Seminar P2P Netzwerke, summer term 2009
Computer Networks and Telematics
University of Freiburg

RDFS Reasoning and Query Answering on Top of DHTs

Zoi Kaoudi, Iris Miliaraki, Manolis Koubarakis

July 2009

Presented by:
Christine Haas

Outline

1. Introduction
   1.1 Keywords
   1.2 Storing protocol
   1.3 RDFS Reasoning
2. Reasoning and Querying in distributed environments
   2.1 Forward Chaining (FC)
   2.2 Backward Chaining (BC)
3. Comparison FC and BC
4. Experimental Evaluation

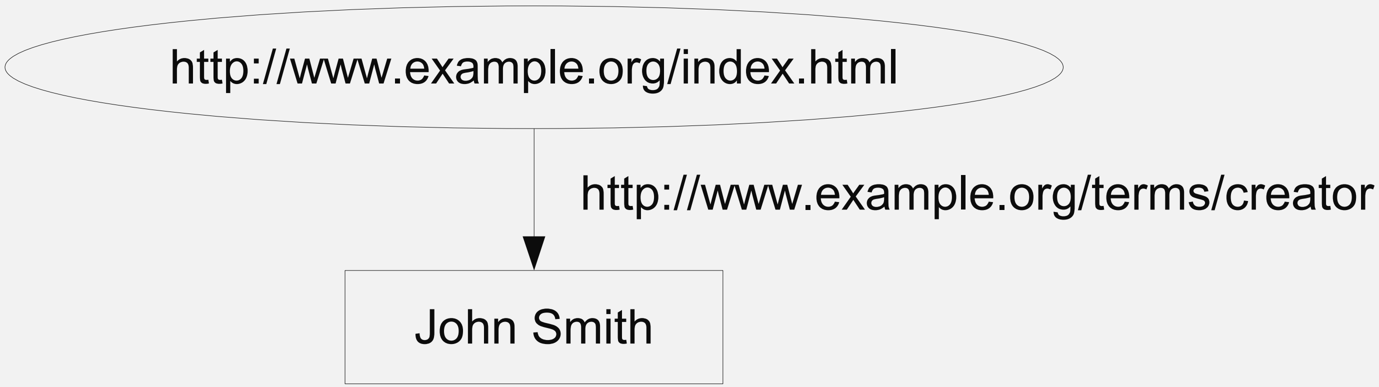
RDF

- **R**esource **D**escription **F**ramework
- XML-based metadata language
- intended for representing metadata about Web resources
- RDF statements: Subject – Predicate – Object

RDF

- **Resource** **Description** **Framework**
- XML-based metadata language
- intended for representing metadata about Web resources
- RDF statements: Subject – Predicate – Object

Directed labelled graph:



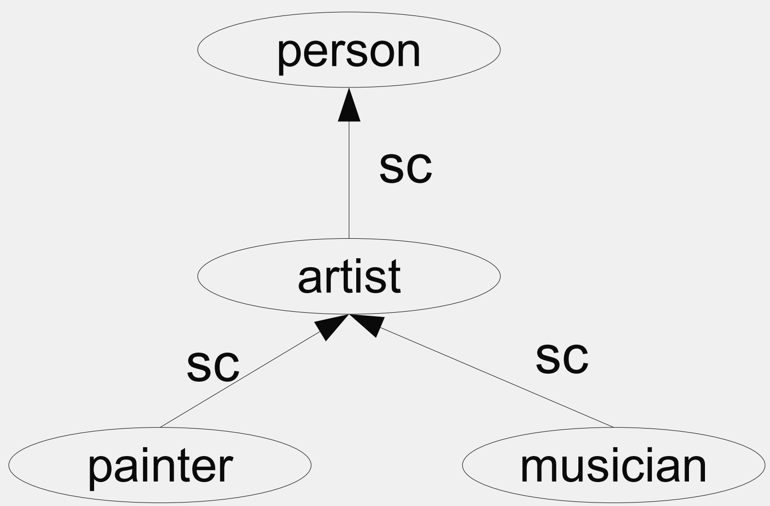
Triple notation (S, P, O):
(http://www.example.org/index.html,http://www.example.org/terms/creator,John Smith)

RDFS

- **R**esource **D**escription **F**ramework **S**chema
- extension of RDF
- provides the means needed to describe classes and properties (type system for RDF)
- specify domain vocabulary and object structures

RDFS

- **R**esource **D**escription **F**ramework **S**chema
- extension of RDF
- provides the means needed to describe classes and properties (type system for RDF)
- specify domain vocabulary and object structures



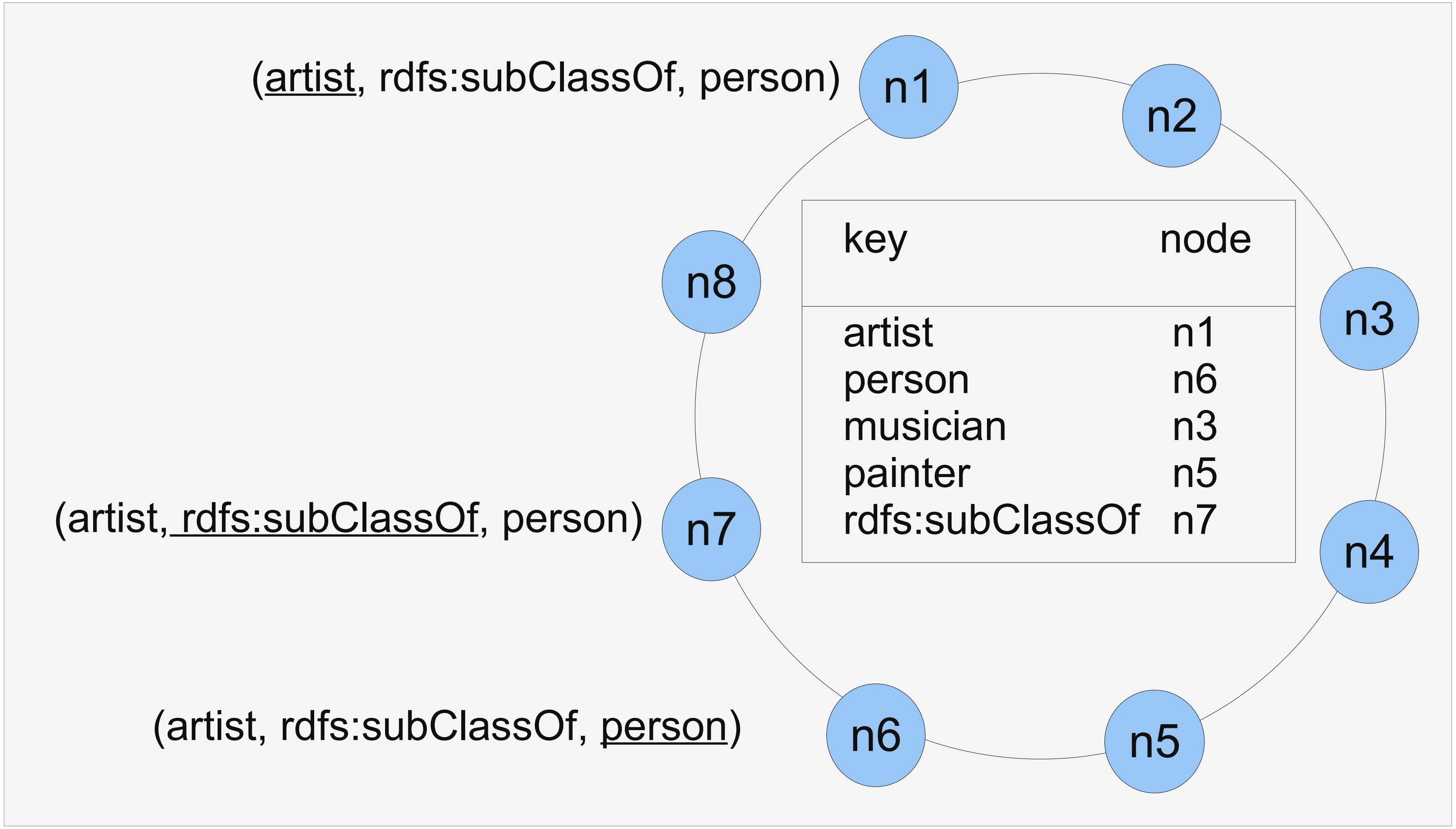
(artist, rdfs:subClassOf, person)

(painter, rdfs:subClassOf, artist)

(musician, rdfs:subClassOf, artist)

Storing protocol

- each triple is indexed in the DHT three times



RDFS Reasoning

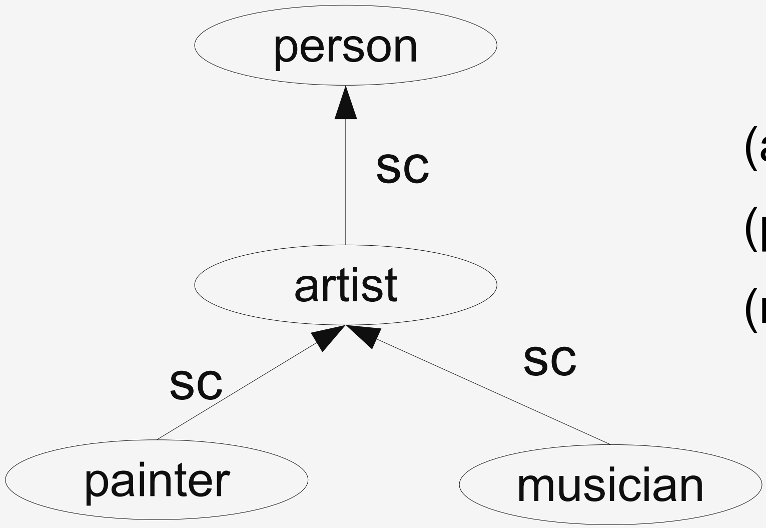
- infer new knowledge from static information
- RDFS entailment rules
- Rules for computing
  - all possible instances of a class
  - the transitive closure of a RDFS property hierarchy
  - the transitive closure of a RDFS class hierarchy

RDFS entailment rules

Rule	Head	Body
1	type(X,Y)	triple(X,rdf:type,Y)
2	type(X,Y)	triple(X,P,Z),triple(P,rdfs:domain,Y)
3	type(X,Y)	triple(Z,P,X), triple(P,rdfs:range,Y)
4	type(X,Y)	type(X,Z), subclass(Z,Y)
5	subProperty(X,Y)	triple(X,rdfs:subPropertyOf,Y)
6	subProperty(X,Y)	triple(X,rdfs:subPropertyOf,Z), subProperty(Z,Y)
7	subClass(X,Y)	triple(X,rdfs:subClassOf,Y)
8	subClass(X,Y)	triple(X,rdfs:subClassOf,Z), subClass(Z,Y)

RDFS Reasoning

- the transitive closure of an RDFS class hierarchy
  - subClass(X,Y) <- triple(X,rdfs:subClassOf,Y)
  - subClass(X,Y) <- triple(X,rdfs:subClassOf,Z),
                                subClass(Z,Y)



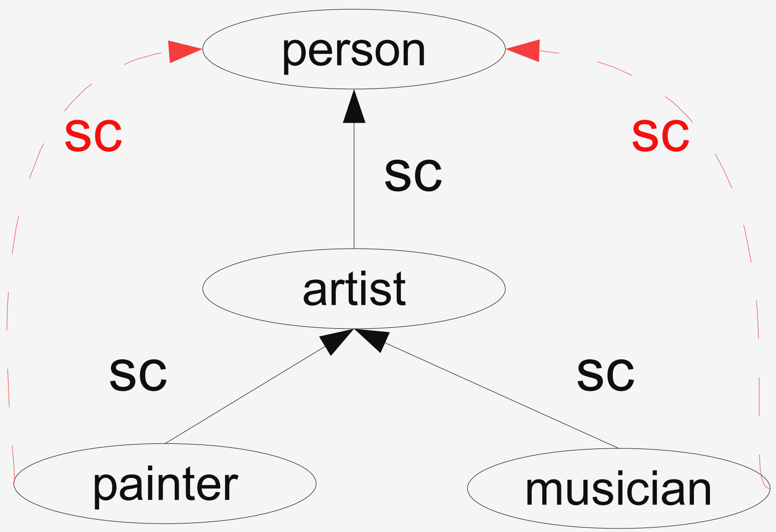
(artist, rdfs:subClassOf, person)

(painter, rdfs:subClassOf, artist)

(musician, rdfs:subClassOf, artist)

RDFS Reasoning

- the transitive closure of an RDFS class hierarchy
  - subClass(X,Y) <- triple(X,rdfs:subClassOf,Y)
  - subClass(X,Y) <- triple(X,rdfs:subClassOf,Z),
    subClass(Z,Y)



