
AWS
re:Invent
DECEMBER 1 – 5, 2025 | LAS VEGAS, NV

DAT443

Symbolic AI in the age of LLMs

Dr. Ora Lassila

Principal Technologist
Amazon Neptune

Co-chair
W3C RDF & SPARQL Working Group

SPEAKER INTRODUCTION

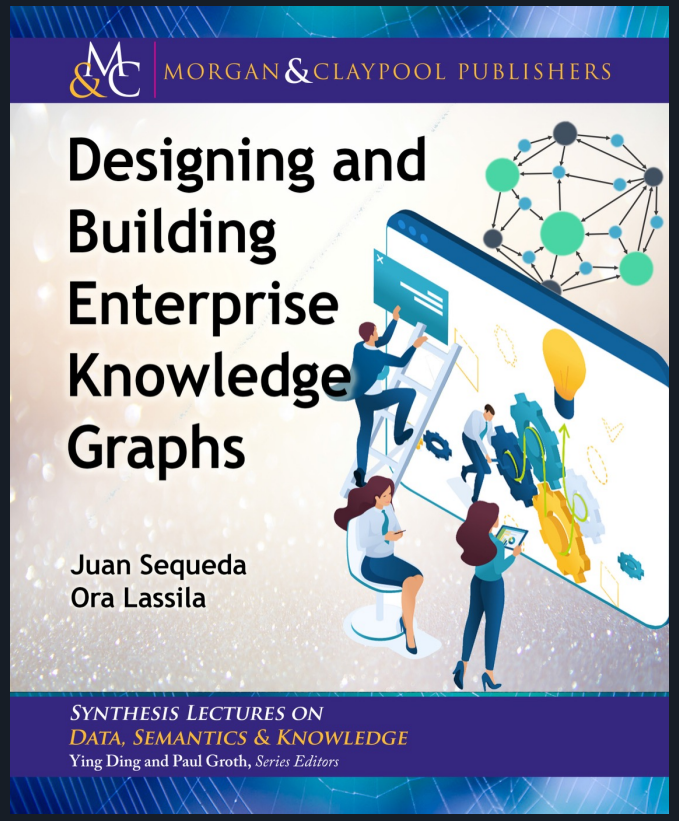
Ora Lassila

Ph.D (D.Sc) AI + CS,
Helsinki University of Technology

The “Semantic Web”

First RDF specification

This book →



My software flew the NASA Deep Space 1 probe

Agenda

Background

Very brief history of AI

Elements of symbolic AI

Symbolic AI today

What is a knowledge graph?

Working with ontologies

Working with reasoning

Symbolic AI & LLMs

Challenges

Hybrid systems

“Neuro-symbolic AI”

Disclaimer: This is not a tutorial on RDF, SPARQL, ontologies, reasoning, etc., but hopefully it will whet your appetite, and we can continue talking…

Background

Things you should know about AI

BRIEF HISTORY OF AI

Cycles of “boom & bust”

1940s: Connectionism

1950s: The Dartmouth Workshop of 1956
The term “Artificial Intelligence” is coined

1960s: The 1st AI Winter

1980s: Huge AI boom

1990s: The 2nd AI Winter

2000s: “Miraculous success” of **non-symbolic AI**
Also: revival of symbolic AI (Semantic Web)

BRIEF HISTORY OF AI

Notable parts of symbolic AI

Knowledge representation and reasoning

Theorem-proving

Heuristic search

Rule-based systems

Constraint solving

Planning

Agents

Programming languages for AI (Lisp, Prolog)

BRIEF HISTORY OF AI

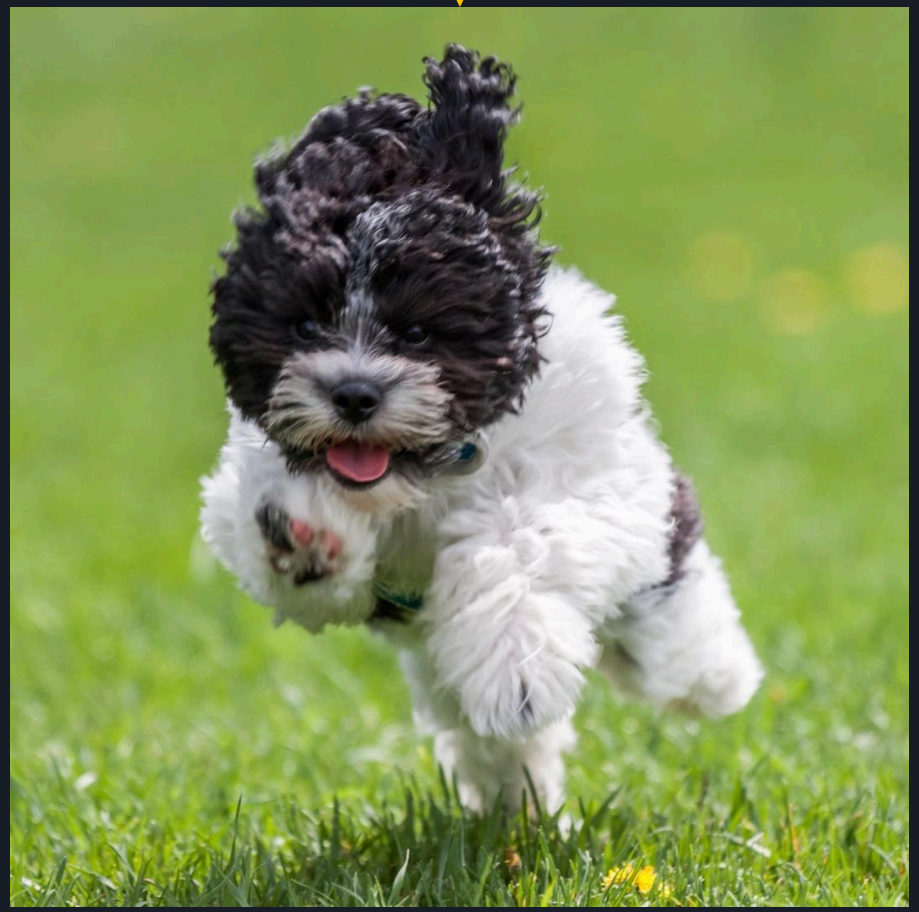
Symbolic reasoning?

