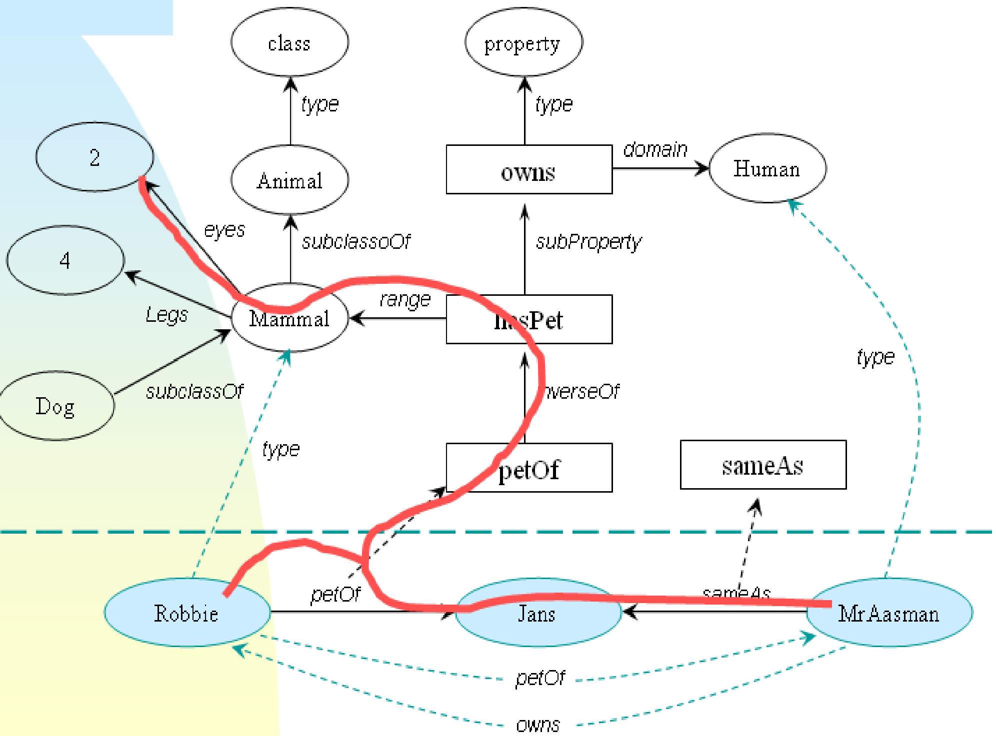
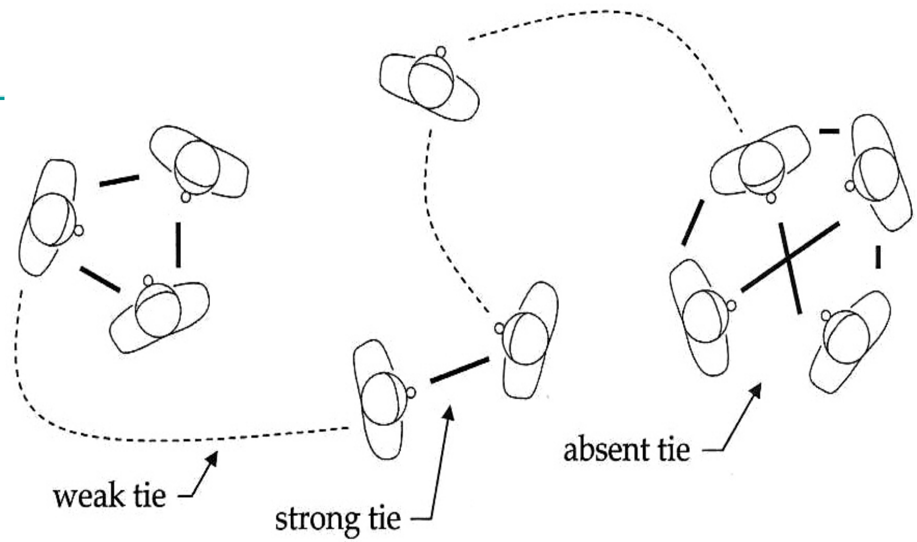
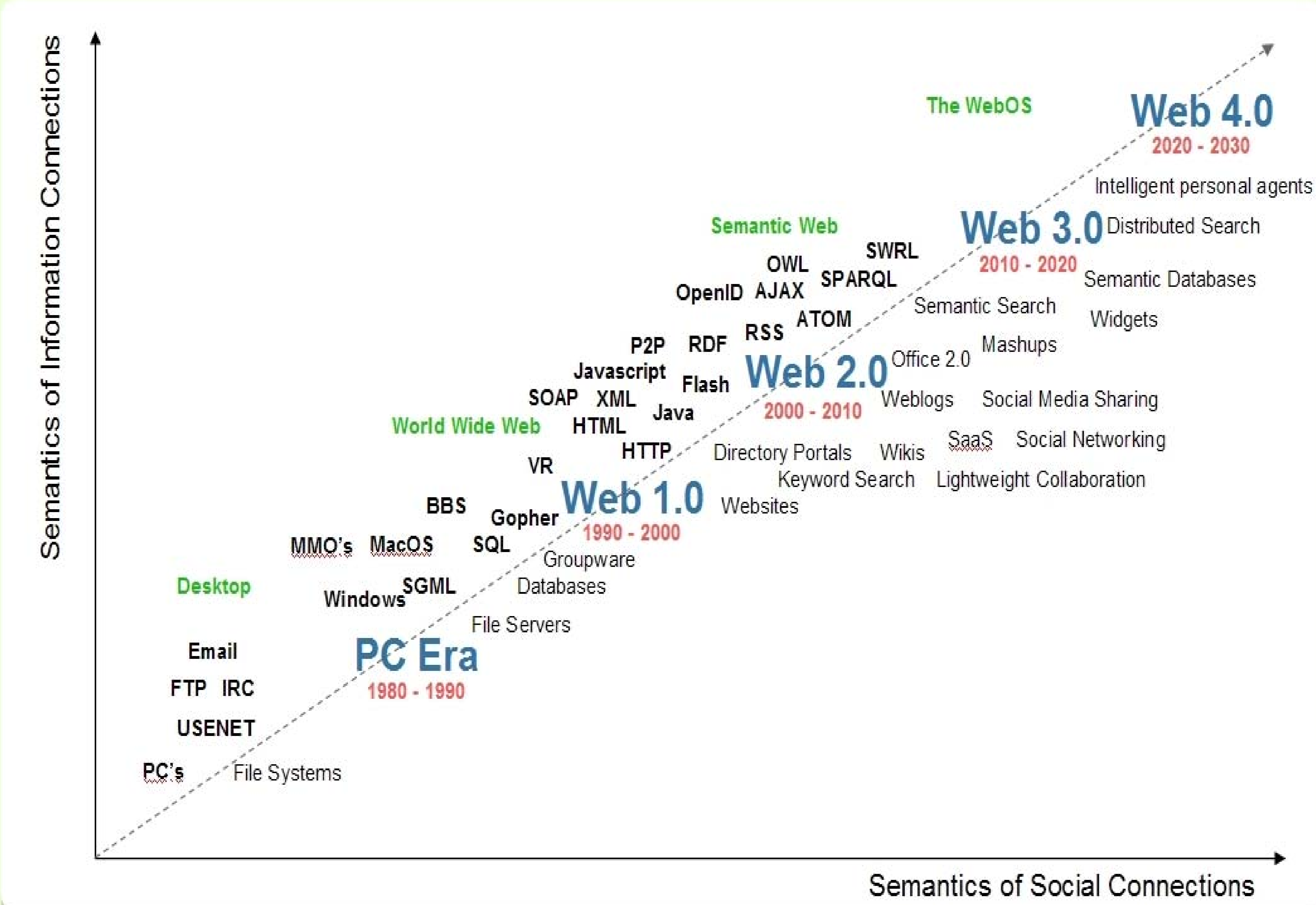
AllegroGraph 2.1