(artist, rdfs:subClassOf, person)
(painter, rdfs:subClassOf, artist)
(musician, rdfs:subClassOf, artist)
(painter, rdfs:subClassOf, person)
(musician, rdfs:subClassOf,person)

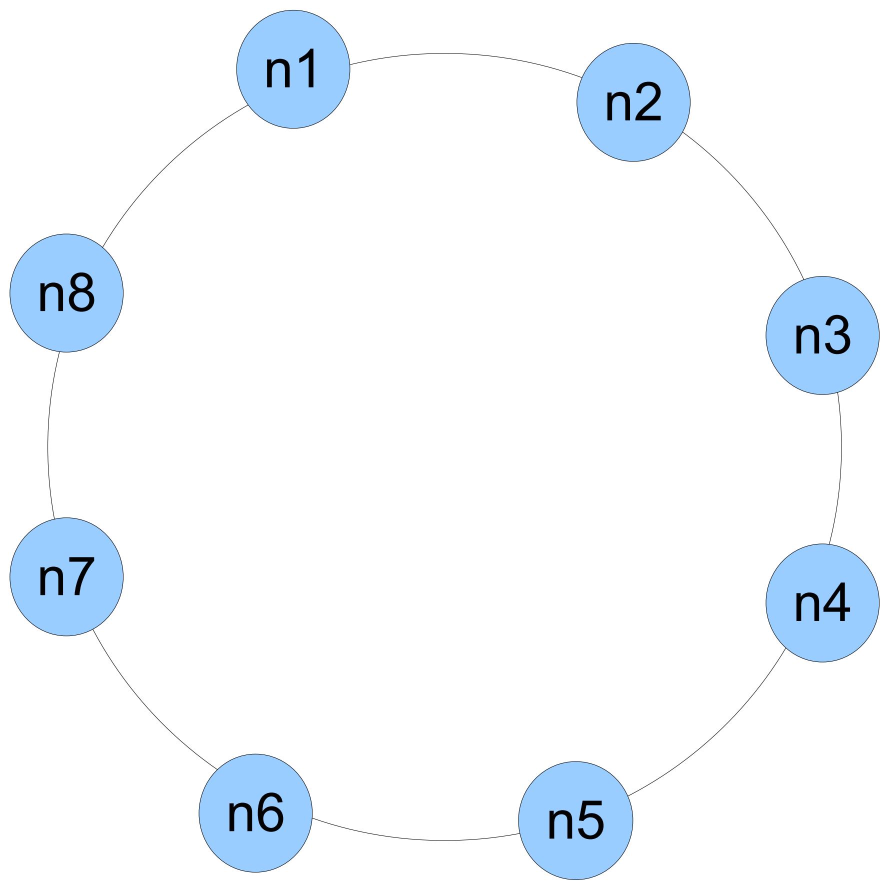
Problem:

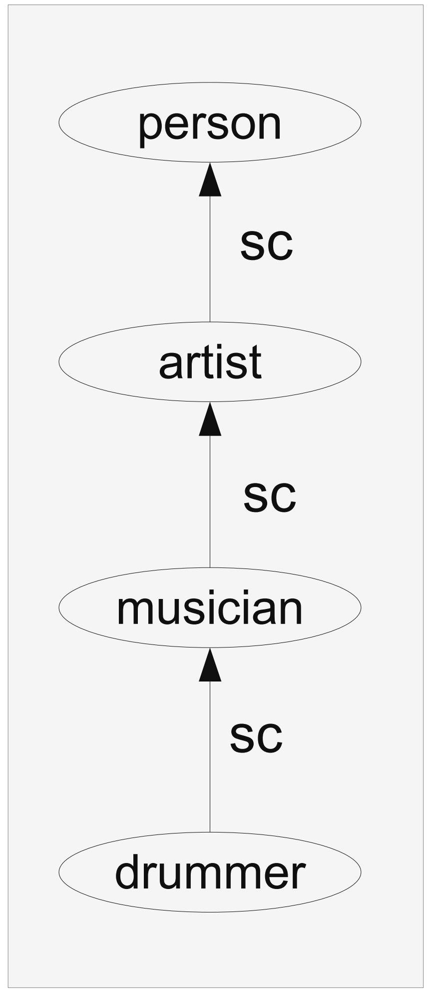
- RDFS Reasoning and Query answering in distributed environments

Solution:

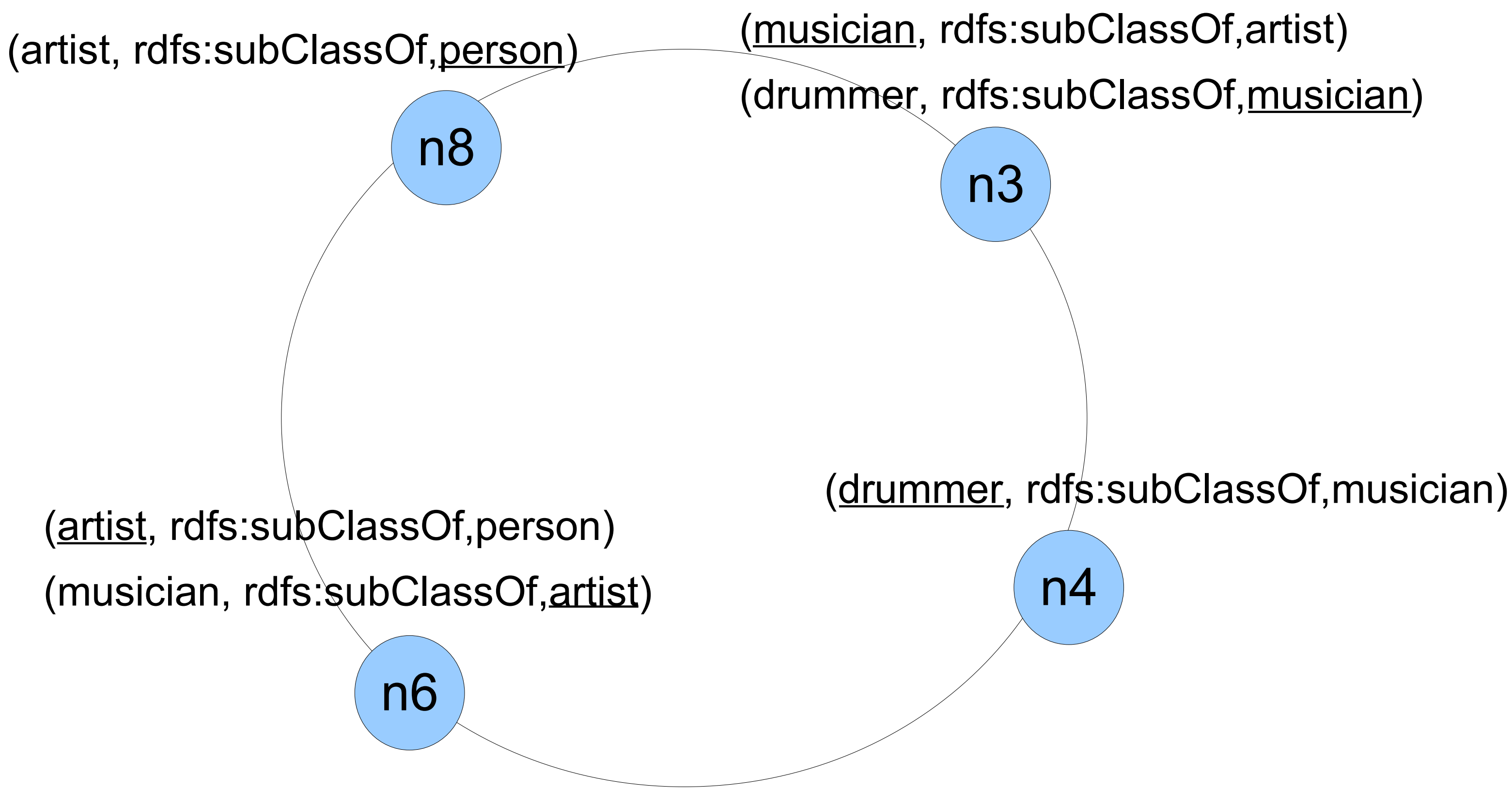
- ***Forward Chaining Algorithm***
  - data driven approach
  - all inferred triples are precomputed and stored in the network
- ***Backward Chaining Algorithm***
  - only evaluates RDFS entailment rules at query processing time

