Knowledge Representation & Reasoning: From Foundations to Products

Ian Horrocks

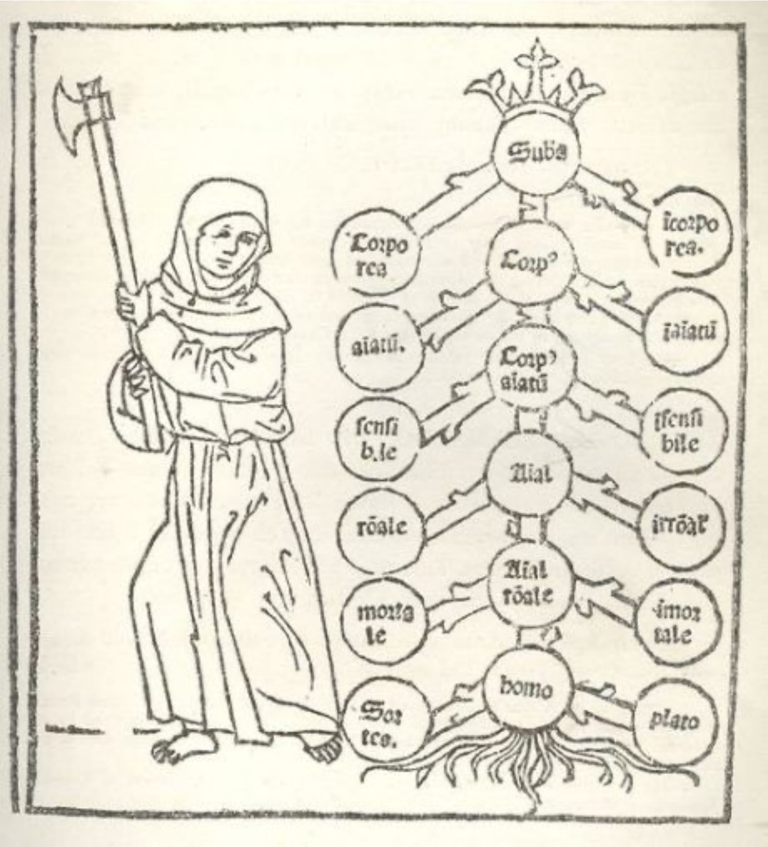
UNIVERSITY OF OXFORD

DEPARTMENT OF COMPUTER SCIENCE

Background and Motivation

KRR Historical Perspective

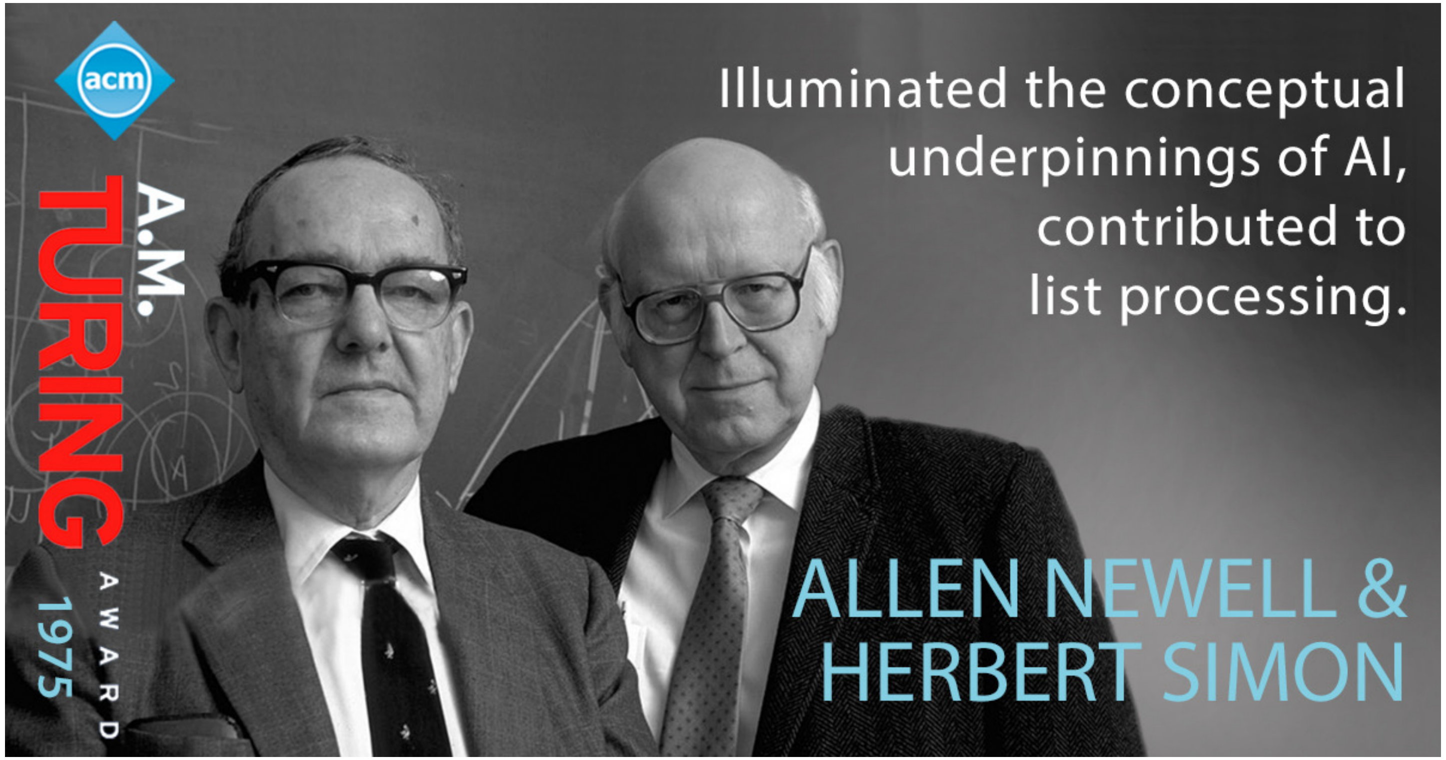
- Long and distinguished history



Porphyry’s depiction of Aristotle’s categories

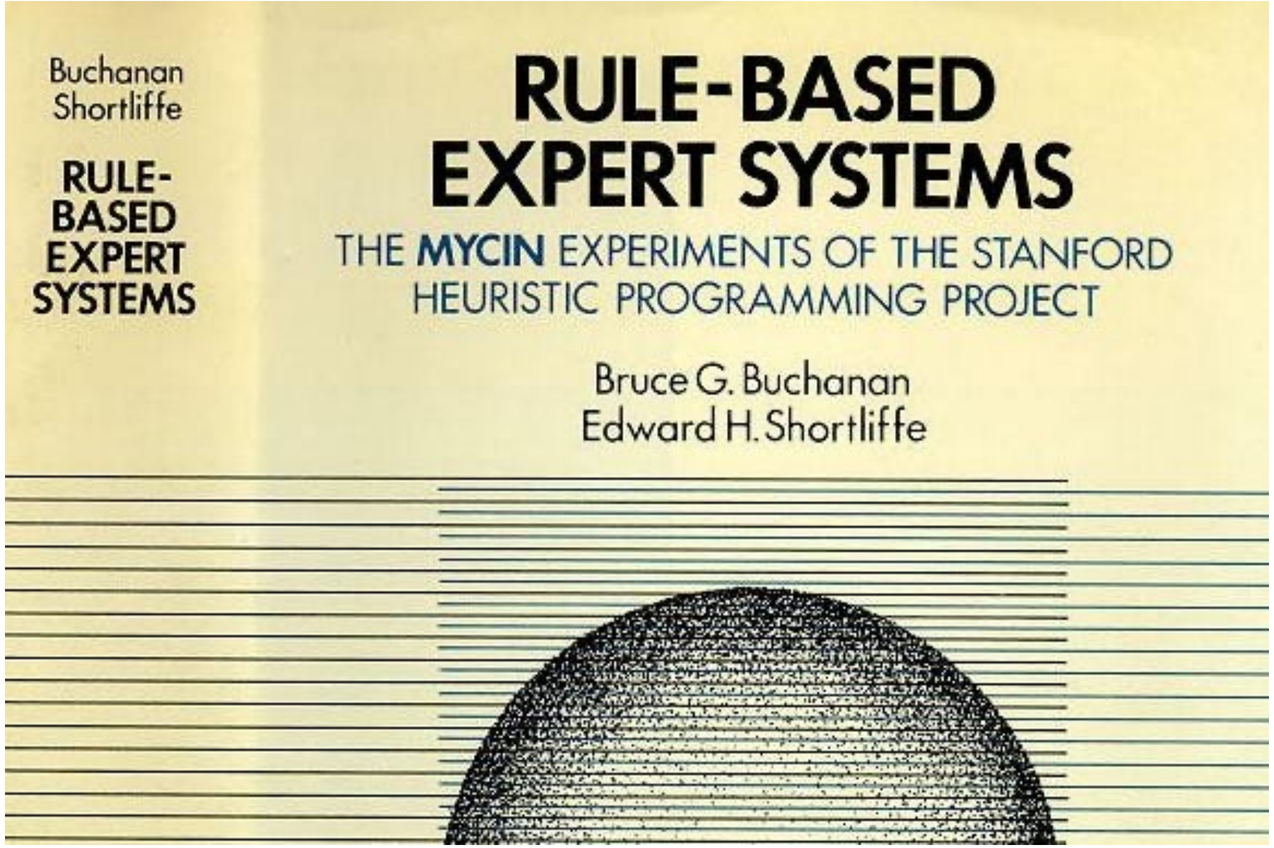
KRR Historical Perspective

- Long and distinguished history
- General Problem Solver (GPS)