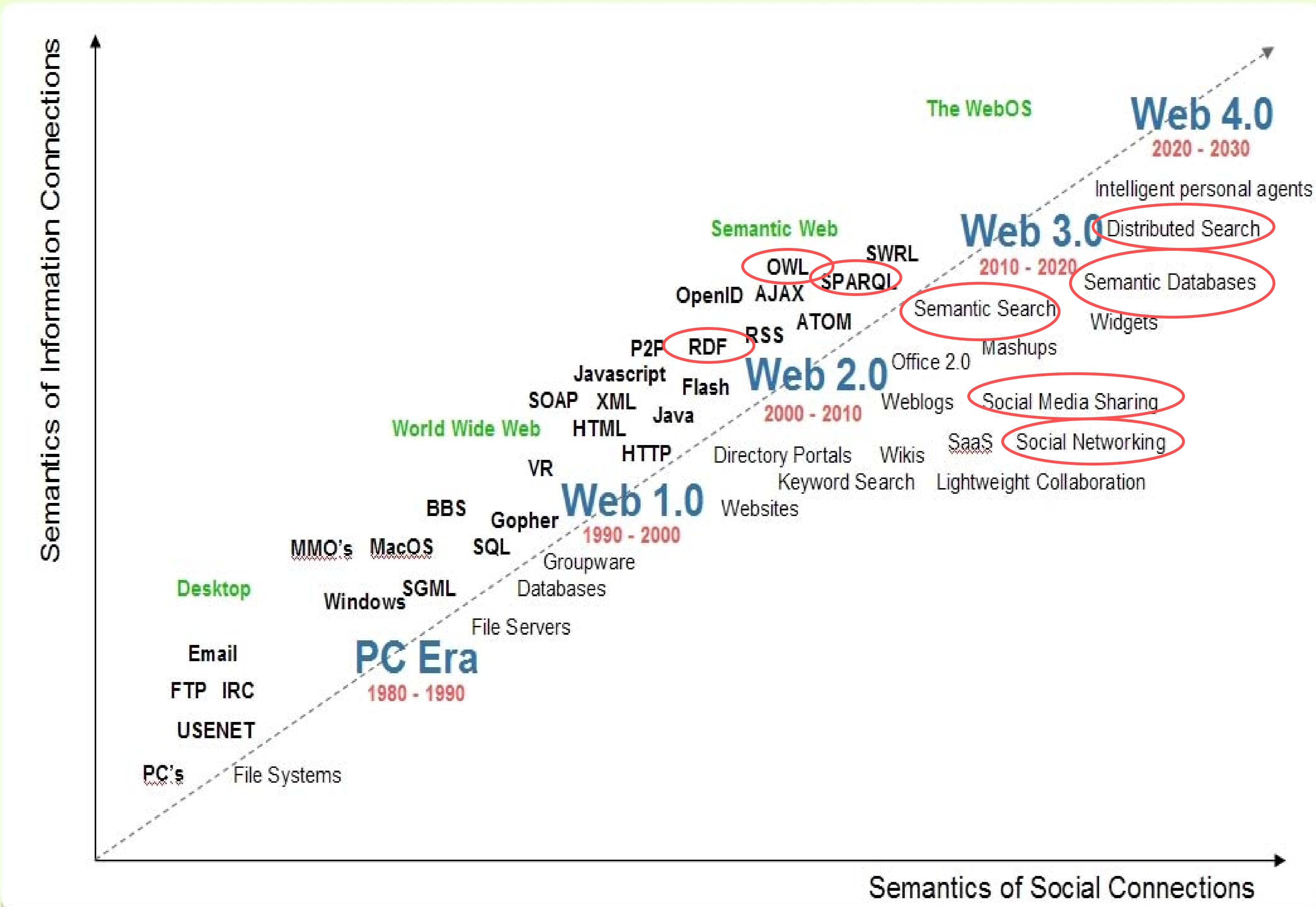
Reasoning with A Web 3.0 Database



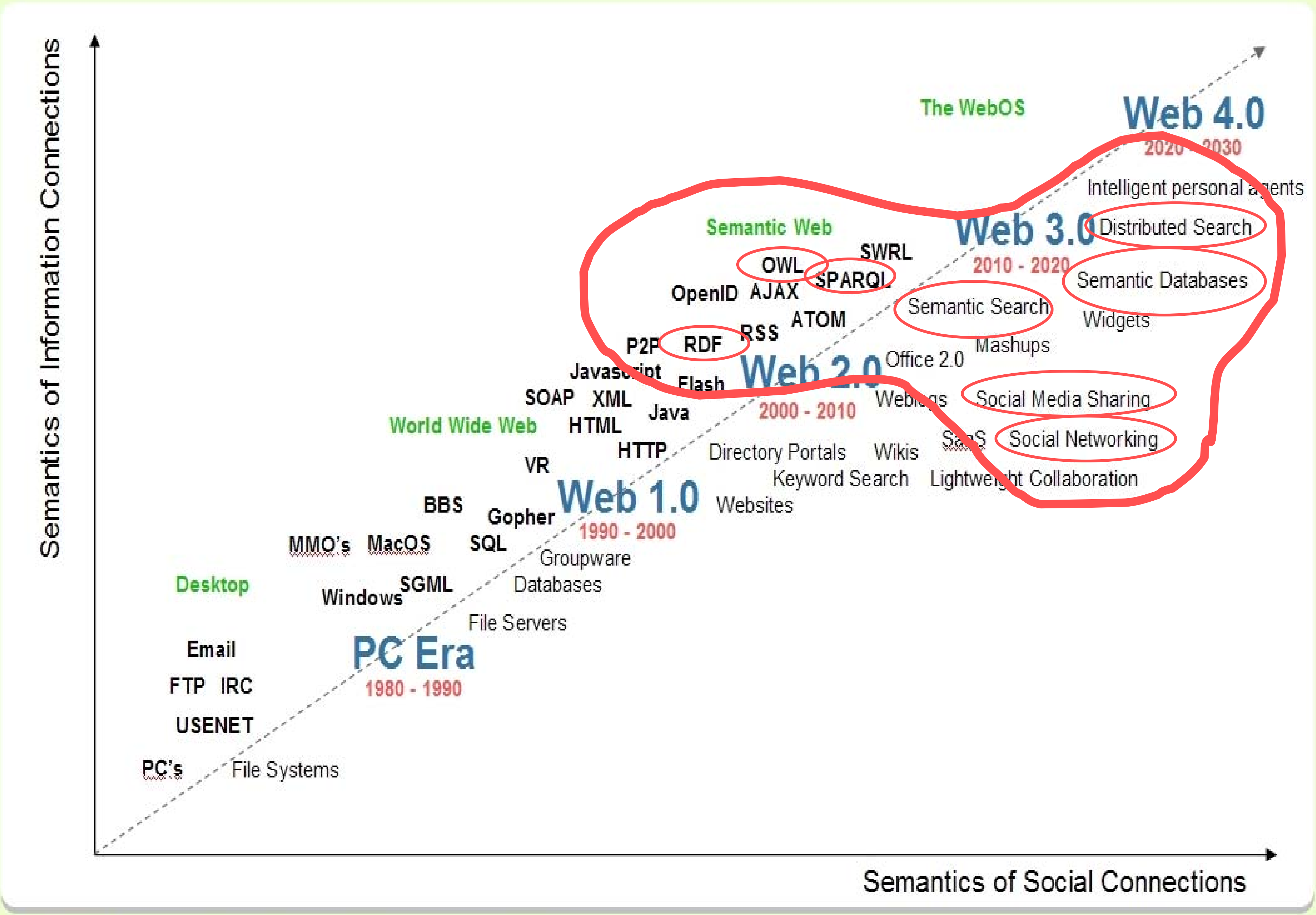




Source: Radar Networks & Nova Spivack, 2007 – www.radarnetworks.com



Source: Radar Networks & Nova Spivack, 2007 – www.radarnetworks.com