We can represent dogs as a set: Dogs

Meet Wally: Wally ∈ Dogs

"Wally is a member of Dogs"

He is a Good Boy



$x \in C$

BRIEF HISTORY OF AI

Symbolic reasoning?

$x \in C \land C \subset D$

BRIEF HISTORY OF AI

Symbolic reasoning?

BRIEF HISTORY OF AI

Symbolic reasoning?

$$x \in C \land C \subset D \Rightarrow x \in D$$

“if x belongs in C, and C is a subset of D, then x belongs in D”

BRIEF HISTORY OF AI

Symbolic reasoning?

$x \in C \land C \subset D \Rightarrow x \in D$

"if x belongs in C, and C is a subset of D, then x belongs in D"

$Dogs \subset Mammals$

"Dogs is a subset of Mammals"

$Wally \in Dogs \Rightarrow Wally \in Mammals$

BRIEF HISTORY OF AI

Symbolic reasoning?

$x \in C \land C \subset D \Rightarrow x \in D$

"if x belongs in C, and C is a subset of D, then x belongs in D"

$Dogs \subset Mammals$

"Dogs is a subset of Mammals"

$Wally \in Dogs \Rightarrow Wally \in Mammals$

Symbolic AI today

Knowledge graphs and ontologies

SYMBOLIC AI TODAY

What is a knowledge graph?

Graph

- A structure of **nodes** and **edges**

Knowledge graph (KG)

- Nodes represent (real-world) **entities**
- Edges represent **relationships** between the entities

KGs are used to link and organize information

- Especially information of highly connected nature

KGs are the modern form of the “Semantic Web”

SYMBOLIC AI TODAY

Role of knowledge graphs

Primary purpose is to capture and maintain collective, organizational knowledge

Other key uses:

- Making knowledge accessible for humans and LLMs
- Data integration
- “Digital twins”

To build one, you need to model the information therein → you need an ontology

To store and manipulate one, you need a graph database → e.g., Amazon Neptune

SYMBOLIC AI TODAY

What is an ontology?

A logical model that captures the semantics of a domain

An ontology consists of

1. **concepts**	Dog
2. **properties** thereof	fur-color
3. **relations** in-between	owner
4. **constraints** that must hold	Dog ⋂ Cat = ∅
5. **individuals**	Wally

Ontologies are written in ontology languages:

RDF, OWL, SHACL

SYMBOLIC AI TODAY

What is semantics?

Informally, the set of definitions for software to interpret your data

1. Relationship of data to definitions
2. Relationship of data to other data
3. Hardwired in code

People have trouble separating their own interpretation from formal semantics

As a result, the expectations of how data can be interpreted (by a machine) are sometimes overly optimistic

…and we get *“JSON is easy to understand”*

SYMBOLIC AI TODAY

Example: “JSON is easy to understand”

```
{
    “first name”: “Ora”,
    “family name”: “Lassila”,
    “degree”: “Ph.D”,
    “place of birth”: “Helsinki”,
    “hobbies”: [ “photography”,
                 “scale models” ]
    “dogs”: [ “Wally”,
              “Coco” ]
}
```

SYMBOLIC AI TODAY

Example: “JSON is easy to understand”

Unless you read Finnish, this is what data looks like to machines

In the English version, ***you*** "grounded" the symbols

```
{
    “etunimi”: “Ora”,
    “sukunimi”: “Lassila”,
    “tutkinto”: “TkT”,
    “syntymäpaikka”: “Helsinki”,
    “harrastukset”: [ “valokuvaus”,
                      “pienoismallit” ]
    “koirat”: [ “Wally”,
                “Coco” ]
}
```

STRATEGIES & ADVICE

Working with ontologies

“Build or buy?”

WORKING WITH ONTOLOGIES

Your choices

1. **Use** an existing ontology to model your data

   Assess suitability carefully

WORKING WITH ONTOLOGIES

Your choices

1. **Use** an existing ontology to model your data

   Assess suitability carefully

2. **Extend** an existing ontology

   Understand what you are extending

   “Upper ontology” as a starting point?

   Every ontology makes decisions on how to model

WORKING WITH ONTOLOGIES

Your choices

1. **Use** an existing ontology to model your data

   Assess suitability carefully

2. **Extend** an existing ontology

   Understand what you are extending

   “Upper ontology” as a starting point?

   Every ontology makes decisions on how to model

3. **Define** your own ontology

   Prepare to do more work

WORKING WITH ONTOLOGIES

How to assess an ontology?

Use cases

Competency questions

Coverage

WORKING WITH ONTOLOGIES

How to assess an ontology?

Use cases

Competency questions

Coverage

Commitments

“Law of unintended consequences”

WORKING WITH ONTOLOGIES

How to assess an ontology?

Use cases

Competency questions

Coverage

Commitments

- “Law of unintended consequences”

Expressive power

- What do you need from the reasoner?
- Will it scale?