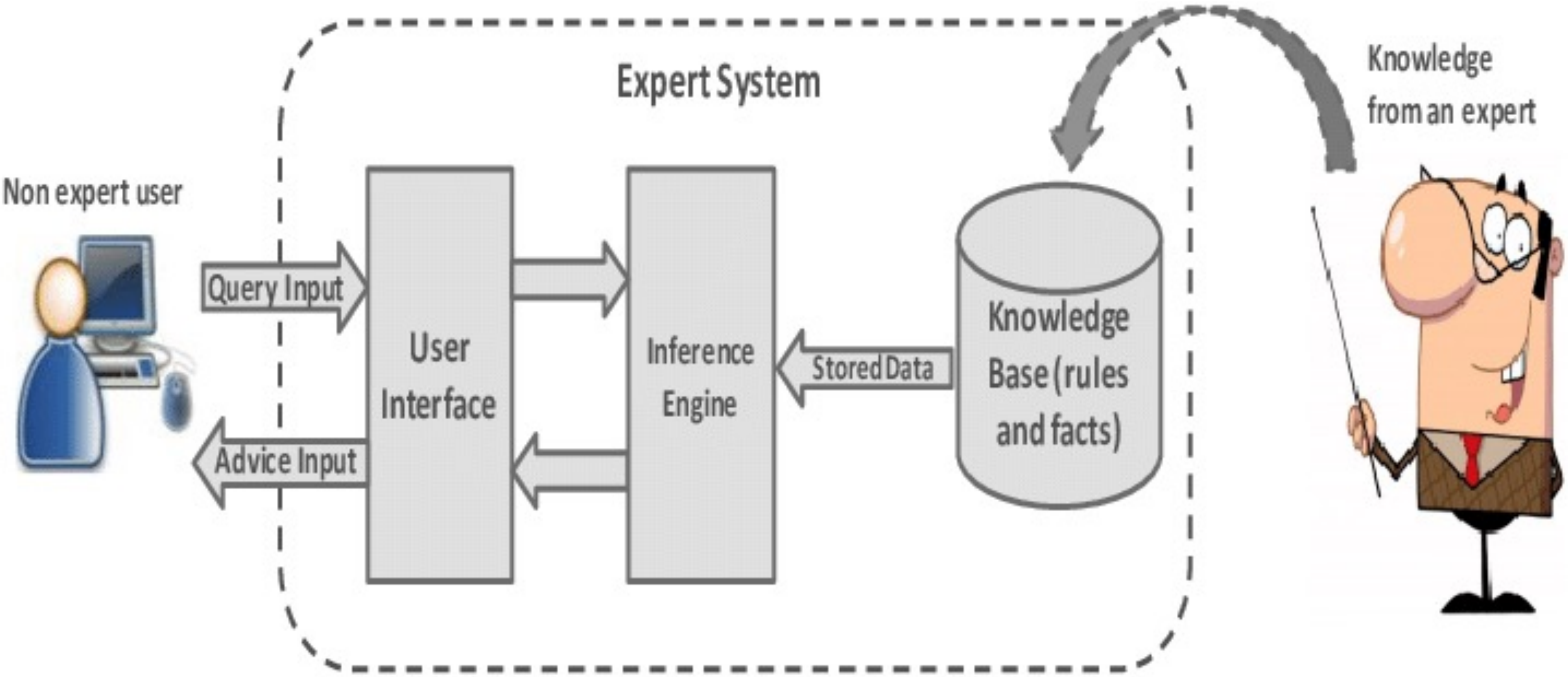


KRR Historical Perspective

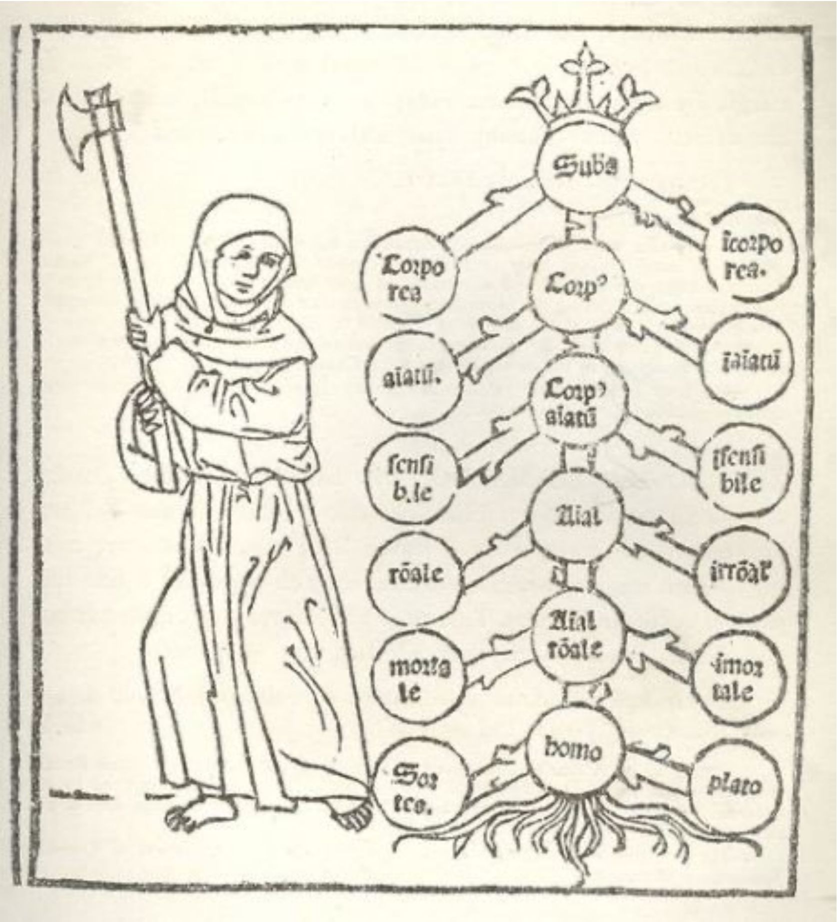
- Long and distinguished history
- General Problem Solver (GPS)
- Expert systems