Source: Radar Networks & Nova Spivack, 2007 – www.radarnetworks.com

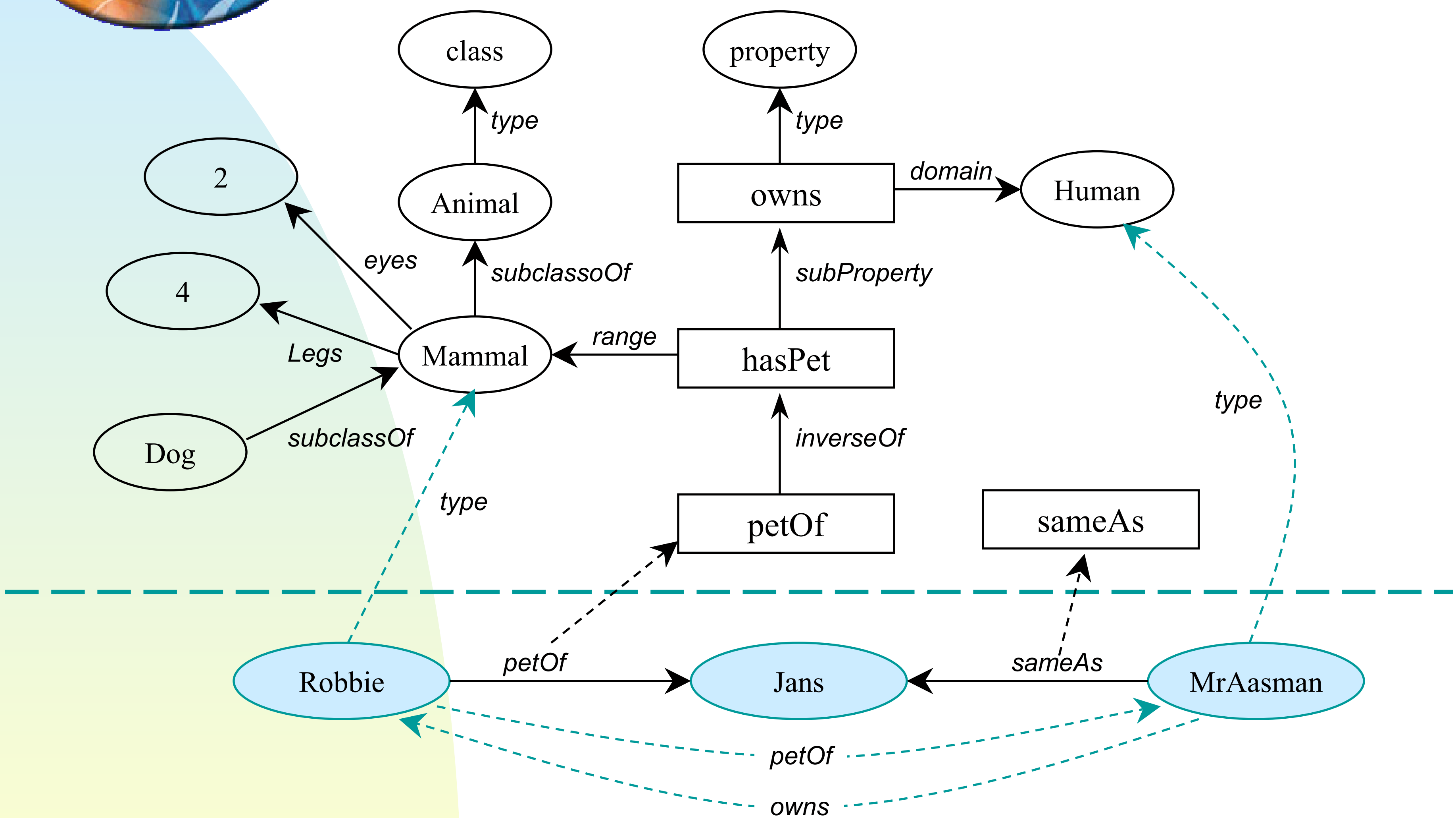
This Seminar

- The basics of RDF and Triples
- AllegroGraph as a triple-store
- Advanced Features
- Reasoning with Prolog
- Reasoning with RDFS++
- Demo…