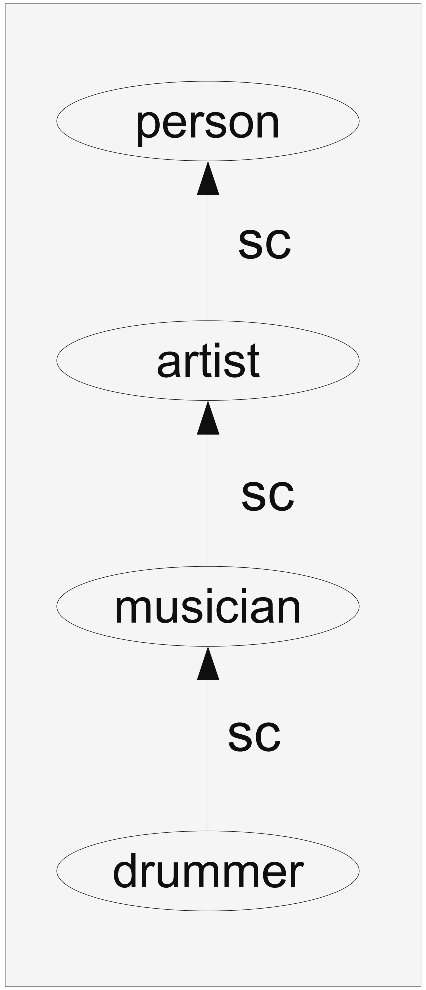
Forward Chaining Algorithm



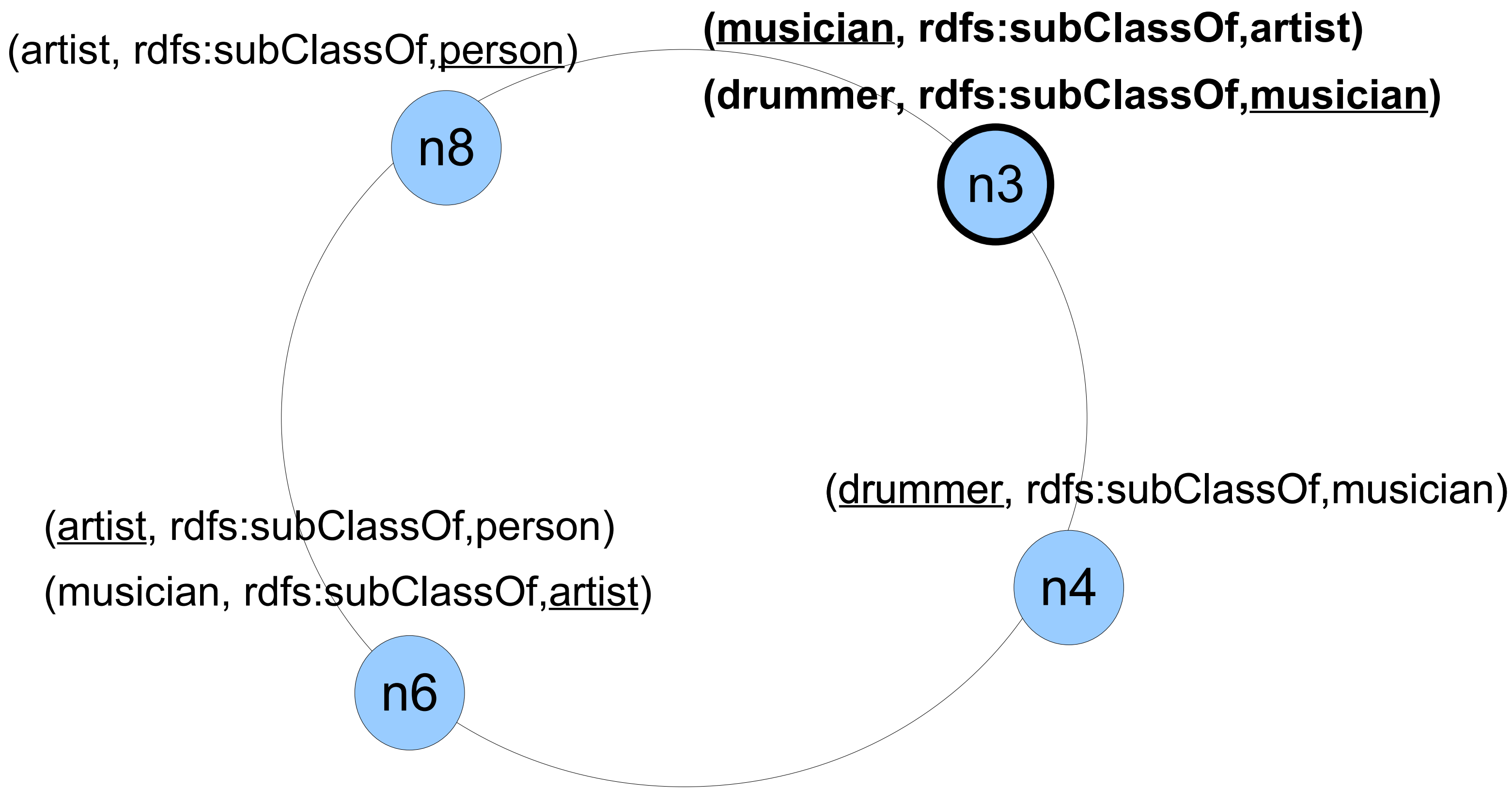


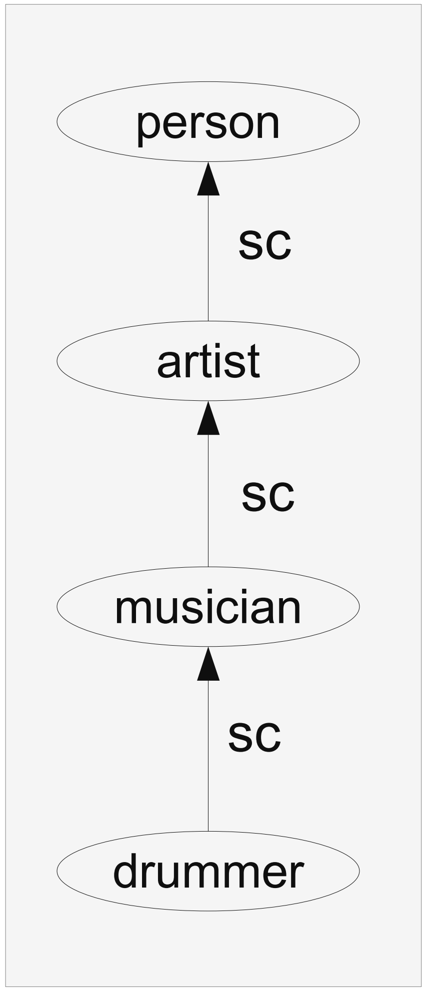
Forward Chaining Algorithm



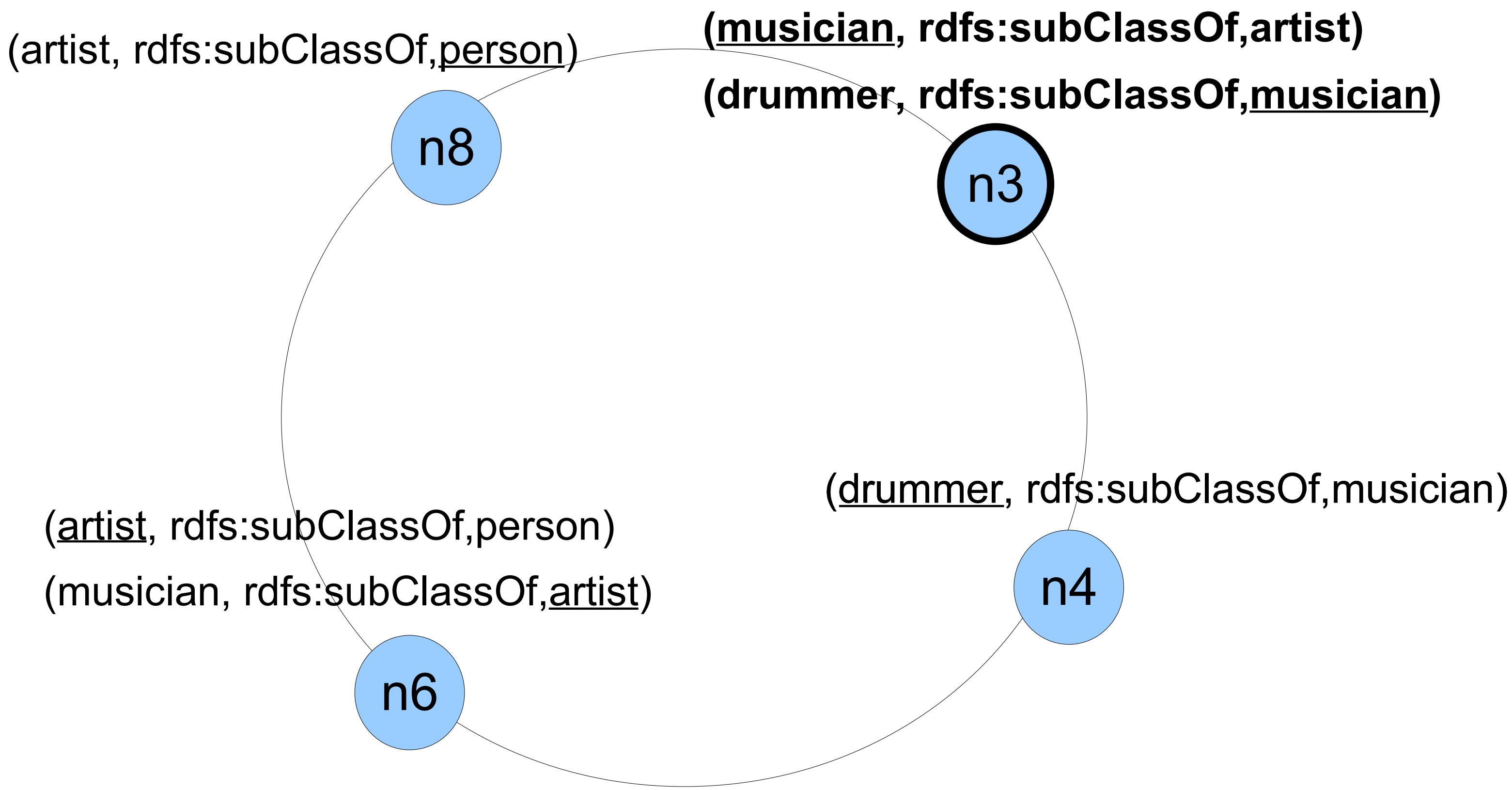


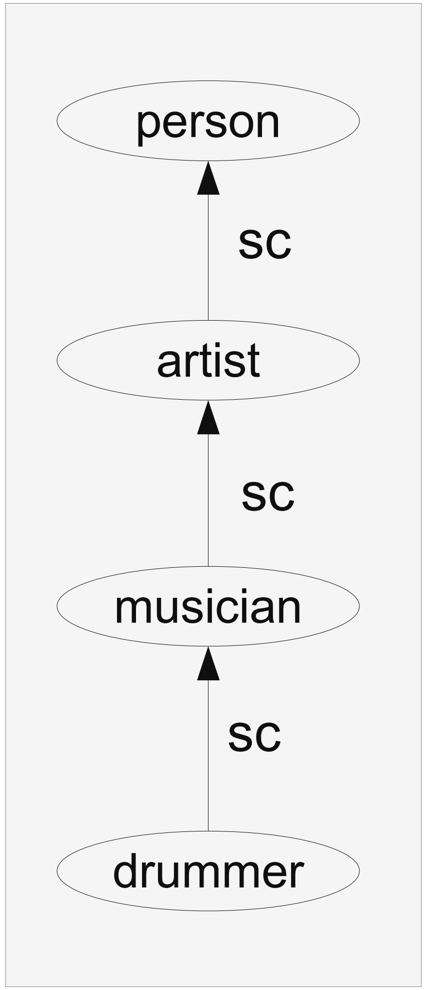
Forward Chaining Algorithm





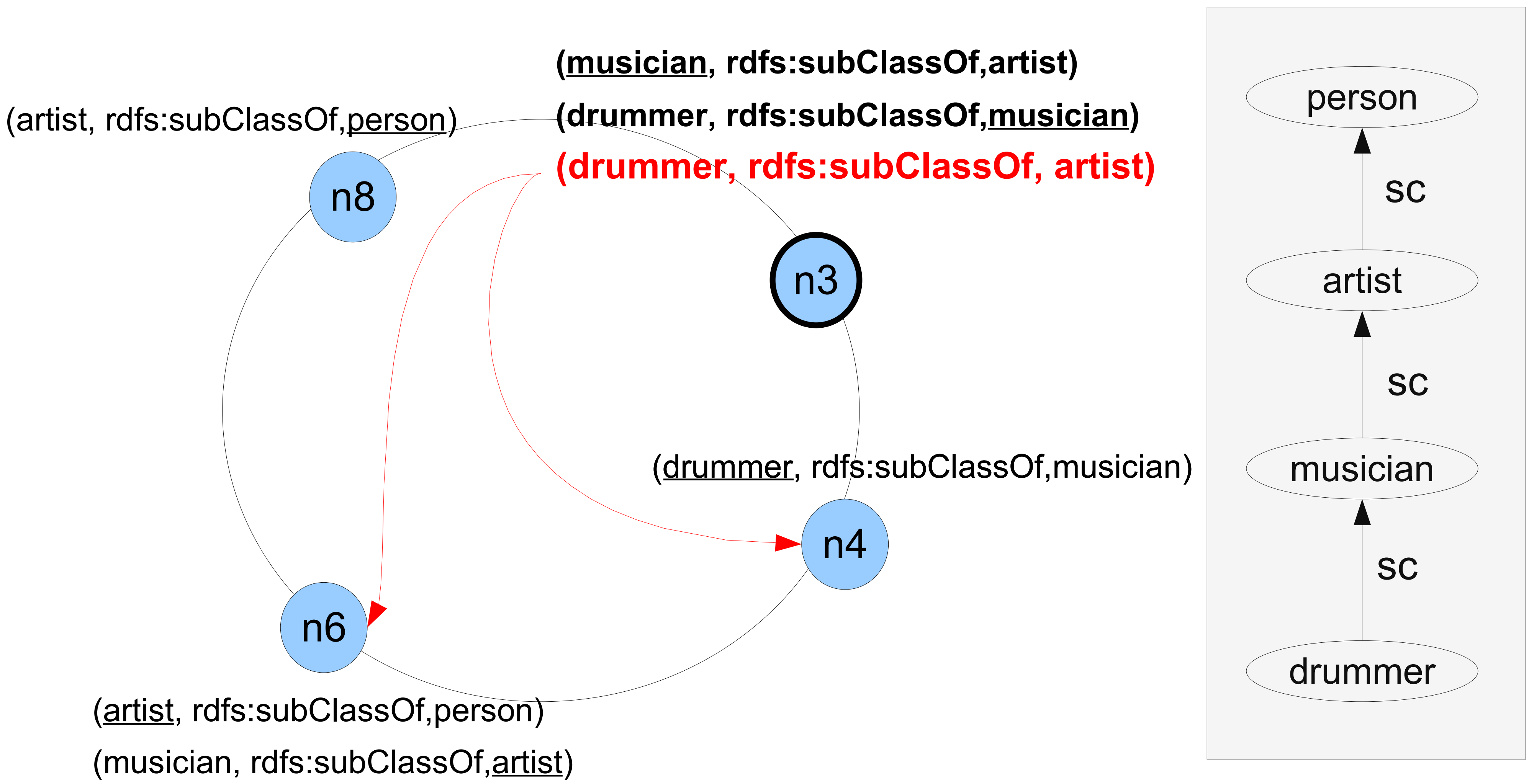
Forward Chaining Algorithm



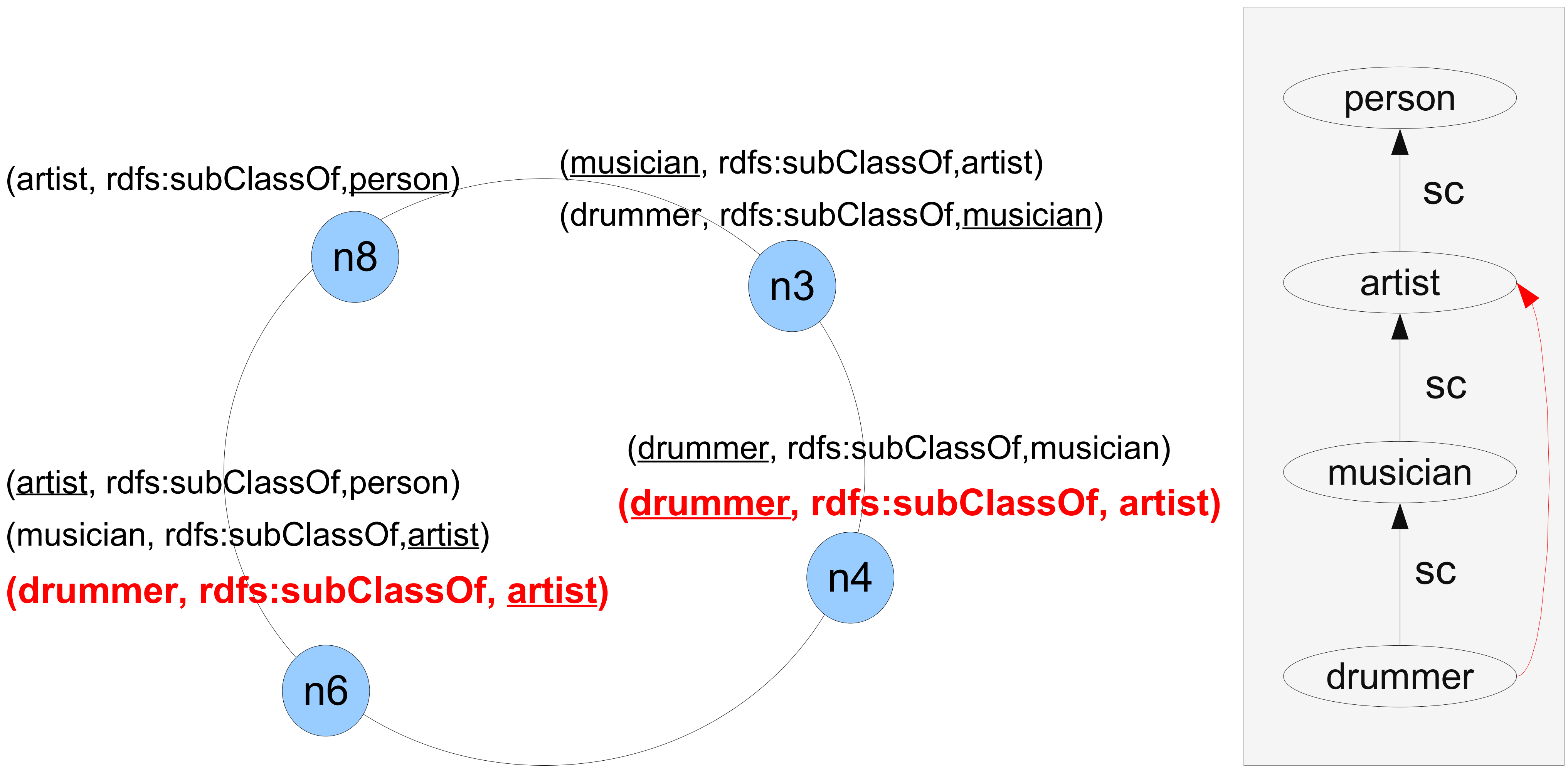


subClass(X,Y) :- triple(X,rdfs:subClassOf,Y)
subClass(X,Y) :- triple(X,rdfs:subClassOf,Z),subClass(Z,Y)

Forward Chaining Algorithm



Forward Chaining Algorithm



Forward Chaining Algorithm

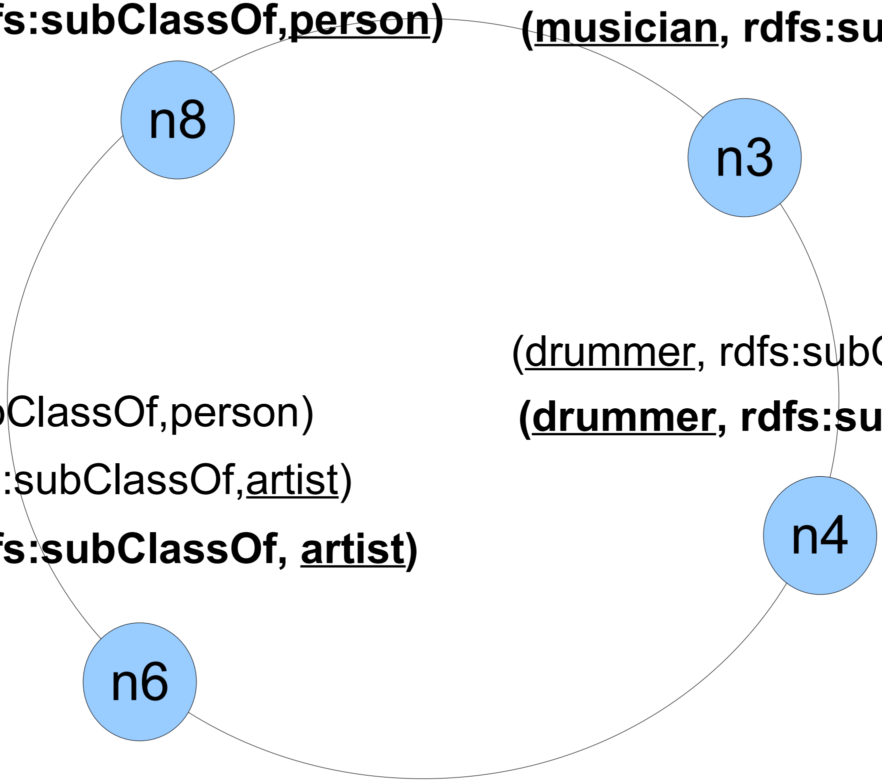
(artist, rdfs:subClassOf,person)