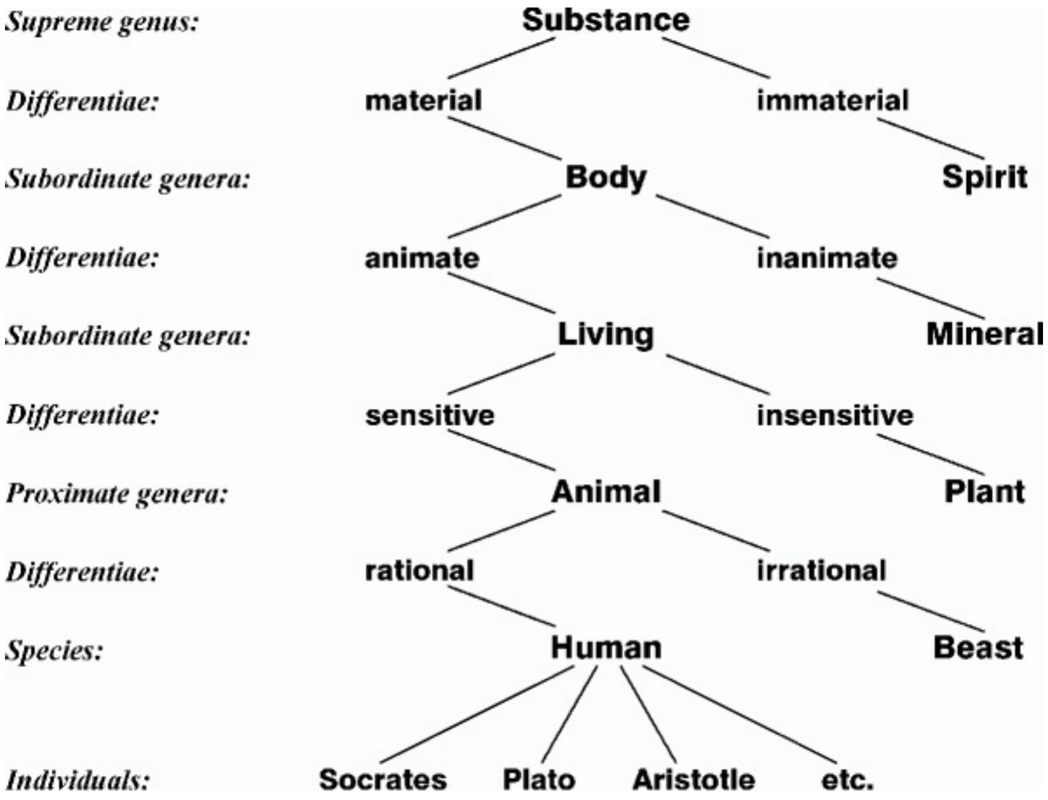


KRR Historical Perspective

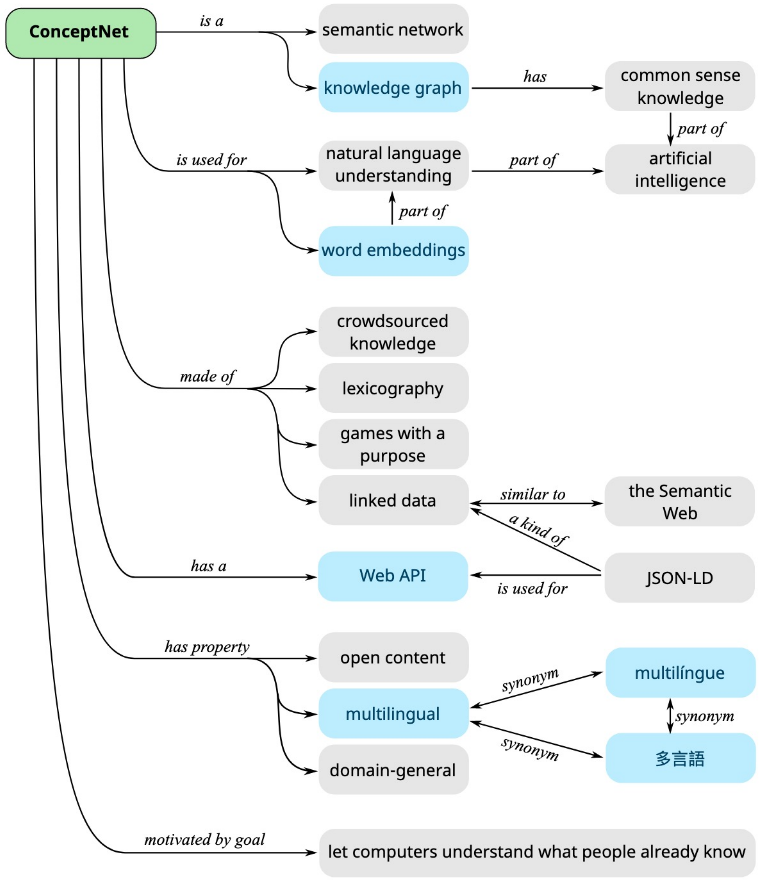