WORKING WITH ONTOLOGIES

Using ontologies in practice

Use ontologies to

1. Operationalize/automate processing of data
2. Uncover implicit information in data
3. Document your data
4. Establish a shared understanding of data

Stay away from closed, proprietary ontologies

Examples of shared public ontologies

Upper

Basic Formal Ontology (BFO)
very abstract

gist
practical, general purpose

schema.org
for annotating Web content

Examples of shared public ontologies

Upper

Basic Formal Ontology (BFO)
very abstract

gist
practical, general purpose

schema.org
for annotating Web content

Domain-specific

FIBO
financial concepts

SNOMED CT
clinical terminology

CIDOC-CRM
cultural heritage

Examples of shared public ontologies

Upper

Basic Formal Ontology (BFO)
very abstract

gist
practical, general purpose

schema.org
for annotating Web content

Narrow

Dublin Core (DC)
document metadata

PROV-O
provenance and lineage

DCAT
catalogs and dataset sharing

SKOS
taxonomies and thesauri

Domain-specific

FIBO
financial concepts

SNOMED CT
clinical terminology

CIDOC-CRM
cultural heritage

WORKING WITH ONTOLOGIES

Ontology example

```
:Dog a rdfs:Class ;
    rdfs:subClassOf :Mammal, :Pet ;
    owl:disjointWith :Cat .
```

WORKING WITH ONTOLOGIES

Ontology example

```
:Dog a rdfs:Class ;
    rdfs:subClassOf :Mammal, :Pet ;
    owl:disjointWith :Cat .

:Cat a rdfs:Class ;
    rdfs:subClassOf :Mammal, :Pet .

:Human a rdfs:Class ;
    rdfs:subClassOf :Mammal, foaf:Person .

:Mammal a rdfs:Class .

:Pet a rdfs:Class .
```

WORKING WITH ONTOLOGIES

Ontology example

```
:Dog a rdfs:Class ;
    rdfs:subClassOf :Mammal, :Pet ;
    owl:disjointWith :Cat .

:Cat a rdfs:Class ;
    rdfs:subClassOf :Mammal, :Pet .

:Human a rdfs:Class ;
    rdfs:subClassOf :Mammal, foaf:Person .

:Mammal a rdfs:Class .

:Pet a rdfs:Class .

:fur-color a owl:DatatypeProperty ;
    rdfs:domain :Dog ;
    rdfs:range xsd:string .

:owner a owl:ObjectProperty ;
    rdfs:domain :Pet ;
    rdfs:range :Human .
```

WORKING WITH ONTOLOGIES

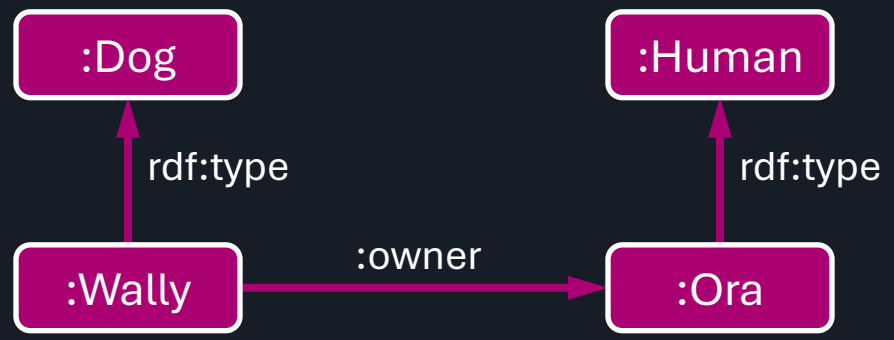
Ontology example

```
:Wally a :Dog ;
    foaf:name “Wally” ;
    :fur-color “white”, “black” ;
    :owner :Ora .

:Coco a :Dog ;
    foaf:name “Coco” ;
    :fur-color ”black” ;
    :owner :Ora .

:Ora a :Human ;
    foaf:name “Ora Lassila” ;
    foaf:homePage <https://www.lassila.org> .
```