RDF: Resource Description Framework

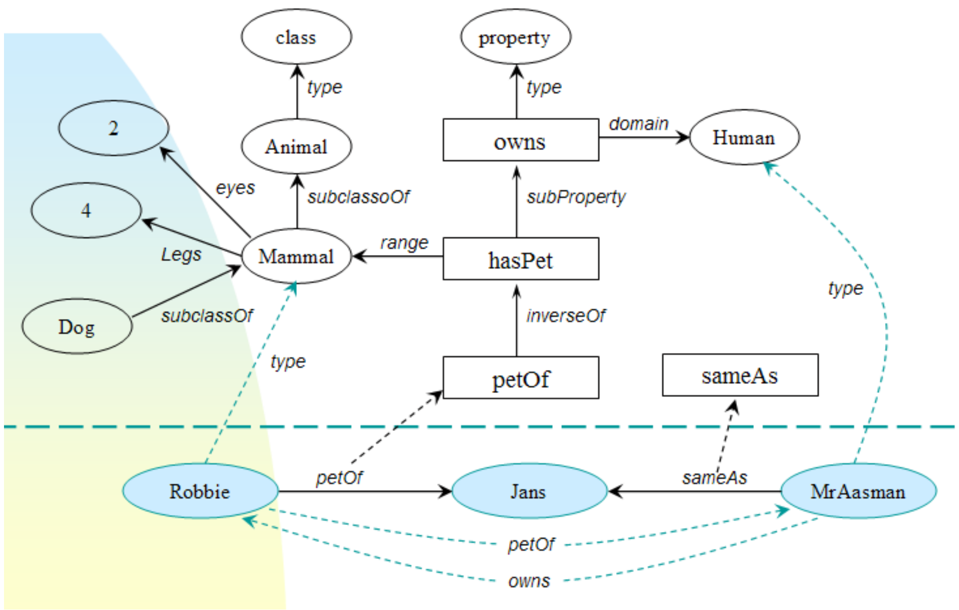
- W3C's knowledge representation standard for the semantic web.
- Semantic web is basically the Web 3.0
  - Metadata for content (webpages, multimedia contents, versioning) allows machines to help people search information and organize their lives.
- Quickly became standard for metadata in general
- But: nothing more than a way to serialize old-fashioned semantic networks.

A typical semantic network from the late sixties.



The same network serialized into <subject, predicate, object> triples

- Animal type class
- Mammal subclassOf Animal
- Mammal eyes 2
- Mammal legs 4
- Dog subclassOf Mammal
- owns type Property
- owns domain Human
- hasPet subproperty owns
- hasPet range Mammal
- hasPet inverseOf petOf
- Robbie petOf Jans
- MrAasman sameAs Jans



RDF: Subject, predicate, object turned into Resources (URIs) and literals.

Resource

literal

```
<http://www.franz.com/simple#Animal>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.w3.org/2002/07/owl#Class>.
<http://www.franz.com/simple#Mammal>
   <http://www.w3.org/2000/01/rdf-schema#subClassOf>
   <http://www.franz.com/simple#Animal> .
<http://www.franz.com/simple#Mammal>
   <http://www.franz.com/simple#eyes> "two" .
<http://www.franz.com/simple#Mamma>
   <http://www.franz.com/simple#legs> "four" .
<http://www.franz.com/simple#Dog>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.franz.com/simple#Mammal> .
<http://www.franz.com/simple#owns>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.franz.com/simple#property> .
<http://www.franz.com/simple#owns>
   <http://www.w3.org/2000/01/rdf-schema#domain>
   <http://www.franz.com/simple#Human> .
<http://www.franz.com/simple#haspet>
   <http://www.franz.com/simple#subproperty>
   <http://www.franz.com/simple#owns> .
```

Triples in RDF/XML


```
<AdultMaleHuman_Terrorist rdf:ID="AbuAbdullahAbuDijana">
    <rdfs:comment>An Al-Qaida member who, along with Abu Fadhl
        al-Makkee, directed al-Fadl to set up meetings in order to
        purchase uranium.</rdfs:comment>
    <guid>5a248a52-3916-11d6-8000-00a0c99cc5ae</guid>
</AdultMaleHuman_Terrorist>
<owl:Class rdf:ID="IraqiPerson_AdultMaleHuman_Terrorist_Leader">
    <rdfs:subClassOf rdf:resource="#IraqiPerson"/>
    <rdfs:subClassOf rdf:resource="#AdultMaleHuman"/>
    <rdfs:subClassOf rdf:resource="#Terrorist"/>
    <rdfs:subClassOf rdf:resource="#Leader"/>
</owl:Class>
<IraqiPerson_AdultMaleHuman_Terrorist_Leader rdf:ID="AbuAyoubAlIraqi">
    <rdfs:comment>Al Qaeda member.  Attended the first Al Qaeda
        formation meeting in Khost, Afghanistan.  Emir of the Al
        Qaeda formation meeting, and originally the emir of Al Qaeda.</rdfs:comme
    <guid>0356bdfa-37bb-11d6-8000-00a0c99cc5ae</guid>
    <boss rdf:resource="#OsamaBinLaden"/>
    <hasBeenIn rdf:resource="#CityOfJajiAfghanistan"/>
    <hasBeenIn rdf:resource="#CityOfKhostAfghanistan"/>
</IraqiPerson_AdultMaleHuman_Terrorist_Leader>
<owl:Class rdf:ID="Terrorist_Mullah_MaleHuman">
    <rdfs:subClassOf rdf:resource="#Terrorist"/>
    <rdfs:subClassOf rdf:resource="#Mullah"/>
    <rdfs:subClassOf rdf:resource="#MaleHuman"/>
</owl:Class>
<Terrorist_Mullah_MaleHuman rdf:ID="AbuBakarBashir">
    <rdfs:comment/>
    <guid>003b0ed1-d1c2-11d7-9801-0002b35bb117</guid>
</Terrorist_Mullah_MaleHuman>
```

Triples in a triple-store..