(musician, rdfs:subClassOf,person)

(musician, rdfs:subClassOf,artist)

(drummer, rdfs:subClassOf,musician)

(musician, rdfs:subClassOf,person)



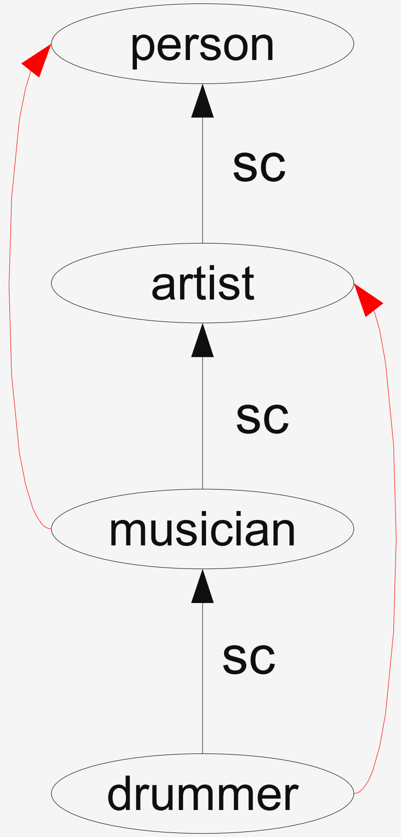
(drummer, rdfs:subClassOf,musician)

(drummer, rdfs:subClassOf, artist)

(artist, rdfs:subClassOf,person)

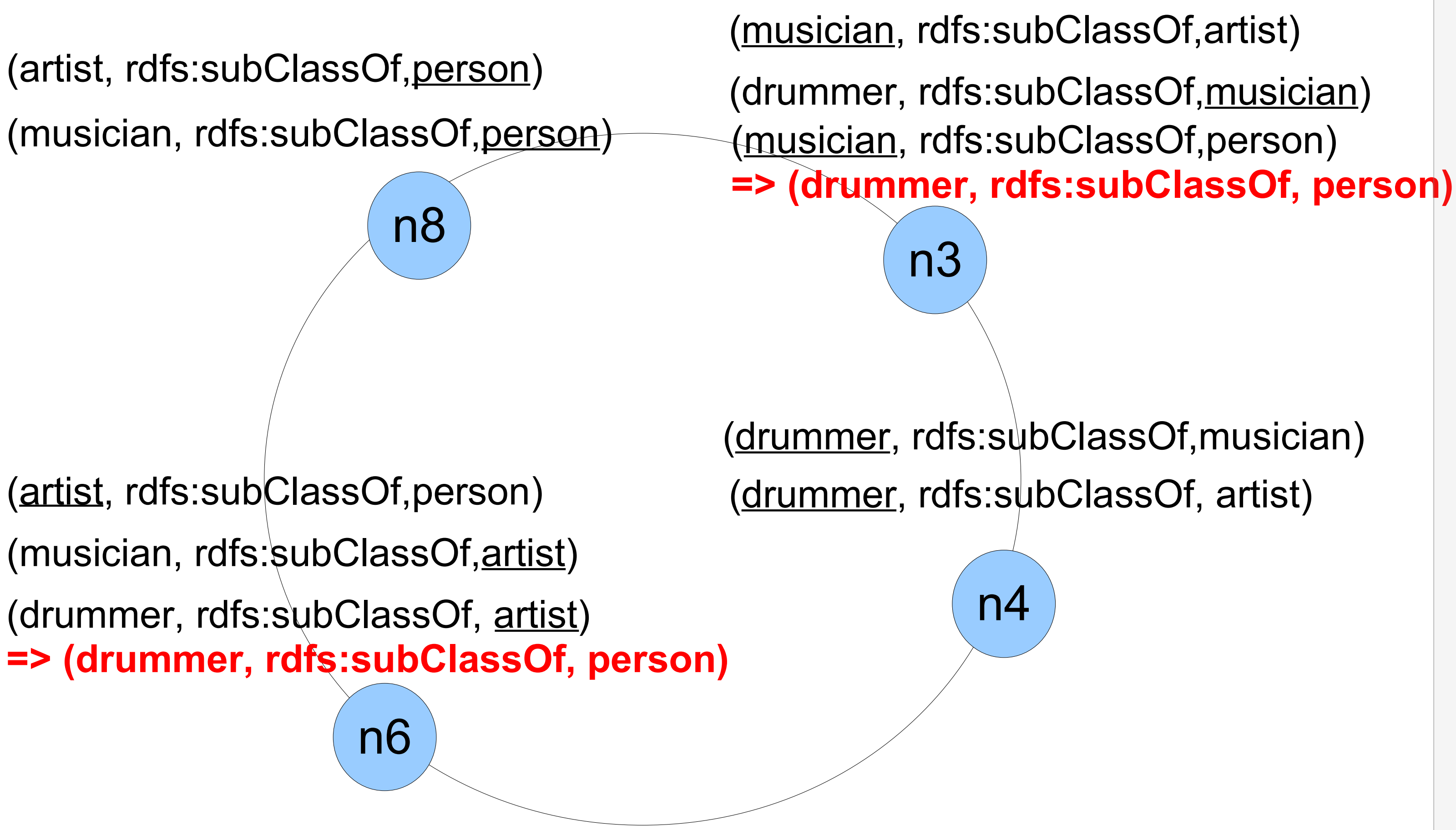
(musician, rdfs:subClassOf,artist)

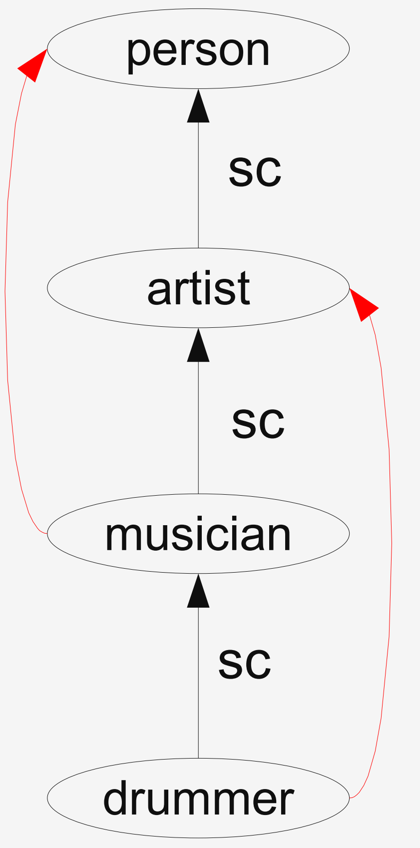
(drummer, rdfs:subClassOf, artist)



result after 1. iteration

Forward Chaining Algorithm





result after 1. iteration

Forward Chaining Algorithm

(artist, rdfs:subClassOf,person)

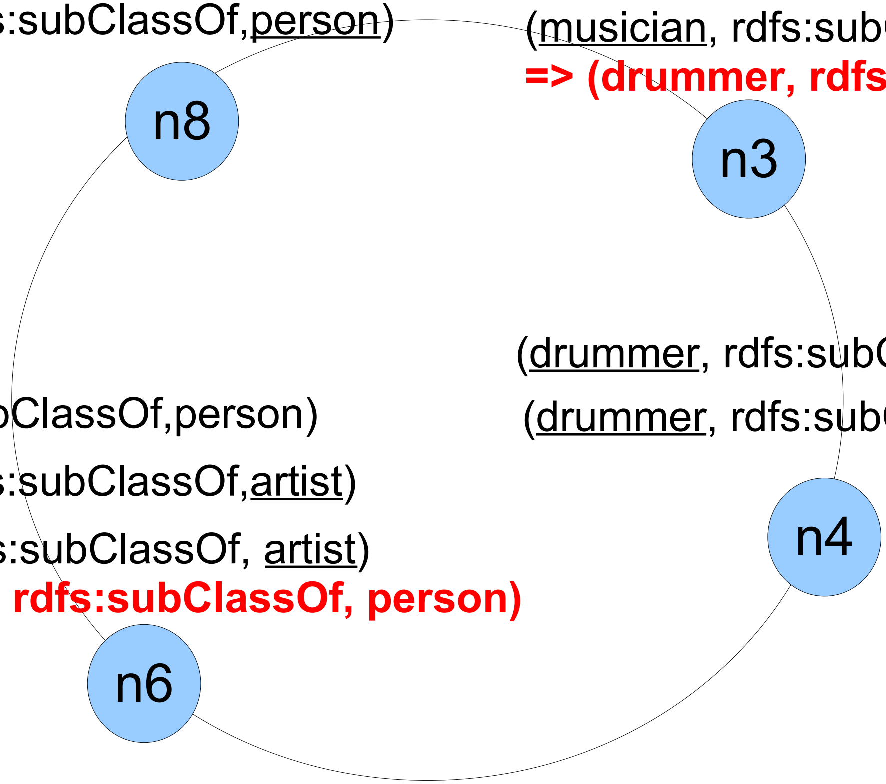
(musician, rdfs:subClassOf,person)