WORKING WITH ONTOLOGIES

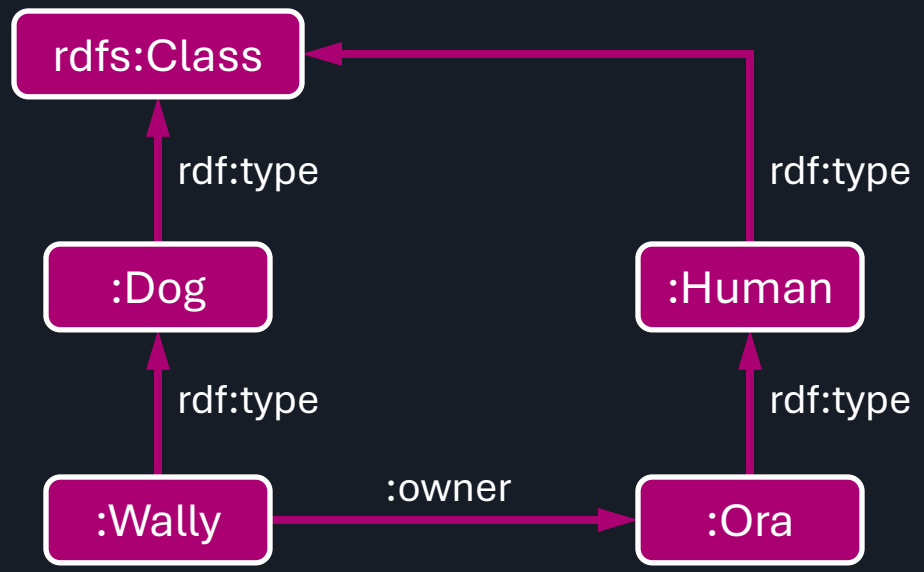
Ontologies as graphs

Wally and his owner as an RDF graph structure



WORKING WITH ONTOLOGIES

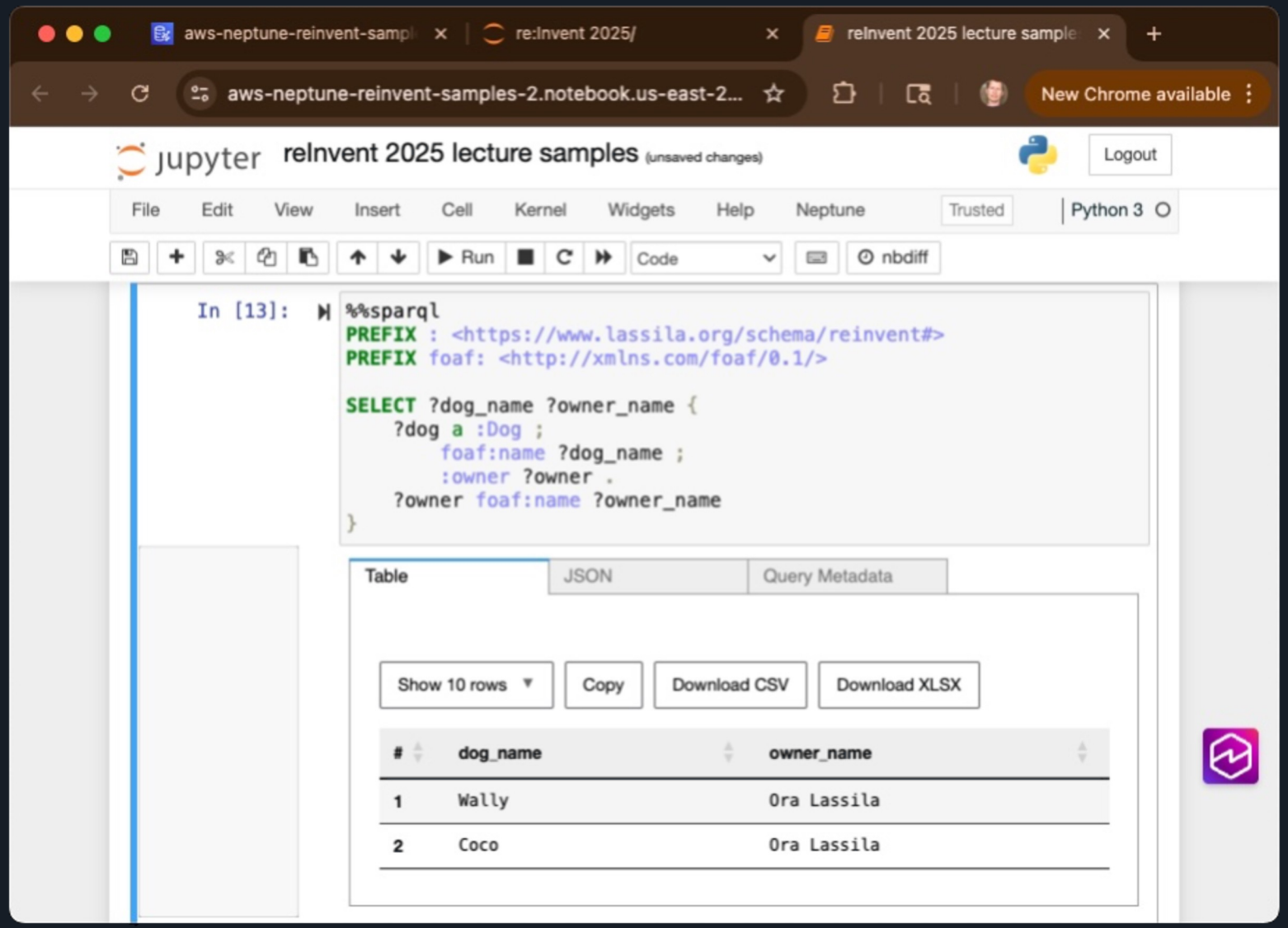
Ontologies as graphs



Querying Neptune in Jupyter Notebook

SPARQL query from a Jupyter Notebook cell

Results can be tabular or can be visualized as a graph



WORKING WITH ONTOLOGIES

Different ontology languages

Using RDF Schema

```
:Pet a rdfs:Class .

:owner a owl:ObjectProperty ;
    rdfs:domain :Pet ;
    rdfs:range :Human .
```

Domain and range are used to infer new data

```
:Wally a :Dog ;
    :owner :Ora .
```

The reasoner can infer that :Ora is a :Human

WORKING WITH ONTOLOGIES

Different ontology languages

Using OWL

```
:Pet a rdfs:Class ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty :owner ;
        owl:cardinality 1 ;
        owl:allValuesFrom :Human
    ] .
```

A :Pet must have exactly one :owner and it must be a :Human

Restrictions are introduced as super-classes

Unsatisfied restrictions → report inconsistencies or infer new information

WORKING WITH ONTOLOGIES

Different ontology languages

Using SHACL

```
:Pet a rdfs:Class, sh:NodeShape ;
    sh:property [
        sh:path :owner ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:class :Human
    ] .
```

A :Pet must have exactly one :owner and it must be a :Human

Constraints are introduced as “shapes”

Unsatisfied constraints → report violations

SHACL does not infer new data

WORKING WITH ONTOLOGIES

World assumptions

Why did RDF and OWL reasoners produce new information, but SHACL did not?

Closed World Assumption (CWA)

Reasoning is based on what we know

Negation-by-failure:
"What cannot be proven to be true, is false"

Used by databases

Also: Unique Name Assumption

WORKING WITH ONTOLOGIES

World assumptions

Why did RDF and OWL reasoners produce new information, but SHACL did not?

Closed World Assumption (CWA)

- Reasoning is based on what we know
- Negation-by-failure:
  "What cannot be proven to be true, is false"
- Used by databases
- Also: Unique Name Assumption

Open World Assumption (OWA)

- We assume we do not know everything
- Negation-by-failure cannot be used
- Used on the Semantic Web

WORKING WITH ONTOLOGIES

Understanding modeling commitments

We defined that a :Pet can only have one :owner

What about joint ownership?

Sometimes, limitations like this pop up way later in your project

Solutions to this problem:

- If you control the upper definitions, you have some refactoring to do
- If you don’t, maybe introduce :co-owner?