Triples are number vectors in memory and on disk.

```
#(2 4 5 6)
#(7 9 2 10)
#(7 11 12 13)
#(14 15 16 17)
#(18 4 7 19)
#(20 4 21 22)
#(20 23 24 25)
#(26 27 20 28)
#(26 29 7 30)
#(31 32 33 34)
```

....

Dictionary

```
franz:Animal = 2
rdf:type = 4
owl:Class = 5
franz:Eyes = 9
```

…….

Reverse Dictionary

```
2 = franz:Animal
4 = rdf:Type
5 = owl:class
6 = Triple-id
7 = franz:Mammal
```

……

Triples are indexed in three ways…

- SPO
- POS
- OSP

```
Get-triples(jans,?x,?y)
Get-triples(jans,isa,?x)
Get-triples(jans,isa,psychologist)
Get-triples(?x,isa,?y)
Get-triples(?x,isa,psychologist)
Get-triples(?x,?y,psychologist)
Get-triples(jans,?y,psychologist)
```

SPO → Get-triples(jans,?x,?y), Get-triples(jans,isa,?x), Get-triples(jans,isa,psychologist)

POS → Get-triples(?x,isa,?y), Get-triples(?x,isa,psychologist)

OSP → Get-triples(?x,?y,psychologist), Get-triples(jans,?y,psychologist)

And six ways with named graphs.

The difference with a relational database?

```
<triple 32: "person2" "type" "person">
<triple 33: "person2" "first-name" "Rose">
<triple 34: "person2" "middle-initial" "Elizabeth">
<triple 35: "person2" "last-name" "Fitzgerald">
<triple 36: "person2" "suffix" "none">
<triple 37: "person2" "alma-mater" "Sacred-Heart-Convent">
<triple 38: "person2" "birth-year" "1890">
<triple 39: "person2" "death-year" "1995">
<triple 40: "person2" "sex" "female">
<triple 41: "person2" "spouse" "person1">
<triple 58: "person2" "has-child" "person17">
<triple 56: "person2" "has-child" "person15">
<triple 54: "person2" "has-child" "person13">
<triple 52: "person2" "has-child" "person11">
<triple 50: "person2" "has-child" "person9">
<triple 48: "person2" "has-child" "person7">
<triple 46: "person2" "has-child" "person6">
<triple 44: "person2" "has-child" "person4">
<triple 42: "person2" "has-child" "person3">
<triple 60: "person2" "profession" "home-maker">
```

An artist's impression of the same info in a RDBM

Table Person

ID	First-Name	Last-Name	Middle-In.	DOB	DOD	PlaceOB	Sex
2	Rose	Fitzgerald	Elizabeth	1890	1995	1	F

Table Spouses

ID1	ID2
2	1

Table to-schools

ID1	SchoolID
2	3

Table Schools

ID	Name
3	Sacred-Heart

Table has-profession

ID1	ProfID
2	3

Table Professions

ID	Name
3	Home-maker

Table Has-Child

ID1	ID2
2	17
2	15
2	14
2	13

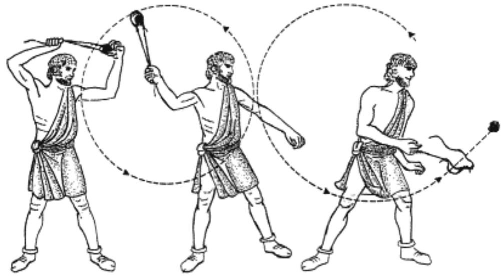
Table Place

ID	Name	State	Longitude	Latitude
1	Boston	MA	42.3	-71.4

With a graph database

- you add new predicates without changing any schema
- one-to-many relations are directly encoded without the indirection of tables
- You never think about what to index because all the predicates are indexed