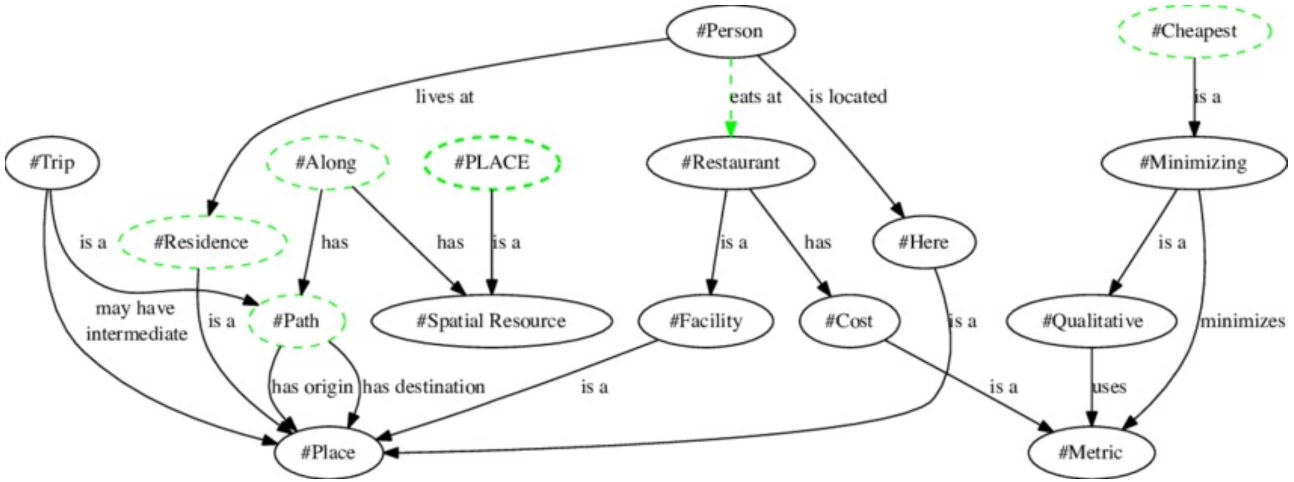
Why use a graph?



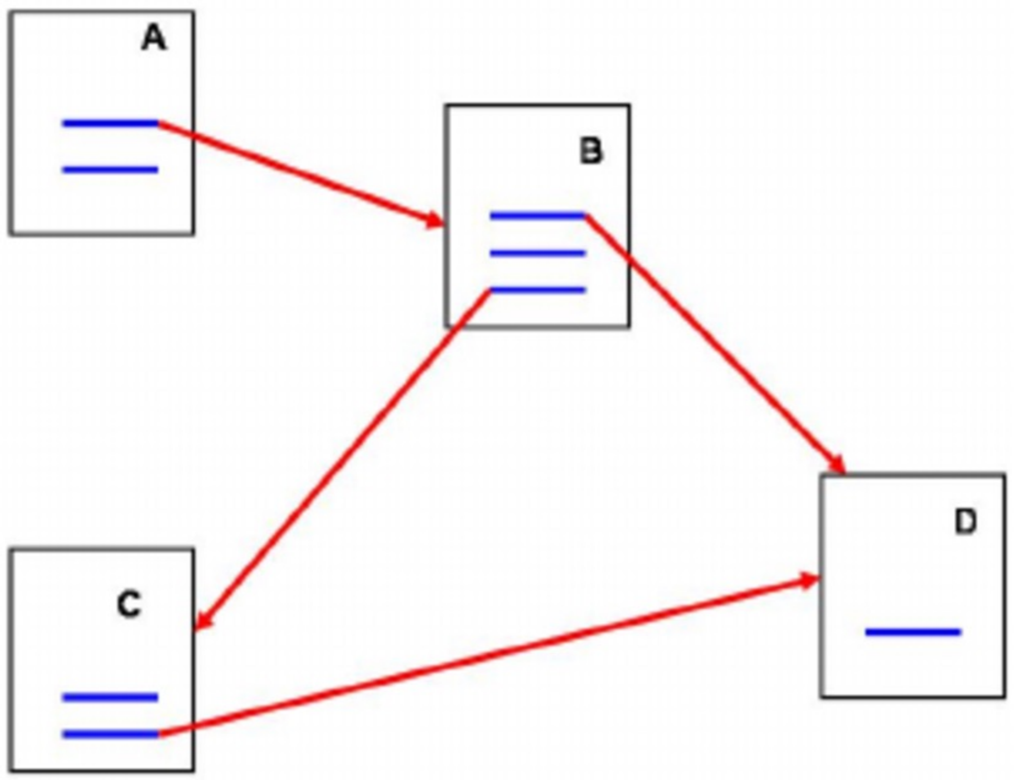


Why use a graph?