WORKING WITH ONTOLOGIES

Class hierarchy vs. concept taxonomy

When using classes, you can define variations as a class hierarchy

Alternatively, keep the number of classes at minimum, and use a “concept scheme” to provide information about variations

The SKOS vocabulary can be very helpful here

Example: What if I want to model dog breeds?

WORKING WITH ONTOLOGIES

Class hierarchy example

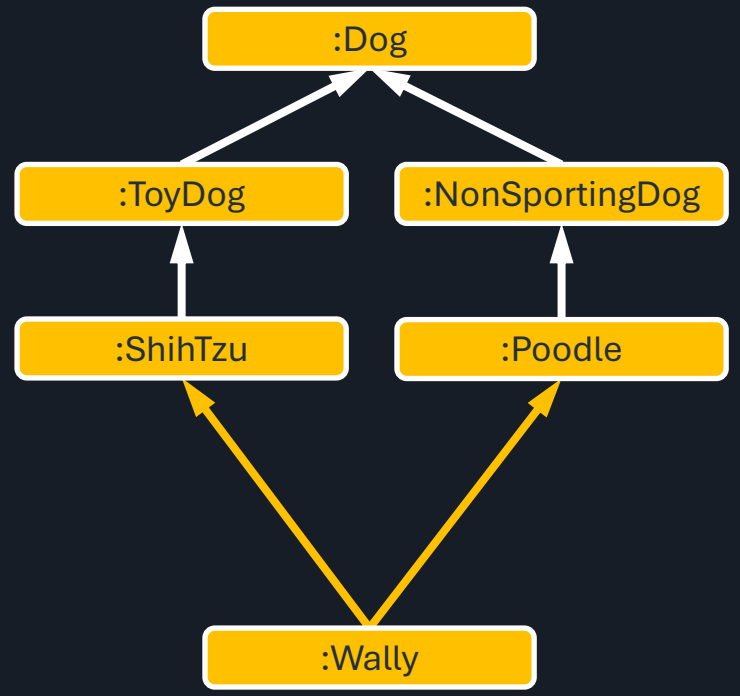
```
:Dog a rdfs:Class .

:ToyDog a rdfs:Class ; rdfs:subClassOf :Dog .

:ShihTzu a rdfs:Class ; rdfs:subClassOf :ToyDog .

:NonSportingDog a rdfs:Class ; rdfs:subClassOf :Dog .

:Poodle a rdfs:Class ; rdfs:subClassOf :NonSportingDog .


:Wally a :ShihTzu, :Poodle .
```

WORKING WITH ONTOLOGIES

Class hierarchy example

```
:Dog a rdfs:Class .

:ToyDog a rdfs:Class ; rdfs:subClassOf :Dog .

:ShihTzu a rdfs:Class ; rdfs:subClassOf :ToyDog .

:NonSportingDog a rdfs:Class ; rdfs:subClassOf :Dog .

:Poodle a rdfs:Class ; rdfs:subClassOf :NonSportingDog .
```

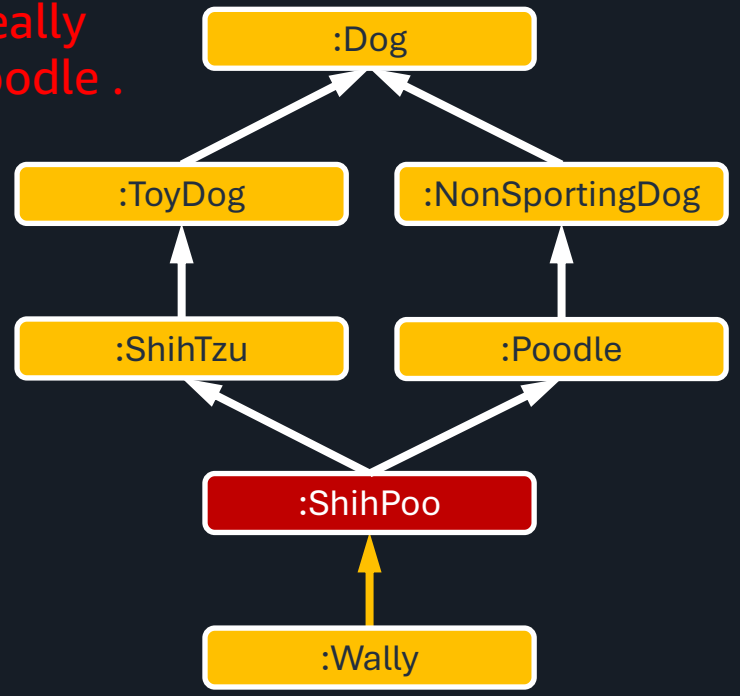
```
:Wally a :ShihTzu, :Poodle .
```



WORKING WITH ONTOLOGIES

Class hierarchy example

```
:Dog a rdfs:Class .

:ToyDog a rdfs:Class ; rdfs:subClassOf :Dog .

:ShihTzu a rdfs:Class ; rdfs:subClassOf :ToyDog .

:NonSportingDog a rdfs:Class ; rdfs:subClassOf :Dog .

:Poodle a rdfs:Class ; rdfs:subClassOf :NonSportingDog .

:ShihPoo a rdfs:Class ;  # yep, really
    rdfs:subClassOf :ShihTzu, :Poodle .

:Wally a :ShihPoo .
```



WORKING WITH ONTOLOGIES

Concept taxonomy example

```
:Dog a rdfs:Class .

:breed a owl:ObjectProperty ;
    rdfs:domain :Dog ;
    rdfs:range skos:Concept .
```