AllegroGraph is

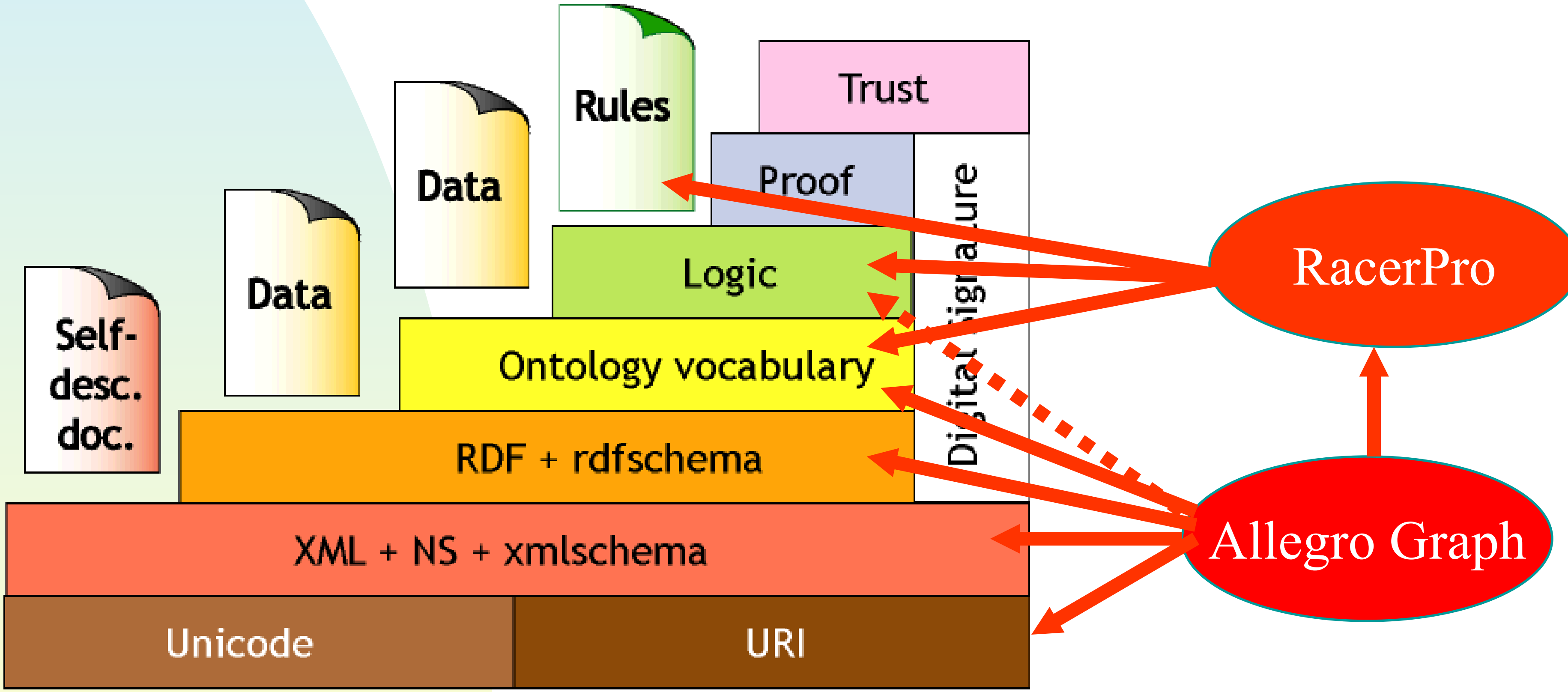


- A scalable persistent triple store
  - 1.1 Billion triples in 23 hours on a $5000 dollar box
  - 20 to 40,000 triples per second,
  - Record query performance on LUBM benchmark queries.
- Based on standards
  - RDF, RDFS, OWL, SPARQL, Named Graphs
- Two modes of working
  - Standalone for analytics
  - Client/Server for real time services
- Accessible from any language
  - Java: we adhere to Sesame and Jena remote repository APIs
  - .Net, Python, Ruby, Lisp, C through REST interface
- Reasoning
  - Prolog, RDFS++ and Description Logics (direct connection with Racer)
- GUI & Ontology Management
  - TopBraid Composer, RacerPorter

AllegroGraph Unique Features

- RDFS++ Reasoner
- Direct reification
  - Triples point to triples
- Named Graphs fully supported
  - But slot can also be used for weights, trust factors, provenance, distance, etc.
- Native data types and efficient range queries
  - Existing triple stores store all data as strings, range queries inefficient
  - AllegroGraph supports most xml schema types (dates, times, longitudes, latitudes, durations, telephone numbers, etc)
- Basic geospatial and temporal primitives
- Social Network Analysis library
- Combine it all with Prolog & Sparql

AllegroGraph as RDF database



AllegroGraph Stack

http/xml-rpc/soap/custom	Custom Apps
Server	Java, c/++, .Net
Query Engines: SPARQL, RDF-Prolog, Lisp	
Reasoners: RDFS++, Racer	
API: store(s,p,o), get([s],[p],[o]), read-file(uri)	
Graph Database: Dictionaries, Indices, Caches	
Allegro CL Compiler	
Machine Code (on 18 platforms)	

Allegro Graph Stack

http/xml-rpc/soap/	Custom Apps	TopBraid Composer
Server	FFI: java, c/++, .Net	
(Black Box)		

Reasoning with RDF Prolog

RDF Prolog

- An Industrial strength Prolog embedded in ACL, completely geared to RDF.
- Prolog clauses are compiled to machine code
- Conforms to Clocksin & Mellishs Prolog and ISO kernel specification
- Competitive with commercial Prologs

The Kennedy family

<triple 1: "http://www.franz.com/simple#person1" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://www.franz.com/simple#person">
<triple 2: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#first-name" "http://www.franz.com/simple#Joseph">
<triple 3: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#middle-initial" "http://www.franz.com/simple#Patrick">
<triple 4: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#last-name" "http://www.franz.com/simple#Kennedy">
<triple 5: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#suffix" "http://www.franz.com/simple#none">
<triple 6: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#alma-mater" "http://www.franz.com/simple#Harvard">
<triple 7: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#birth-year" "http://www.franz.com/simple#1888">
<triple 8: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#death-year" "http://www.franz.com/simple#1969">
<triple 9: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#sex" "http://www.franz.com/simple#male">
<triple 10: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#spouse" "http://www.franz.com/simple#person2">
<triple 27: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person17">
<triple 25: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person15">
<triple 23: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person13">
<triple 21: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person11">
<triple 19: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person9">
<triple 17: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person7">
<triple 15: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person6">
<triple 13: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person4">
<triple 11: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#has-child" "http://www.franz.com/simple#person3">
<triple 31: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#profession" "http://www.franz.com/simple#ambassador">
<triple 30: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#profession" "http://www.franz.com/simple#producer">
<triple 29: "http://www.franz.com/simple#person1" "http://www.franz.com/simple#profession" "http://www.franz.com/simple#banker">

In shorthand notation

```
<triple 32: "person2" "type" "person">
<triple 33: "person2" "first-name" "Rose">
<triple 34: "person2" "middle-initial" "Elizabeth">
<triple 35: "person2" "last-name" "Fitzgerald">
<triple 36: "person2" "suffix" "none">
<triple 37: "person2" "alma-mater" "Sacred-Heart-Convent">
<triple 38: "person2" "birth-year" "1890">
<triple 39: "person2" "death-year" "1995">
<triple 40: "person2" "sex" "female">
<triple 41: "person2" "spouse" "person1">
<triple 58: "person2" "has-child" "person17">
<triple 56: "person2" "has-child" "person15">
<triple 54: "person2" "has-child" "person13">
<triple 52: "person2" "has-child" "person11">
<triple 50: "person2" "has-child" "person9">
<triple 48: "person2" "has-child" "person7">
<triple 46: "person2" "has-child" "person6">
<triple 44: "person2" "has-child" "person4">
<triple 42: "person2" "has-child" "person3">
<triple 60: "person2" "profession" "home-maker">
```

In shorthand notation

```
<triple 32: "person2" "type" "person">
<triple 33: "person2" "first-name" "Rose">
<triple 34: "person2" "middle-initial" "Elizabeth">
<triple 35: "person2" "last-name" "Fitzgerald">
<triple 36: "person2" "suffix" "none">
<triple 37: "person2" "alma-mater" "Sacred-Heart-Convent">
<triple 38: "person2" "birth-year" "1890">
<triple 39: "person2" "death-year" "1995">
<triple 40: "person2" "sex" "female">
<triple 41: "person2" "spouse" "person1">
<triple 58: "person2" "has-child" "person17">
<triple 56: "person2" "has-child" "person15">
<triple 54: "person2" "has-child" "person13">
<triple 52: "person2" "has-child" "person11">
<triple 50: "person2" "has-child" "person9">
<triple 48: "person2" "has-child" "person7">
<triple 46: "person2" "has-child" "person6">
<triple 44: "person2" "has-child" "person4">
<triple 42: "person2" "has-child" "person3">
<triple 60: "person2" "profession" "home-maker">
```