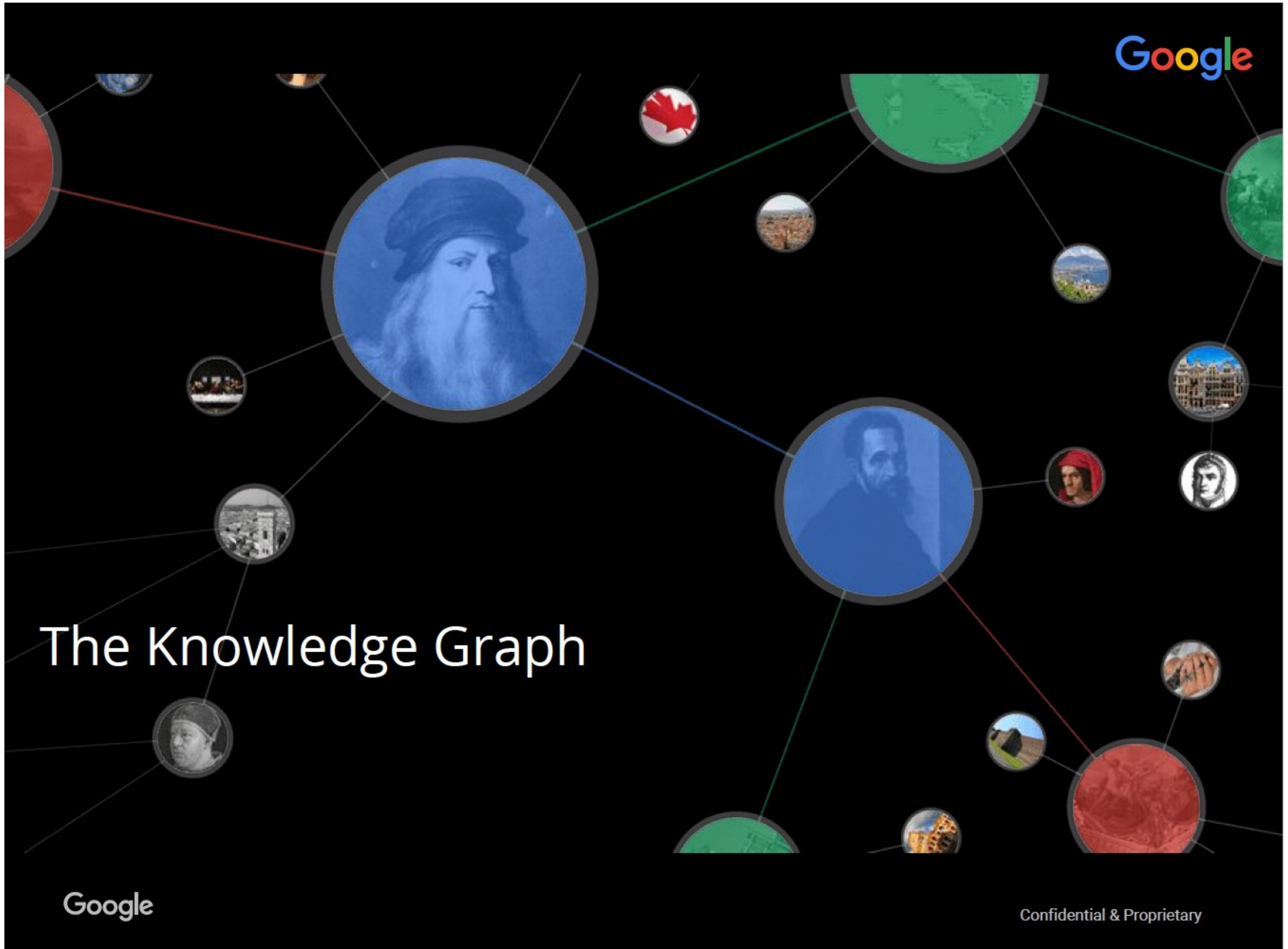


Why use a graph?



Why use a graph?

Why use a graph?



Why use a graph?