(musician, rdfs:subClassOf,artist)

(drummer, rdfs:subClassOf,musician)

(musician, rdfs:subClassOf,person)

=> (drummer, rdfs:subClassOf, person)



(drummer, rdfs:subClassOf,musician)

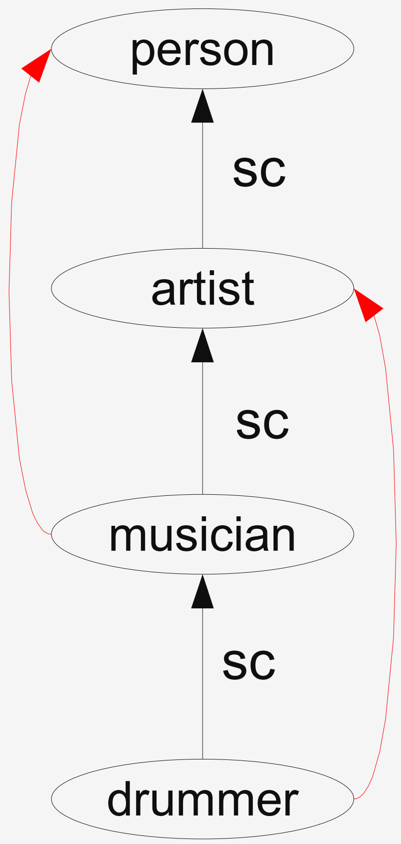
(drummer, rdfs:subClassOf, artist)

(artist, rdfs:subClassOf,person)

(musician, rdfs:subClassOf,artist)

(drummer, rdfs:subClassOf, artist)

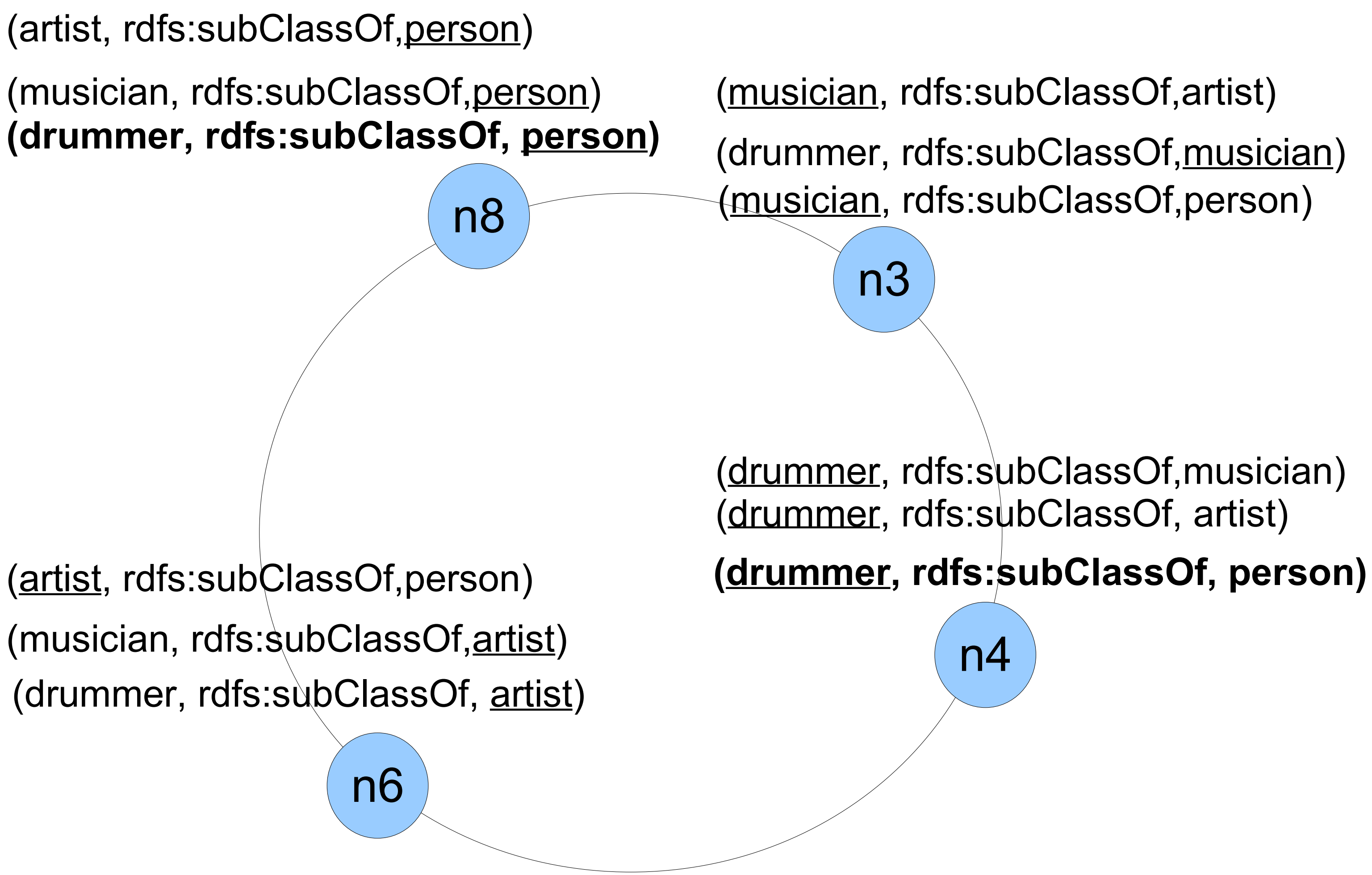
=> (drummer, rdfs:subClassOf, person)

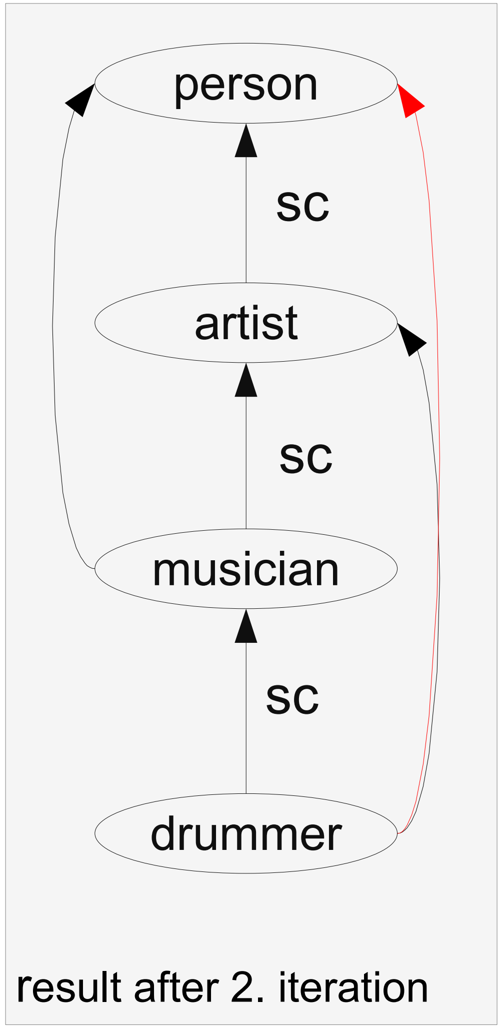


result after 1. iteration

n3 and n6 both infer tripel (drummer, rdfs:subClassOf,person)
=> triple is sent to the network to be stored twice

Forward Chaining Algorithm

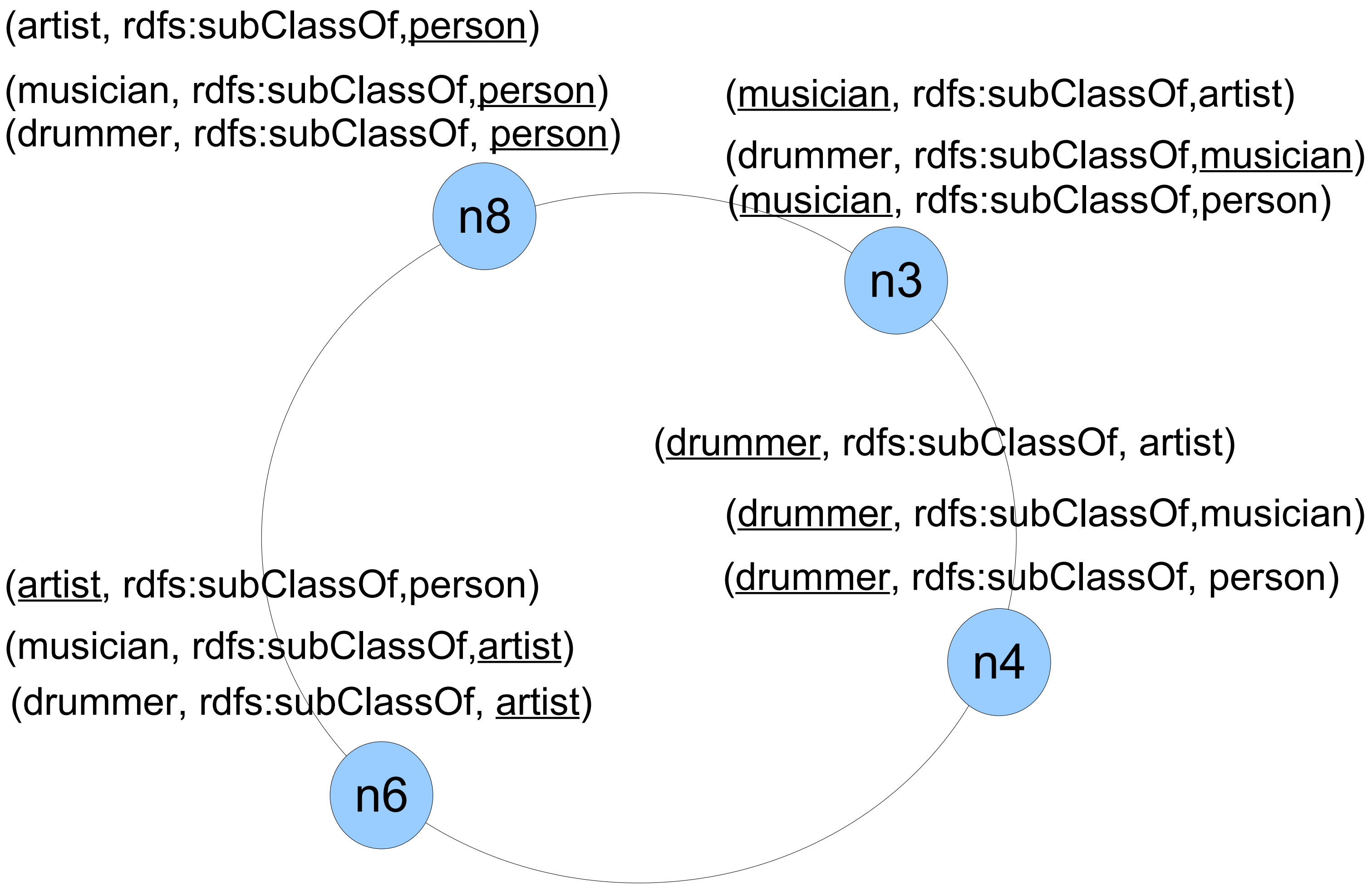




Forward Chaining Algorithm - Querying

Find all subclasses to artist:

(X, rdfs:subClassOf, artist)