WORKING WITH ONTOLOGIES

Concept taxonomy example

```
:Dog a rdfs:Class .

:breed a owl:ObjectProperty ;
    rdfs:domain :Dog ;
    rdfs:range skos:Concept .

:DogBreed a skos:ConceptScheme ;
    skos:topConcept :ToyDog, :NonSportingDog, …

:ToyDog a skos:Concept .

:ShihTzu a skos:Concept ; skos:broader :ToyDog .

:NonSportingDog a skos:Concept.

:Poodle a skos:Concept ; skos:broader :NonSportingDog .


:Wally a :Dog ;
    :breed :ShihTzu, :Poodle .
```

WORKING WITH ONTOLOGIES

Concept taxonomy example

```
:Dog a rdfs:Class .

:breed a owl:ObjectProperty ;
    rdfs:domain :Dog ;
    rdfs:range skos:Concept .

:DogBreed a skos:ConceptScheme ;
    skos:topConcept :ToyDog, :NonSportingDog, …

:ToyDog a skos:Concept .

:ShihTzu a skos:Concept ; skos:broader :ToyDog .

:NonSportingDog a skos:Concept.

:Poodle a skos:Concept ; skos:broader :NonSportingDog .

:ShihPoo a skos:Concept ; skos:broader :ShihTzu, :Poodle .

:Wally a :Dog ;
    :breed :ShihPoo .
```

STRATEGIES & ADVICE

Working with (symbolic) reasoning

Ph.D in mathematical logic not required…

WORKING WITH REASONING

Reasoning use cases

1. Uncover **implicit** information

   :Wally a :ShihPoo ⇒ :Wally a :Dog

   :Wally a :ShihPoo ⇒ :Wally a :Mammal

WORKING WITH REASONING

Reasoning use cases

1. Uncover **implicit** information

   :Wally a :ShihPoo ⇒ :Wally a :Dog

   :Wally a :ShihPoo ⇒ :Wally a :Mammal

2. Discover **inconsistencies** or other errors

   :Wally a :Dog, :Cat
   → inconsistency: :Dog and :Cat are disjoint

WORKING WITH REASONING

Reasoning use cases

1. Uncover **implicit** information

   :Wally a :ShihPoo ⇒ :Wally a :Dog

   :Wally a :ShihPoo ⇒ :Wally a :Mammal

2. Discover **inconsistencies** or other errors

   :Wally a :Dog, :Cat
   → inconsistency: :Dog and :Cat are disjoint

3. Make **query authoring** easier

   SELECT ?dog { ?dog a :Dog }
   would select :Wally

   Using reasoning allows you to insulate queries from ontology changes

WORKING WITH REASONING

Using RDF Schema

Taxonomical reasoning (mostly)

Class hierarchies: OO modeling

Property hierarchies:

- Useful when extending ontologies
- :p2 rdfs:subPropertyOf :p1 .
- :a :p2 :b ⇒ :a :p1 :b

Example:

- dc:title rdfs:subPropertyOf rdfs:label
- Querying for values of rdfs:label will also give you values of dc:title

WORKING WITH REASONING

Using OWL

With OWL, you can do more sophisticated things

```
:BlackPoodle a owl:Class ;
    owl:EquivalentClass [
        a owl:Class ;
        owl:intersectionOf (
            [
                a owl:Restriction ;
                owl:onProperty :breed ;
                owl:hasValue :Poodle
            ]
            [
                a owl:Restriction ;
                owl:onProperty :fur-color ;
                owl:hasValue “black”
            ]
        )
    ] .
```

WORKING WITH REASONING

Using OWL

```
:BlackPoodle a owl:Class ;
    owl:EquivalentClass [
        a owl:Class ;
        owl:intersectionOf (
            [
                a owl:Restriction ;
                …
```

why not subClassOf?

rdfs:subClassOf → necessary condition

Every class must satisfy

Converse not true

owl:equivalentClass → necessary and sufficient

If conditions are met, we can infer the class membership

WORKING WITH REASONING

How and when to reason?

Some “triple stores” have a built-in reasoner; if not, you have various options…

1. **“Server-side” reasoning**

   Reasoner is like a cache in front of your database

2. **“Client-side” reasoning**

   Retrieve some data, perform local reasoning

3. **Query rewriting**

   SPARQL can be used as a “reasoning engine”

WORKING WITH REASONING

SPARQL as a reasoning engine

Find the classes of an instance:

with a reasoner:

```
SELECT ?instance ?class {
   ?instance a ?class
}
```

without a reasoner:

```
SELECT ?instance ?class {
   ?instance a ?direct_class .
   ?direct_class rdfs:subClassOf* ?class
}
```

```
SELECT ?instance ?class {
   ?instance rdf:type/rdfs:subClassOf* ?class
}
```

(Taxonomical reasoning only)

WORKING WITH REASONING

SPARQL as a reasoning engine

You can materialize results in advance

```
INSERT {
   ?instance a ?class
}
WHERE {
   ?instance rdf:type/rdfs:subClassOf* ?class
}
```

or better:

```
INSERT {
   GRAPH :entailments {
      ?instance a ?class
   }
}
WHERE {
   ?instance rdf:type/rdfs:subClassOf* ?class
}
```

WORKING WITH REASONING

SPARQL as a reasoning engine

Choice of when reasoning happens depends on the nature of your use case

- Data changes a lot: reason at query time
- Data is relatively static: reason in advance

WORKING WITH REASONING

SPARQL as a reasoning engine

Choice of when reasoning happens depends on the nature of your use case

- Data changes a lot: reason at query time
- Data is relatively static: reason in advance

Materializing? → you will need “truth maintenance”

- How to invalidate entailments when data changes?
- Exactly which entailments are invalidated?
- “Poor man’s truth maintenance”: materialize into a separate named graph; drop it and recompute when necessary

Symbolic AI and LLMs

How can we help the “new AI”…?

“Any sufficiently advanced technology is indistinguishable from magic.”