alexa

WIKIDATA

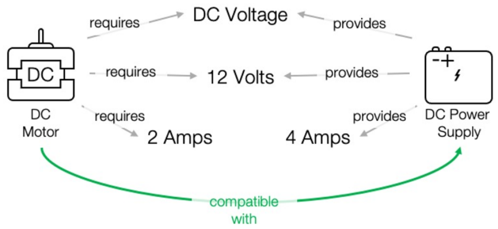
DBpedia

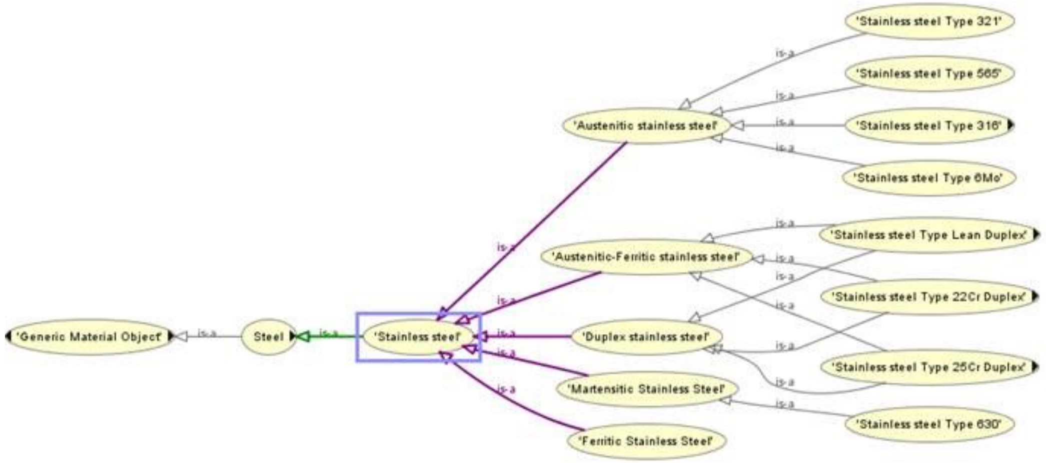
ebay Beam open source

A Distributed Knowledge Graph Store

FIBO

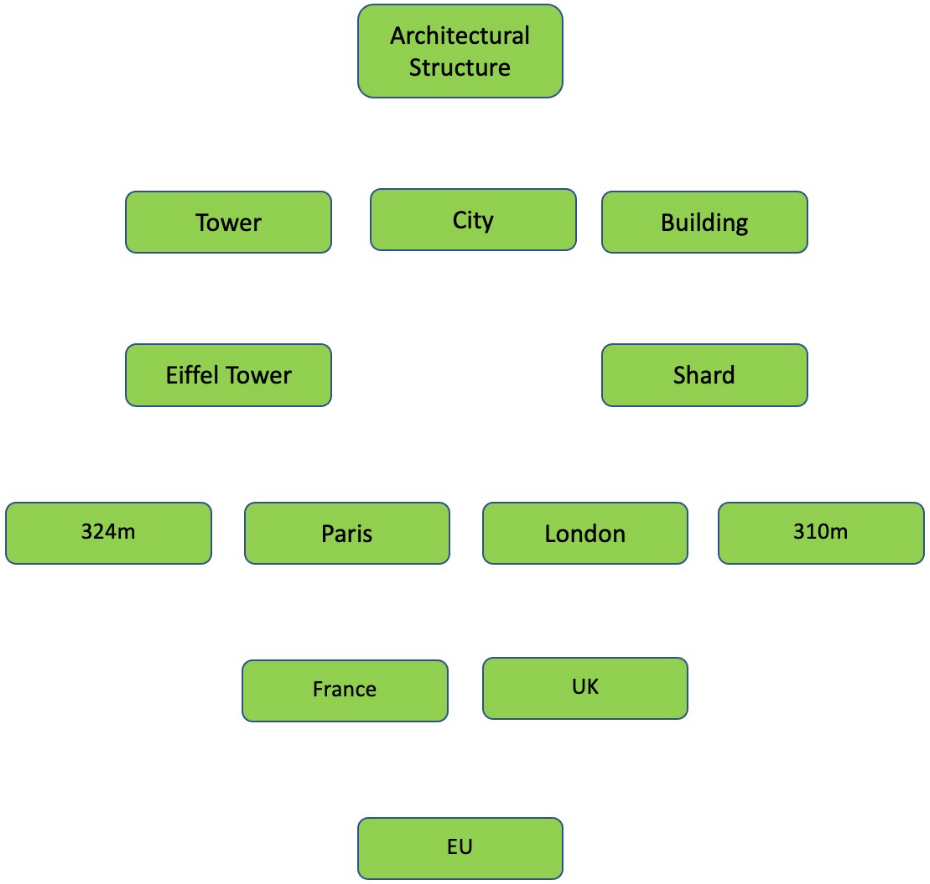
FESTO



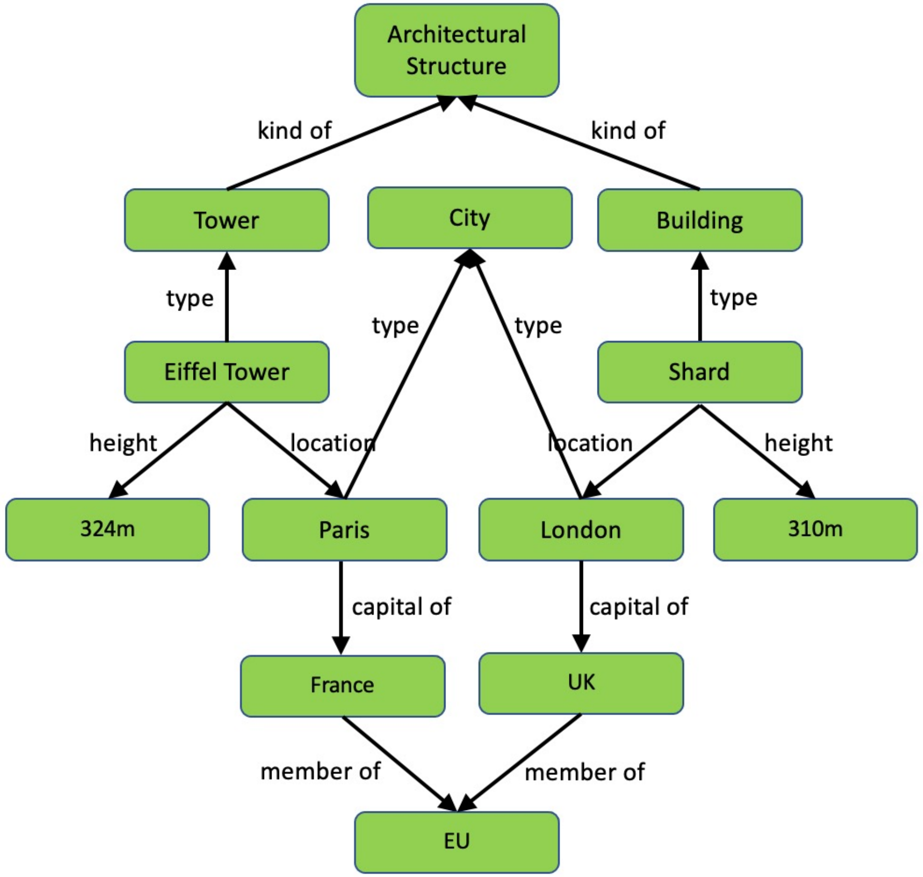


Introduction to Knowledge Graphs