Building semantic relations on top of RDF

```
(<-- (male ?x)
     (q ?x !o:sex !o:male))

(<-- (female ?x)
     (q ?x !o:sex !o:female))

(<-- (father ?x ?y)
     (male ?x)
     (q ?x !o:has-child ?y))

(<-- (mother ?x ?y)
     (female ?x)
     (q ?x !o:has-child ?y))

(<-- (parent ?x ?y)
     (father ?x ?y))

(<- (parent ?x ?y)
    (mother ?x ?y))
```

```
(<-- (grandparent ?x ?y)
     (parent ?x ?z)
     (parent ?z ?y))

(<-- (grandchild ?x ?y)
     (grandparent ?y ?x))

(<-- (ancestor ?x ?y)
     (parent ?x ?y))

(<-  (ancestor ?x ?y)
     (parent ?x ?z)
     (ancestor ?z ?y))

(<-- (descendent ?x ?y)
     (ancestor ?y ?x))
```

Building semantic relations on top of RDF..

```
(<-- (aunt ?x ?y)
     (father ?z ?x)
     (female ?x)
     (father ?z ?w)
     (not (= ?x ?w))
     (parent ?w ?y))

(<-- (uncle ?x ?y)
     (father ?z ?x)
     (male ?x)
     (father ?z ?w)
     (not (= ?x ?w))
     (parent ?w ?y))

(<-- (nephew ?x ?y)
     (aunt ?y ?x)
     (male ?x))

(<- (nephew ?x ?y)
    (uncle ?y ?x)
    (male ?x))

(<-- (niece ?x ?y)
     (aunt ?y ?x)
     (female ?x))

(<- (niece ?x ?y)
    (uncle ?y ?x)
    (female ?x))

(<-- (parent-child-ivy-league ?x ?y)
     (q ?x !!o:alma-mater ?am)
     (q ?am !!o:ivy-league !!o:true)
     (parent ?x ?y)
     (q ?y !!o:alma-mater ?am2)
     (q ?am2 !!o:ivy-league !!o:true))
```

A straight forward query

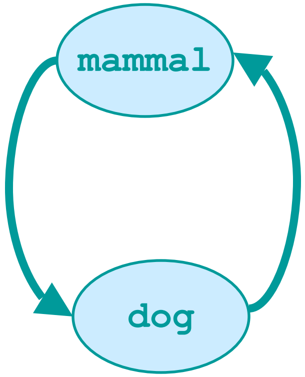
```
rdf(18): ((male ?x)  (full-name ?x ?name) (print ?name))

"Michael nil Allen"
"Alfred nil Tucker"
"Cart Harmon Hood"
"Mark nil Bailey"
"Andrew Mark Cuomo"
"Paul Michael Hill"
"Jeffrey Robert Ruhe"
"David Lee Townsend"
"Robert B Pender"
"James Peter McKelvy"
"Arnold Alois Schwarzenegger"
"Edwin Arthur Schlossberg"
"Patrick Joseph Kennedy"
"Edward M Kennedy"
"William Kennedy Smith"
"Stephen E Smith"
… and twenty more…
```

Advanced query

```
(?- (find-relations ?x ?y 2))

……
 John Fitzgerald Kennedy : Patrick Bouvier Kennedy
   --> (father parent ancestor)

John Fitzgerald Kennedy : John F Kennedy
   --> (father parent ancestor parent-child-have-same-name
     parent-child-went-to-ivy-league-school)

John Fitzgerald Kennedy : Caroline Bouvier Kennedy
   --> (father parent ancestor parent-child-went-to-ivy-
   league-school)

 Rose Elizabeth Fitzgerald : Patrick Joseph Kennedy
   --> (grandparent ancestor)
…….
```

Reasoning with RDFS++

RDFS: putting constraints on RDF



- RDF allowed everything
  - Mammal type class
  - Dog subclass of Mammal
  - Mammal subclass of Dog
- In order to allow for systematic reasoning RDF got semantics (schema)

RDFS

- Core classes
  - rdfs:resource, rdfs:class, rdfs:literal,
  - rdfs: property, rdf:statement
- Defining relationships
  - rdf:type, rdfs:subClassOf, rdfs:subPropertyOf
- Core restrictions
  - rdfs:domain
  - rdfs:range

OWL

- The marriage between
  - Object oriented type system
  - Well understood Description logic
  - Web languages like XML and RDF
- Typical reasoning
  - Class membership
  - Equivalence of classes
  - Consistency
  - Classification

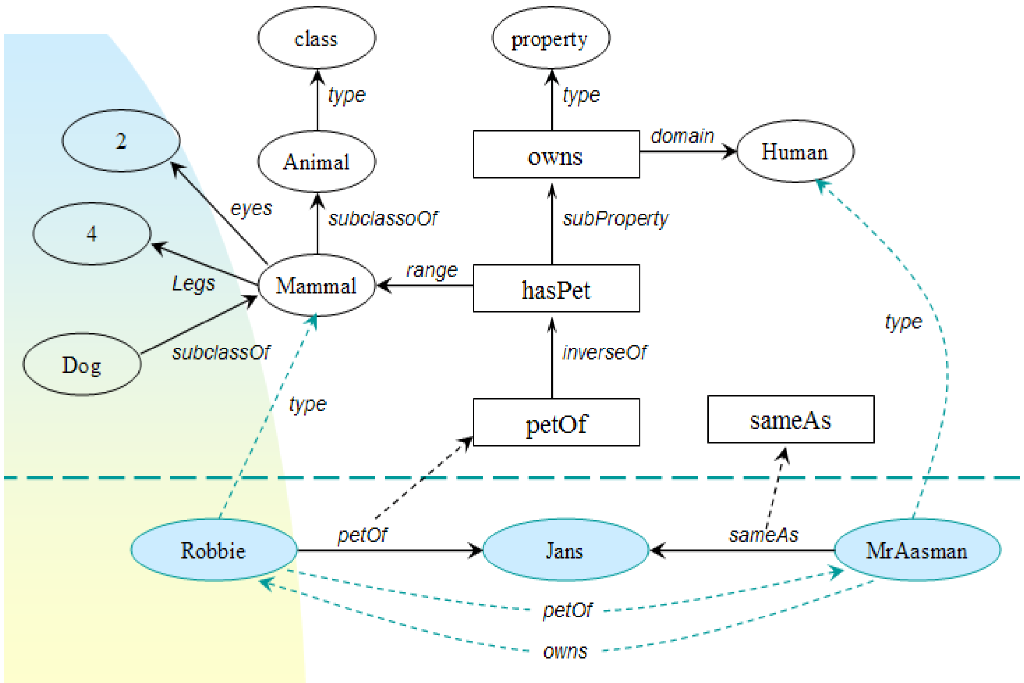
Owl language

- Richer description of objects
  - someValuesFrom (existential quantification), allValuesFrom (universal quantification), hasValue
  - Intersection, union, oneof
  - Cardinality (minCardinality, maxCardinality)
- Owl:sameAs
- owl:inverseOf
  - hasA inverseOf ownedBy
- owl:TransitiveProperty
  - greaterThan type TransitiveProperty
- owl:SymmetricProperty
  - siblingOf type SymmetricProperty
- owl:FunctionalProperty
  - only one value allowed: ex: age.
- owl:InverseFunctionalProperty
  - two different objects cannot have same value