Arthur C. Clarke’s 3rd Law, 1962

LOOKING INTO THE FUTURE

Challenges

Hallucinations

LLMs will not say “I don’t know”… instead, they will make stuff up

(Note: This is a feature, not a bug)

LOOKING INTO THE FUTURE

Challenges

Hallucinations

LLMs will not say “I don’t know”… instead, they will make stuff up

(Note: This is a feature, not a bug)

Anthropomorphism

LLM answers sound convincing, and people will easily start thinking of LLMs as actual persons

LOOKING INTO THE FUTURE

Challenges

Hallucinations

LLMs will not say “I don’t know”… instead, they will make stuff up

(Note: This is a feature, not a bug)

Anthropomorphism

LLM answers sound convincing, and people will easily start thinking of LLMs as actual persons

Computational efficiency

Asking an LLM for anything you already know how to compute is not efficient

Our energy supply is finite

LOOKING INTO THE FUTURE

Historical criticism of AI

Joseph Weizenbaum: “ELIZA” (1966)

Anthropomorphizing of computers

John Searle: “Chinese room experiment” (1980)

What is consciousness? Or “understanding”?

These are relevant today!

LOOKING INTO THE FUTURE

Scarcities?

Historically, scarcities of “ability and capability”

- Algorithms
- Memory & CPU

What scarcities do we experience today?

- Correct answers
- Trustworthiness and accountability
- Explainability
- Energy

SUMMARY

Symbolic AI to the rescue?

Correct answers

- Use knowledge graphs as the source of answers

SUMMARY

Symbolic AI to the rescue?

Correct answers

- Use knowledge graphs as the source of answers

Trustworthiness and accountability

- You want to know where answers are coming from
- Knowledge graphs can be curated and audited

SUMMARY

Symbolic AI to the rescue?

Correct answers

- Use knowledge graphs as the source of answers

Trustworthiness and accountability

- You want to know where answers are coming from
- Knowledge graphs can be curated and audited

Explainability

- This is a prominent feature of symbolic reasoning

SUMMARY

Symbolic AI to the rescue?

Correct answers

Use knowledge graphs as the source of answers

Trustworthiness and accountability

You want to know where answers are coming from

Knowledge graphs can be curated and audited

Explainability

This is a prominent feature of symbolic reasoning

Energy

Getting an answer from a knowledge graph is orders of magnitude more efficient than from an LLM

LOOKING INTO THE FUTURE

Hybrid architectures?

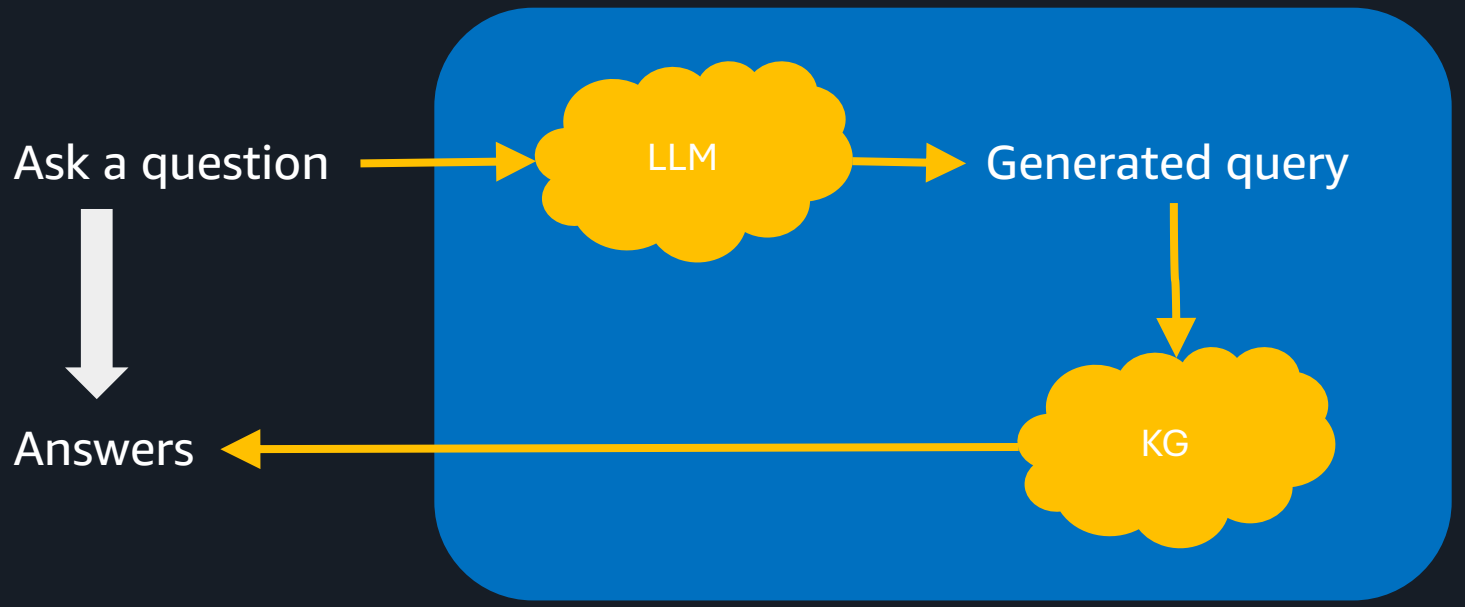
Growing understanding that we need hybrid architectures for intelligent systems

“Dual process models” of cognition

- Fast & unconscious vs. slow & conscious
- “System 1” and “System 2” [Stanovich & West, …]
- “Thinking, Fast and Slow” [Kahneman]
- “Architecture” of intelligence and consciousness [Sloman, Dennett, …]

LOOKING INTO THE FUTURE

Using an LLM with a KG?

