class Main1 {

    public static void main(String[] argv) {
        String ip = "127.0.0.1";
        int port = 8088;
        // String filename="\Applications/RacerPro 2.0 preview/examples/owl/people-pets.owl\"";
        String filename="\C:/Users/trian/Desktop/Sbc/test.racer\"";

        RacerClient racer = new RacerClient(ip, port);
        try {
            racer.openConnection();
            racer.sendRaw("(racer-read-file " + filename + ")");

            String name="wolf";
            racer.sendRaw("(instance "+ name+ " animal)");

            name="moose";
            racer.sendRaw("(instance "+ name+ " animal)");
            name="elk";
            racer.sendRaw("(instance "+ name+ " animal)");

            name="desert_eagle";
            racer.sendRaw("(instance "+ name+ " weapon)");
            name="bow";
            racer.sendRaw("(instance "+ name+ " weapon)");
            name="ak47";
            racer.sendRaw("(instance "+ name+ " weapon)");
            name="caliber150mm";
            racer.sendRaw("(instance "+ name+ " caliber)");
            name="caliber88mm";
            racer.sendRaw("(instance "+ name+ " caliber)");
            name="caliber35mm";
            racer.sendRaw("(instance "+ name+ " caliber)");

            System.out.println(racer.sendRaw("(all-atomic-concepts)+"\n"));
            System.out.println(racer.sendRaw("(concept-instances animal)+"\n"));
            System.out.println(racer.sendRaw("(concept-instances caliber)+"\n"));
            System.out.println(racer.sendRaw("(concept-instances weapon)+"\n"));
            System.out.println(racer.sendRaw("(concept-instances terrain)+"\n"));
            System.out.println(racer.sendRaw("(individual-fillers sniper has-caliber) "+"'\n"));
            System.out.println(racer.sendRaw("(concept-instances foot-print)+"\n"));
            System.out.println(racer.sendRaw("(concept-instances Country)+"\n"));

            /*
            FileWriter fileWriter = new FileWriter("C:/Users/trian/Desktop/Sbc/test.racer", true); //Set true
            racer.out = new PrintWriter(fileWriter);
```

```
        racer.out.println("\n"+data);
        racer.out.close();
    */
        racer.closeConnection();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}
```


Bibliography

Intelligent Systems Group