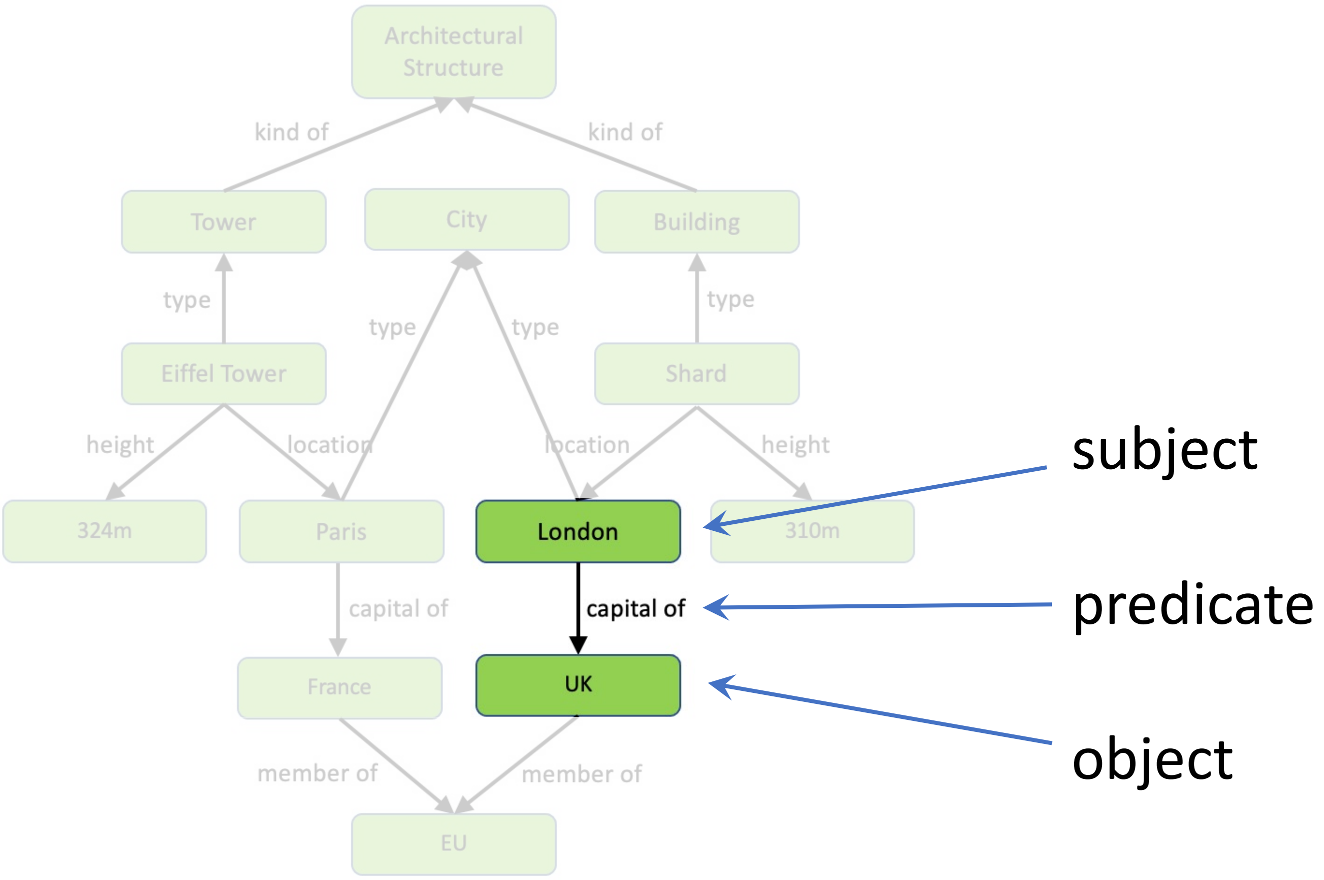
Anatomy of a Knowledge Graph



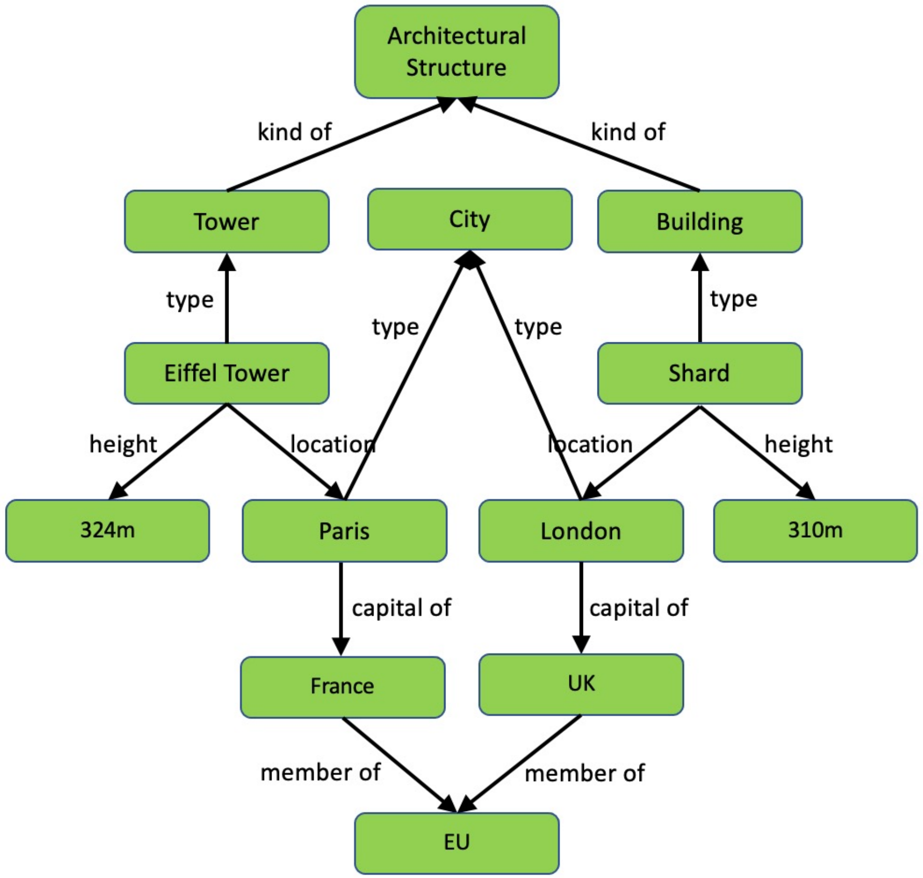
Anatomy of a Knowledge Graph



Anatomy of a Knowledge Graph



Anatomy of a Knowledge Graph



~

Architectural Structure		
Name	Height	Location
Eiffel Tower	324	Paris
Shard	310	London