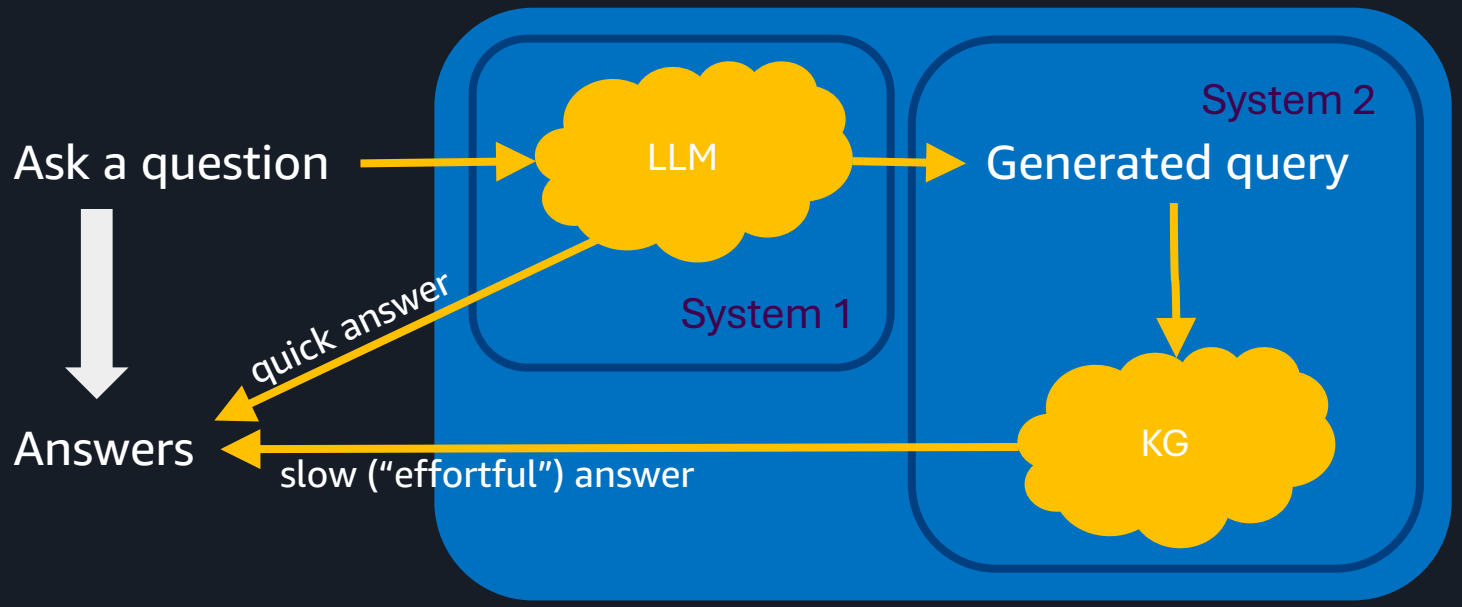


Not just queries against a KG, but also arbitrarily complex symbolic reasoning

KG provides the necessary grounding for the LLM

LOOKING INTO THE FUTURE

Dual process model with an LLM and a KG?



This is a highly trivialized view

Other architectures and models also make sense

LOOKING INTO THE FUTURE

But wait, isn’t this just ”graph RAG”?

Not exactly…

Graph RAG uses a graph to improve LLM performance

What if we instead used an LLM to improve knowledge graph performance or usability?

LOOKING INTO THE FUTURE

What is “neuro-symbolic AI”?

Symbolic AI

Often seen as **idealistic and unrealistic**

BUT

Representations are **transparent**

Results **can be explained**

LOOKING INTO THE FUTURE

What is “neuro-symbolic AI”?

Symbolic AI	Non-symbolic AI
Often seen as **idealistic and unrealistic**	Can easily deal with **“real world” data**
BUT	*BUT*
Representations are **transparent**	Representations are **opaque**
Results **can be explained**	Results **cannot be explained**

Can we leverage the best features of both worlds?

LOOKING INTO THE FUTURE

What is “neuro-symbolic AI”?

Not a single architecture, algorithm, or even methodology

Ways to combine symbolic and non-symbolic AI

1. **Symbolic + non-symbolic:**

   Heuristic search + neural evaluation function

   “System 1” & “System 2” –architecture

2. **Non-symbolic → symbolic**

   Non-symbolic input converted to symbolic representation and processed via a symbolic reasoner

3. **Symbolic → non-symbolic**

   Symbolic rules as training data

Final words

SUMMARY

Symbolic AI

AI has been around for quite some time

Many useful techniques and technologies

Once well understood, these “stop being AI”

Dramatic shift from symbolic to non-symbolic AI

Many symbolic AI techniques are relevant today

KGs and ontologies moving into the mainstream

SUMMARY

Ontologies and reasoning

Ontologies will help you operationalize processing of data

They will also serve as documentation

Reasoning can simplify things

You can do this

SUMMARY

Symbolic AI and LLMs

Symbolic AI techniques can help generative AI

We can mitigate many of the challenges of generative AI with mature symbolic AI techniques

- Ontologies and knowledge graphs
- Planning and agents

Going forward, we will see more hybrid systems

- Many ways to combine symbolic and non-symbolic AI

Questions?

Please complete the session survey in the mobile app

Thank you! Kiitos!

More questions?

I can be reached at ora@amazon.com

Some interesting background material

Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider: *The Description Logic Handbook* (2003)

Brachman & Levesque: *Readings in Knowledge Representation and Reasoning* (1985)

Berners-Lee, Hendler & Lassila: "The Semantic Web", Scientific American 284(5) (May 2001)

Jennings & Wooldridge: *Agent Technology: Foundations, Applications, and Markets* (1998)

Kautz: “The Third AI Summer”, AI Magazine 43(1) (Spring 2022)

Minsky: *A Framework for Representing Knowledge* (1974)

Nilsson: “Artificial Intelligence Prepares for 2001”, AI Magazine 4(4) (December 1983)

Uschold & Gruninger: *Ontologies: Principles, Methods and Applications* (1996)

Please complete the session survey in the mobile app