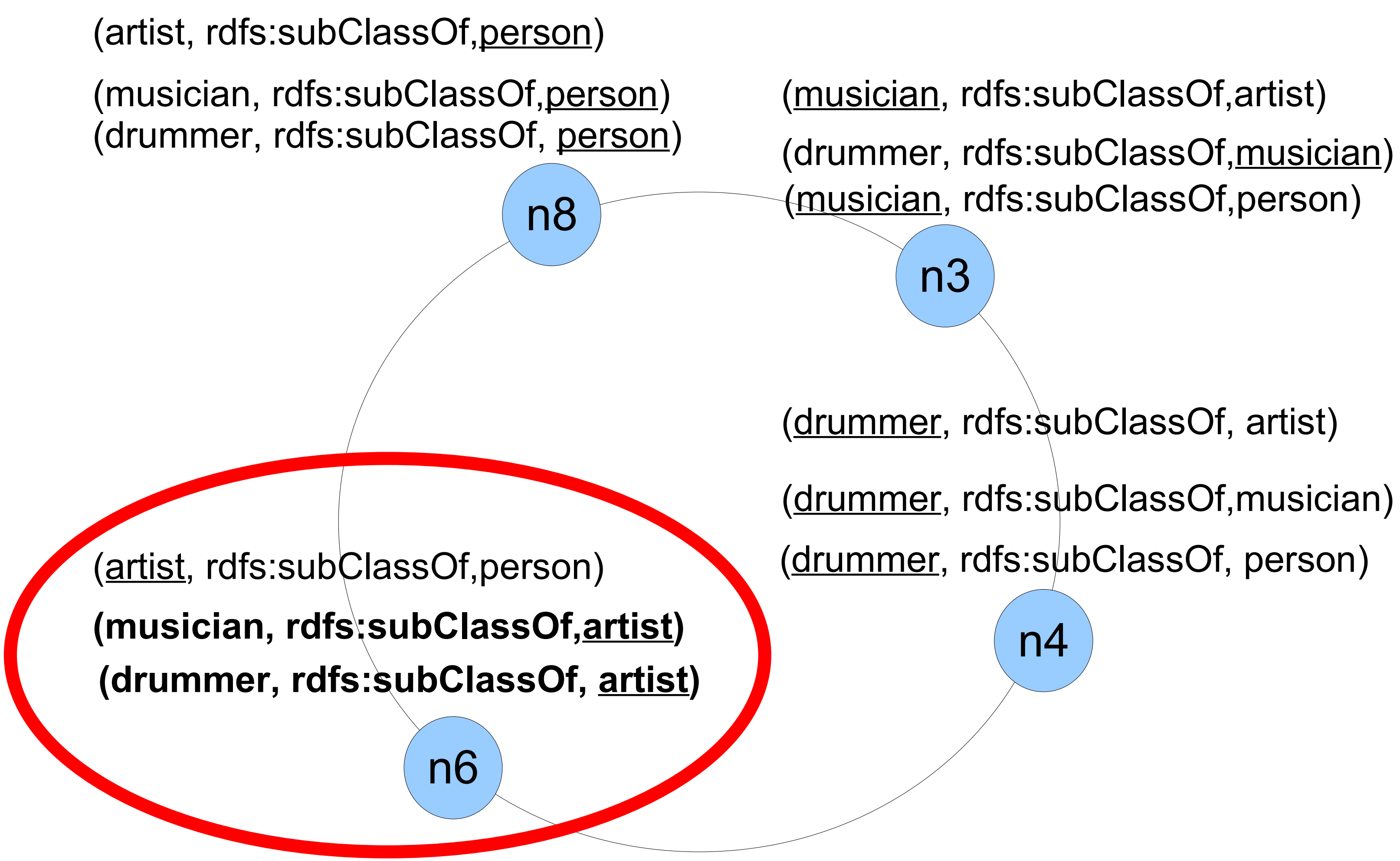


Forward Chaining Algorithm - Querying

Find all subclasses to artist:

(X, rdfs:subClassOf, artist)

- musician
- drummer



Backward Chaining Algorithm

only evaluates RDFS entailment rules at query processing time

Challenge:

- processing recursive rules in a distributed environment

Solution:

- adornment rules to get a good ordering for evaluating predicates

  Adornment of a predicate:

  - ordered string of ks, bs and fs where

    k: argument that is a key and that is bound

    b: bounded argument, that is not the key

    f: free argument

Backward Chaining Algorithm - Adorned RDFS entailment rules

$type^{k,f}(X,Y)$:- $triple^{k,b,f}(X, rdf{:}type, Y)$

$type^{f,k}(X,Y)$:- $triple^{f,b,k}(X, rdf{:}type, Y)$

$type^{k,f}(X,Y)$:- $triple^{k,b,f}(X, rdf{:}type, Z)$, $subClass^{f,f}(Z,Y)$

$type^{f,k}(X,Y)$:- $type^{f,f}(X,Z)$, $triple^{f,b,k}(Z, rdfs{:}subClassOf, Y)$

Backward Chaining Algorithm

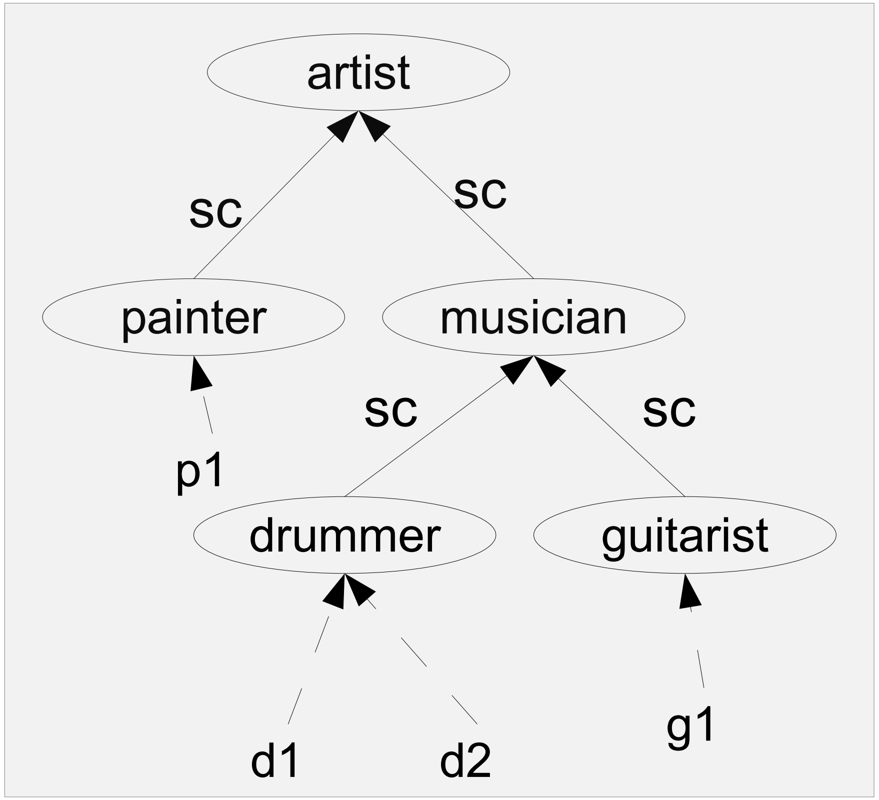
n1:
- (painter, rdfs:subClassOf, artist)
- (musician, rdfs:subClassOf, artist)

n4:
- (drummer, rdfs:subClassOf, musician)
- (d1,rdf:type,drummer)
- (d2,rdf:type,drummer)

n5:
- (guitarist, rdfs:subClassOf, musician)
- (g1,rdf:type,guitarist)

n6:
- (guitarist, rdfs:subClassOf, musician)
- (guitarist, rdfs:subClassOf, musician)
- (musician, rdfs:subClassOf, artist)

n8:
- (painter, rdfs:subClassOf, artist)
- (p1,rdf:type,painter)



Backward Chaining Algorithm

n1:
- (painter, rdfs:subClassOf, artist)
- (musician, rdfs:subClassOf, artist)

n4:
- (drummer, rdfs:subClassOf, musician)
- (d1,rdf:type,drummer)
- (d2,rdf:type,drummer)

n5:
- (guitarist, rdfs:subClassOf, musician)
- (g1,rdf:type,guitarist)

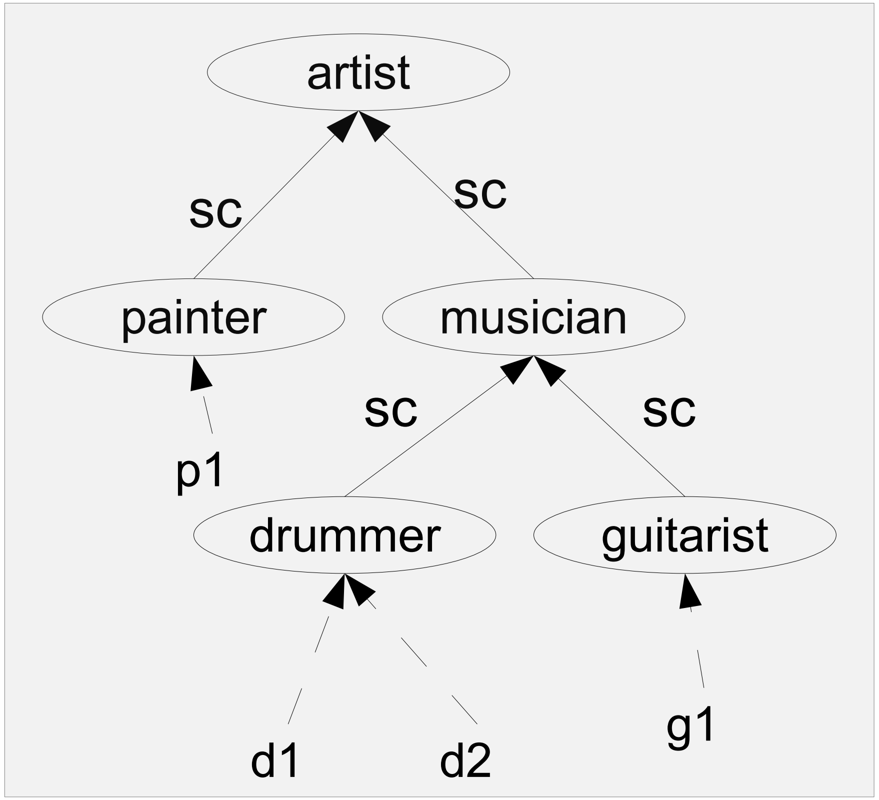
n6:
- (guitarist, rdfs:subClassOf, musician)
- (guitarist, rdfs:subClassOf, musician)
- (musician, rdfs:subClassOf, artist)

n8:
- (painter, rdfs:subClassOf, artist)
- (p1,rdf:type,painter)

Find all instances of class artist:

(X, rdf:type, artist)

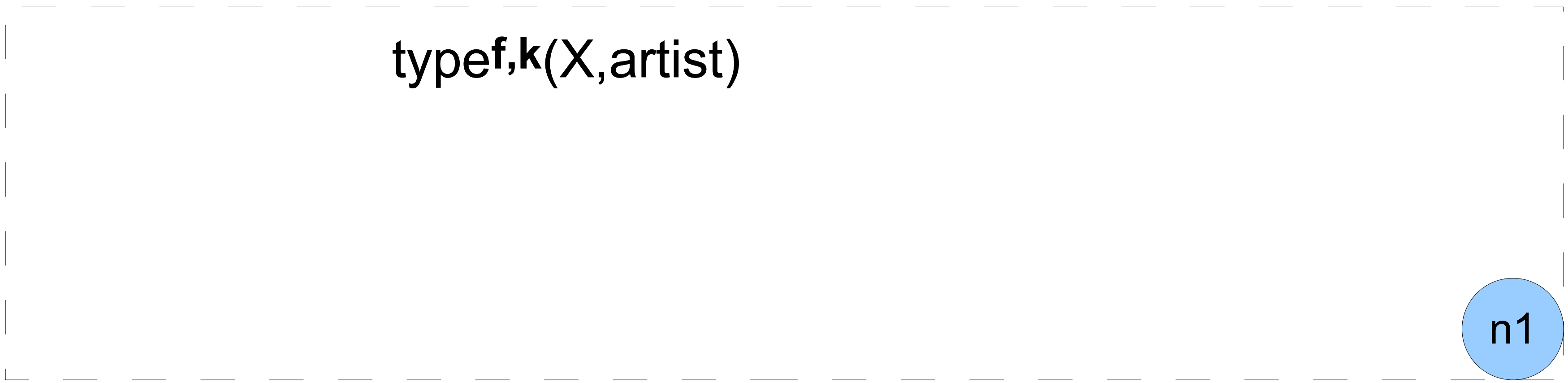
=> $\text{type}^{f,k}$(X,artist)



Backward Chaining Algorithm

Find all instances of class artist:

(X, rdf:type, artist)



Backward Chaining Algorithm - Adorned RDFS entailment rules

$type^{k,f}(X,Y)$:- $triple^{k,b,f}(X$, rdf:type, $Y)$

$type^{f,k}(X,Y)$:- $triple^{f,b,k}(X$, rdf:type, $Y)$

$type^{k,f}(X,Y)$:- $triple^{k,b,f}(X$, rdf:type, $Z)$, $subClass^{f,f}(Z,Y)$

$type^{f,k}(X,Y)$:- $type^{f,f}(X,Z)$, $triple^{f,b,k}(Z$, rdfs:subClassOf, $Y)$

Find all instances of class artist:

(X, rdf:type, artist)

=> $type^{f,k}$(X,artist)

Backward Chaining Algorithm

Find all instances of class artist:

(X, rdf:type, artist)

$type^{\mathbf{f,k}}(X,artist)$

$triple^{fbk}(X, rdf{:}type, artist)$

$triple^{fbk}(Z, rdfs{:}subClassOf, artist)$
$type^{ff}(X,Z)$

Z/painter

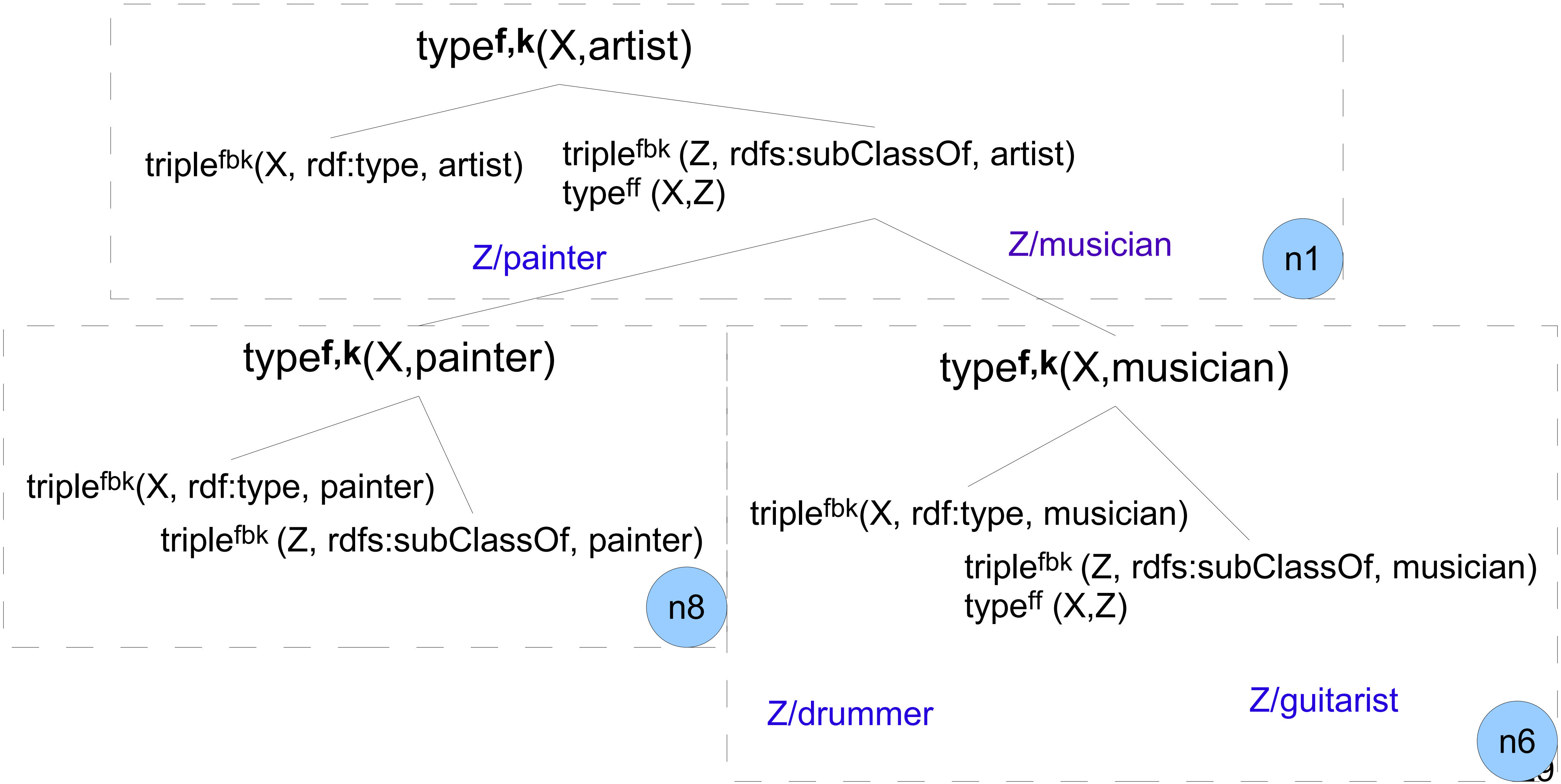
Z/musician

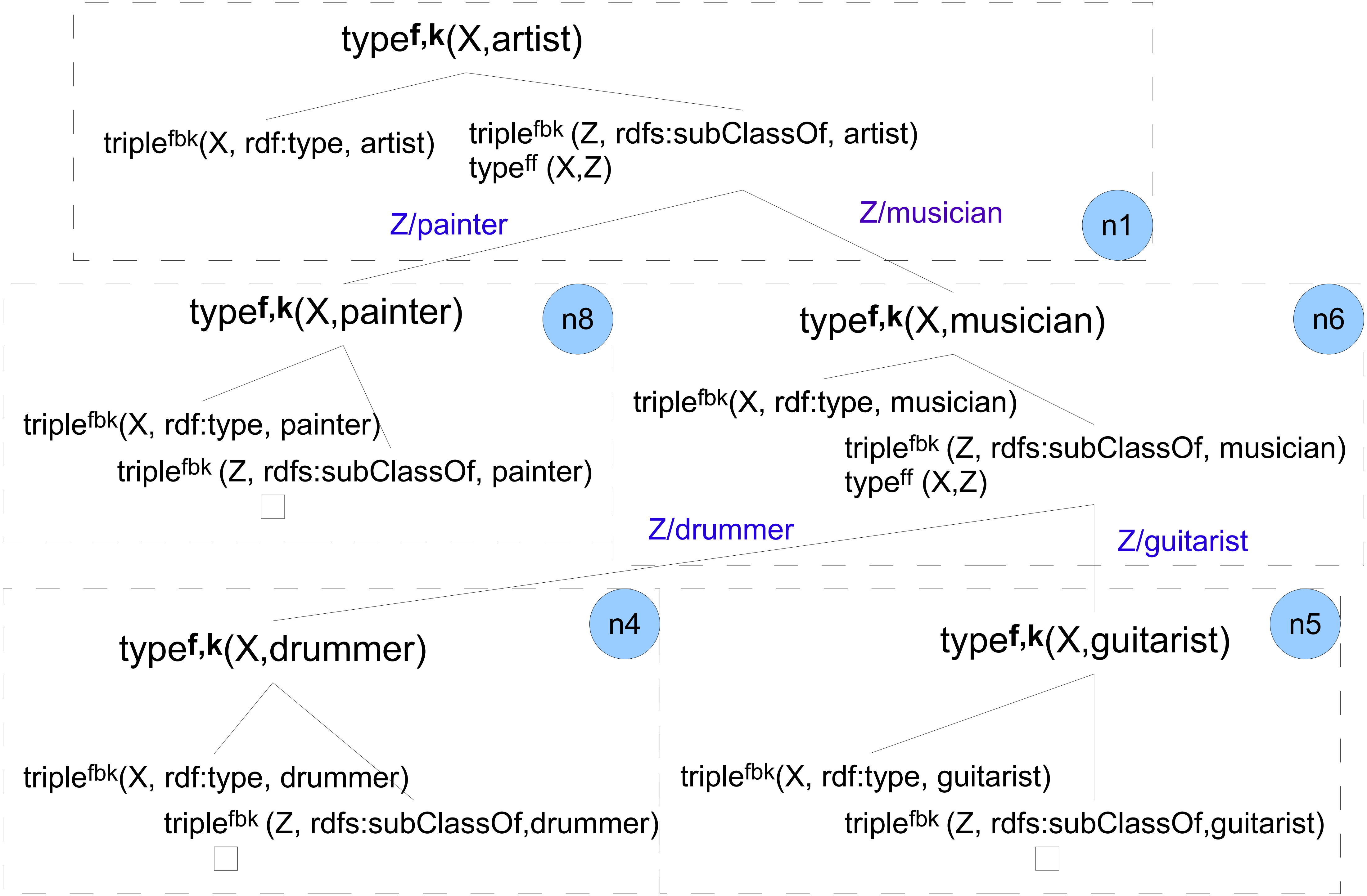
n1

Backward Chaining Algorithm

Find all instances of class artist:

(X, rdf:type, artist)




type^f,k(X,artist)
triple^fbk(X, rdf:type, artist)
triple^fbk (Z, rdfs:subClassOf, artist)
type^ff (X,Z)
Z/painter
Z/musician
n1
type^f,k(X,painter)
n8
triple^fbk(X, rdf:type, painter)
triple^fbk (Z, rdfs:subClassOf, painter)
type^f,k(X,musician)
n6
triple^fbk(X, rdf:type, musician)
triple^fbk (Z, rdfs:subClassOf, musician)
type^ff (X,Z)
Z/drummer
Z/guitarist
type^f,k(X,drummer)
n4
triple^fbk(X, rdf:type, drummer)
triple^fbk (Z, rdfs:subClassOf,drummer)
type^f,k(X,guitarist)
n5
triple^fbk(X, rdf:type, guitarist)
triple^fbk (Z, rdfs:subClassOf,guitarist)

Backward Chaining Algorithm

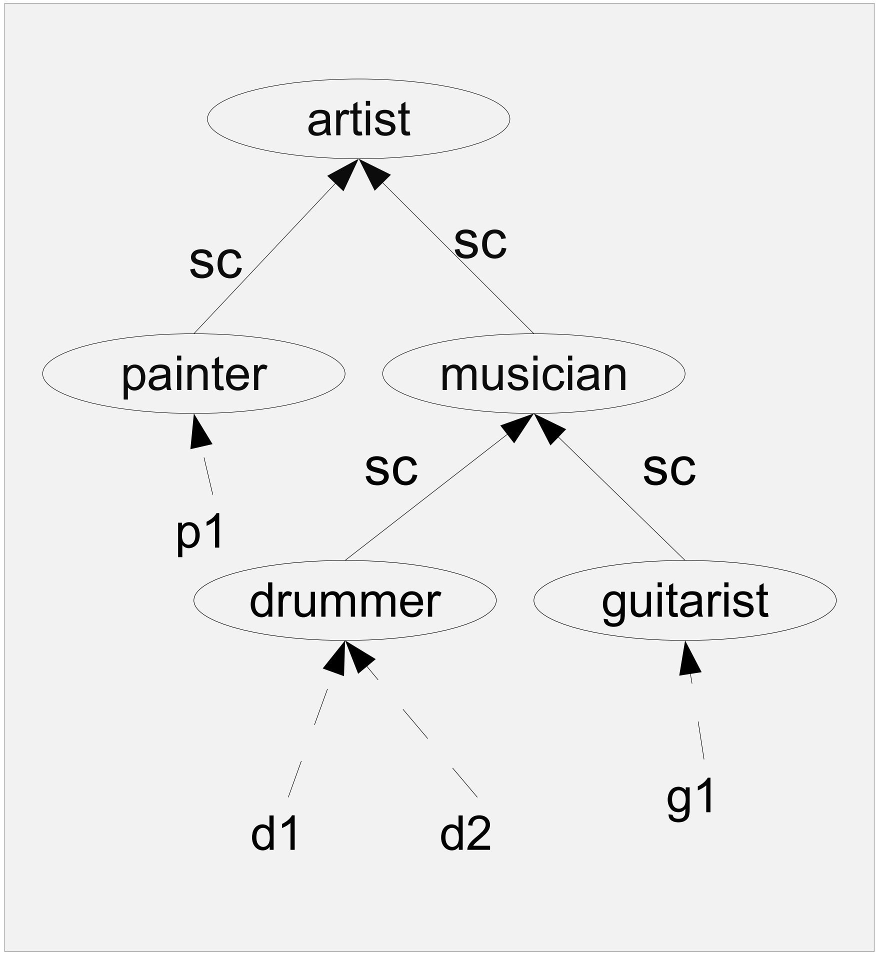
Find all instances of class artist:

(X, rdf:type, artist)

- p1

- d1, d2

- g1



3. Comparison FC and BC

Network traffic while storing:

- FC: network traffic increases as the depth of the rdfs-graph increases
- BC: constant number of messages independent from the rdf schema

Storage load:

- FC:increases with the depth of the rdfs graph because of generated redundencies
- BC:remains constant regardless the depth of the rdfs-graph

Query processing time:

- FC:only one message is sent to the responsible node
- BC: depends on the depth of the rdfs-graph

4. Experimental Evaluation

Setup:

- both algorithms have been implemented on top of Bamboo DHT (Handling Churn in a DHT, S. Rhea, D. Geels, T. Roscoe)