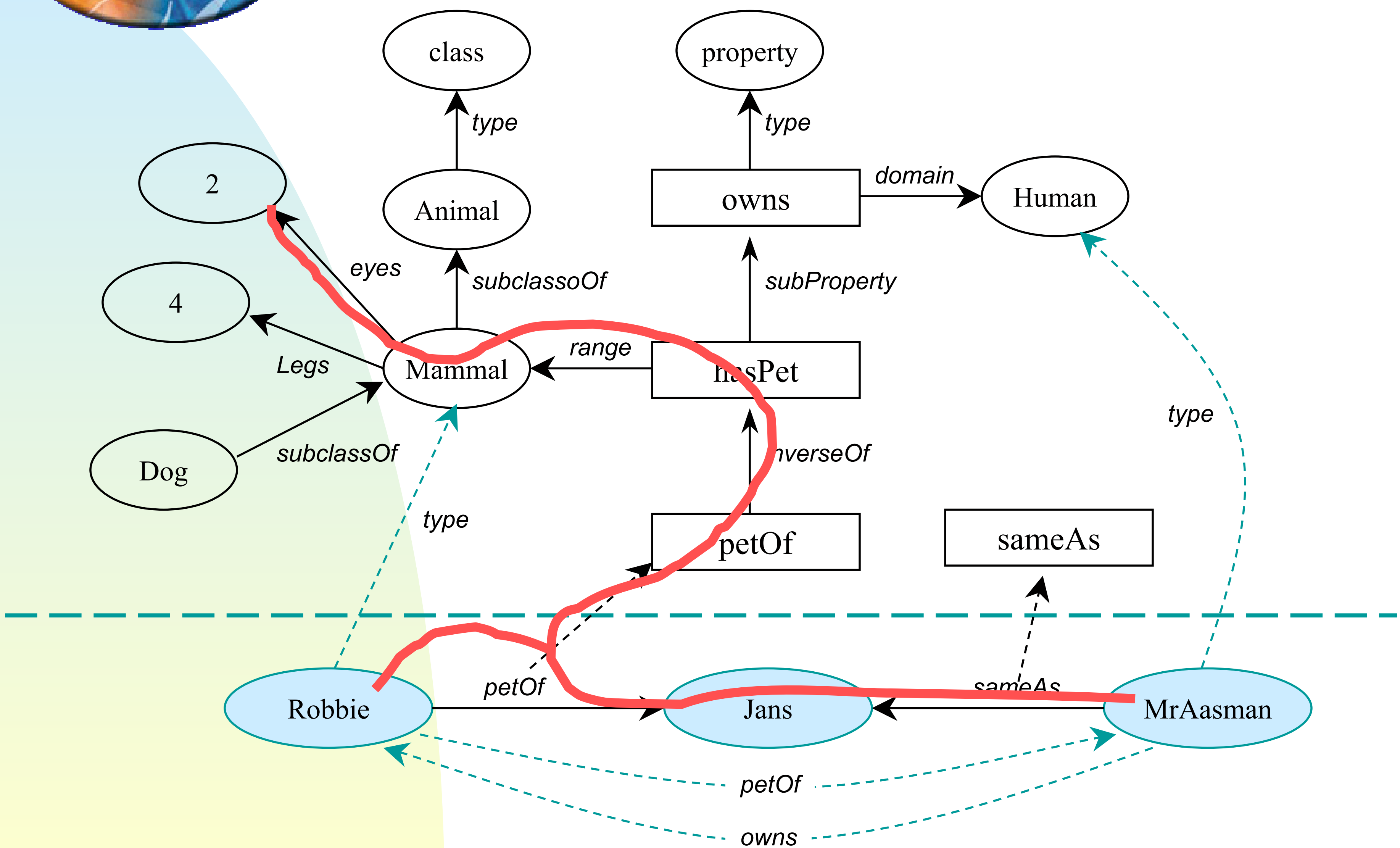
The power of RDFS/OWL: our example again

(query (MrAasman owns ?x)
(?x eyes 2))

- Animal type class
- Mammal subclassOf Animal
- Mammal eyes 2
- Mammal legs 4
- Dog subclassOf Mammal
- owns type Property
- owns domain Human
- hasPet subproperty owns
- hasPet range Mammal
- hasPet inverseOf petOf
- Robbie petOf Jans
- MrAasman sameAs Jans



Example of RDFS (and a little bit of OWL) Reasoning



Why an AllegroGraph reasoner?

- Full description logics
  - Good at handling (complex) ontologies
  - Complete but unpredictable time complexity when the number of individuals increase beyond millions
- Agraph does
  - All of RDFS
  - Most of OWL
  - Nearly complete but predictable, fast performance

What do we support in RDFS++

- See demo!

Future presentations

- July 16th:
  - TopBraidComposer with Agraph
  - Reasoning and Prolog with Agraph…
- September:
  - ->

Combining Geotemporal reasoning with social network analysis

(select (?x ?y)

(qs OsamaBinLaden controls ?x ? ?triple-id) → RDFS++ inference

(q CIA beliefs ?triple-id (> .8)) → Deification and Range Query

(q ?x is-at ?p1 ?time1) → Direct triple look up, time is named G

(after ?time1 “2001-07-28T0:0:0”) → Temporal primitive

(ego-group ?x 2 ?group) → Social networking analysis primitive

(member-of ?y ?group) → Plain prolog

(q ?y is-at ?p2 ?time2) → Direct triple look up, time is named G

(geodist-less ?p1 ?p2 12 kilometers) → Geospational primitive

(tempdist-less ?time1 ?time2 24 hours)) → Temporal primitive