- PlanetLab network (http://www.planet-lab.org) was used as testbed (123 nodes)

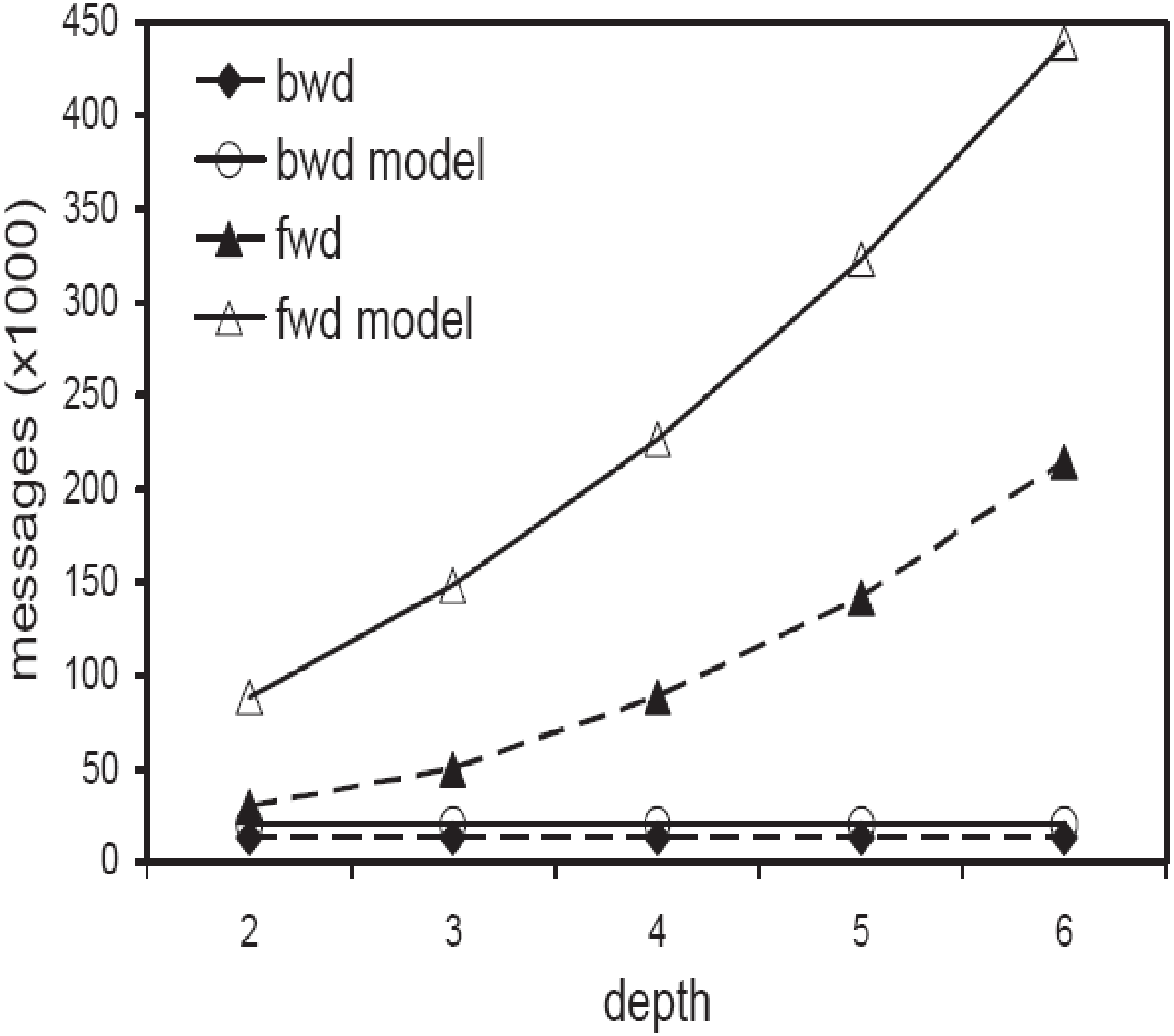
- synthetic data was generated from RBench generator (http://athena.ics.forth.gr:9090/RDF/RBench)

  -> number of instances : 10^4

  -> RDFS class hierarchy tree depth: 2-6

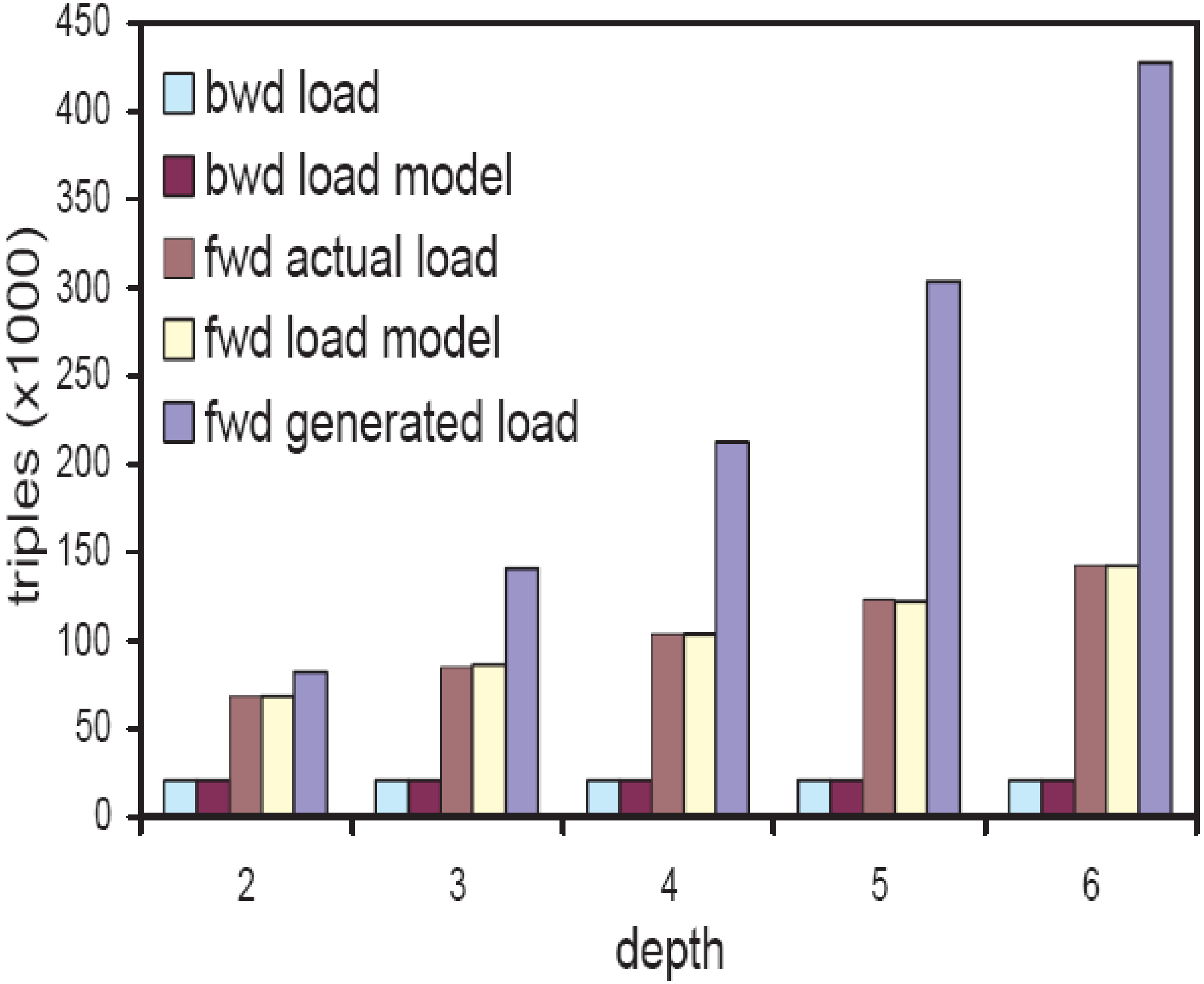
- Query: Find all instances of the root class

4. Experimental Evaluation: Number of messages sent while storing



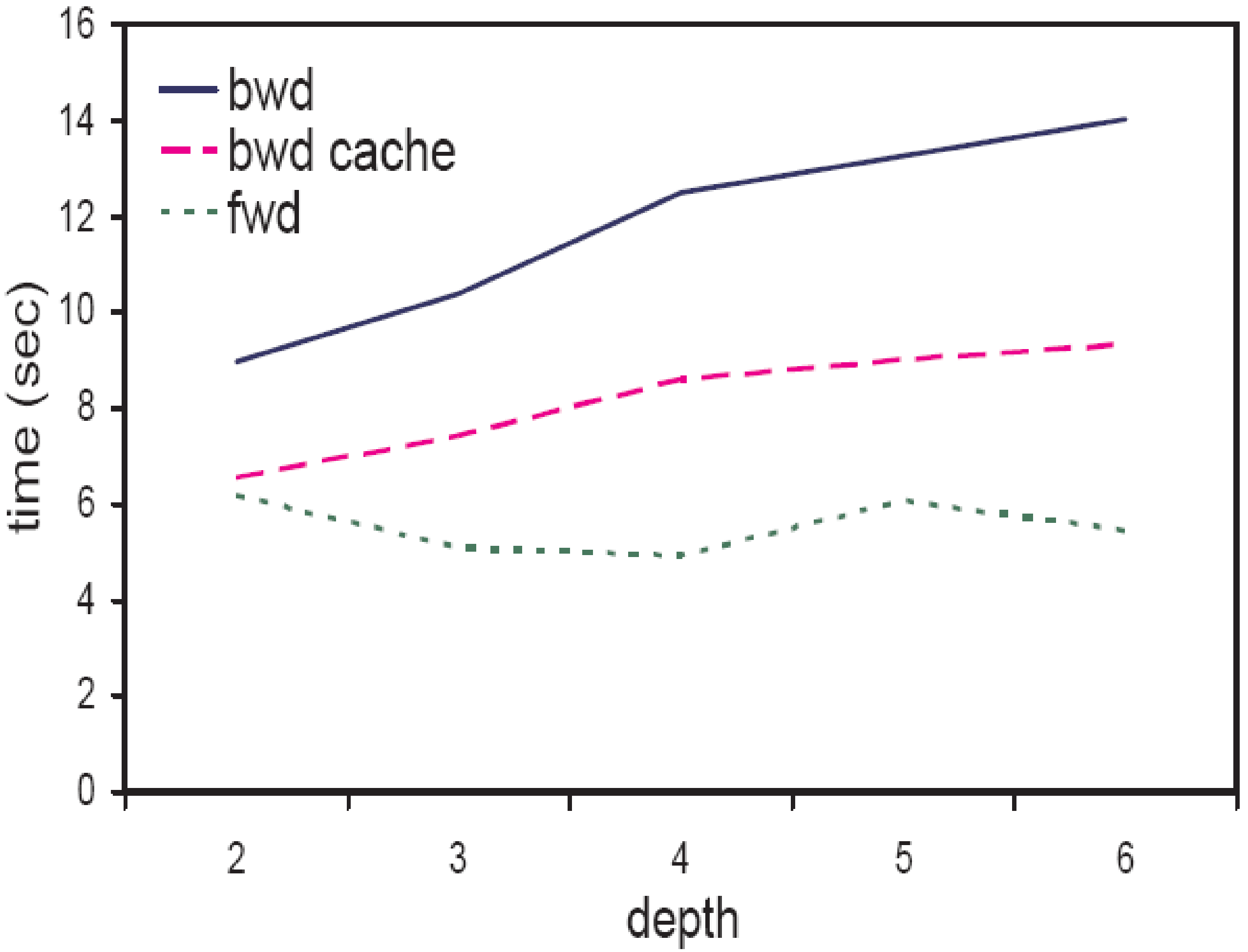
Source: RDFS Reasoning and Query answering on Top of DHTs , Zoi Kaoudi, Iris Miliaraki, Manolis Koubarakis

4. Experimental Evaluation: Storage load



Source: RDFS Reasoning and Query answering on Top of DHTs , Zoi Kaoudi, Iris Miliaraki, Manolis Koubarakis

4. Experimental Evaluation: Query response time



Source: RDFS Reasoning and Query answering on Top of DHTs , Zoi Kaoudi, Iris Miliaraki, Manolis Koubarakis

References:

1. RDFS Reasoning and Query answering on Top of DHTs , Zoi Kaoudi, Iris Miliaraki, Manolis Koubarakis

2. RDF Primer , W3C Recommendation 10 February 2004
http://www.w3.org/TR/rdf-primer/

3. RDF Vocabulary Description Language 1.0: RDF Schema W3C Recommendation 10 February 2004
http://www.w3.org/TR/rdf-schema/

4. RDF Semantics , W3C Recommendation 10 February 2004
http://www.w3.org/TR/rdf-mt/

Thank you!

Questions?