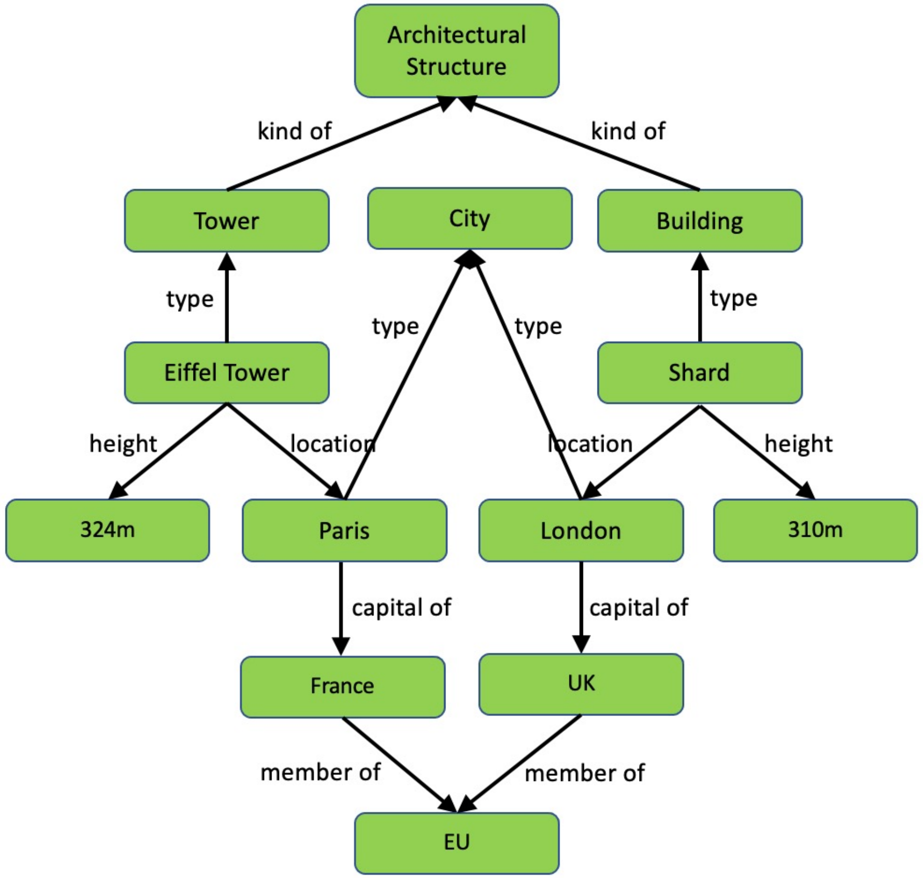
Tower
Name
Eiffel Tower

Building
Name
Shard

City	
Name	Capital Of
Paris	France
London	UK

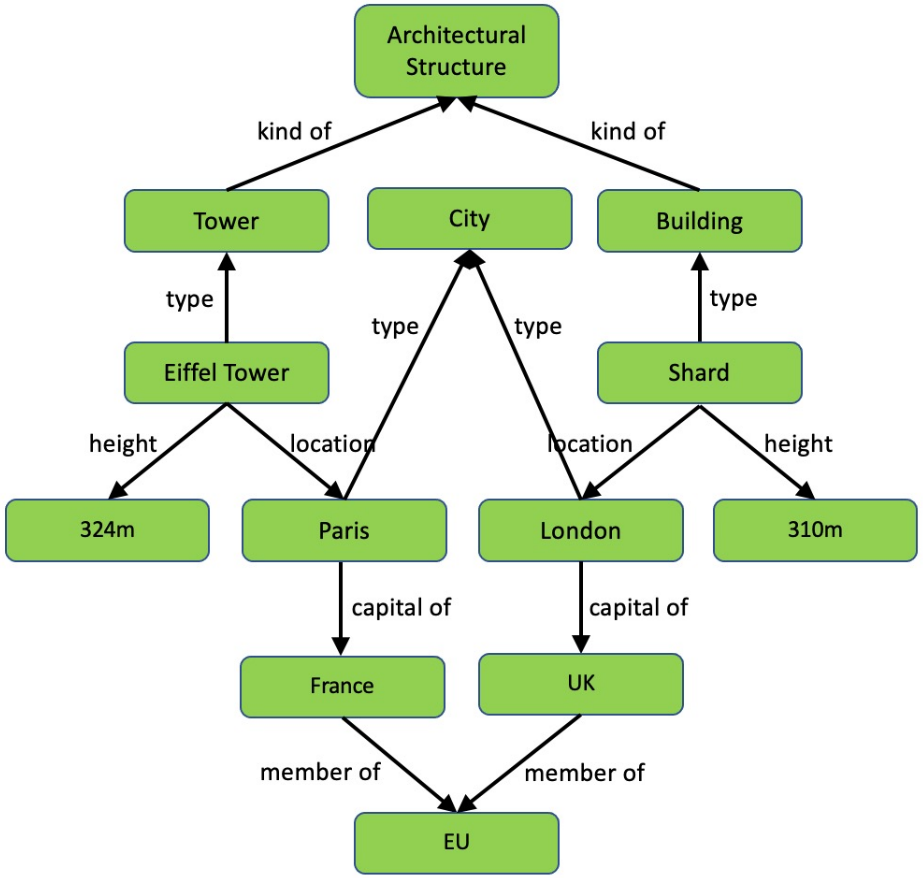
Member	
Country	Union
France	EU
UK	EU

Anatomy of a Knowledge Graph



✓ Intuitive (e.g., no “foreign keys”)

Anatomy of a Knowledge Graph



Architectural Structure		
Name	Height	Location
Eiffel Tower	324	Paris
Shard	310	London

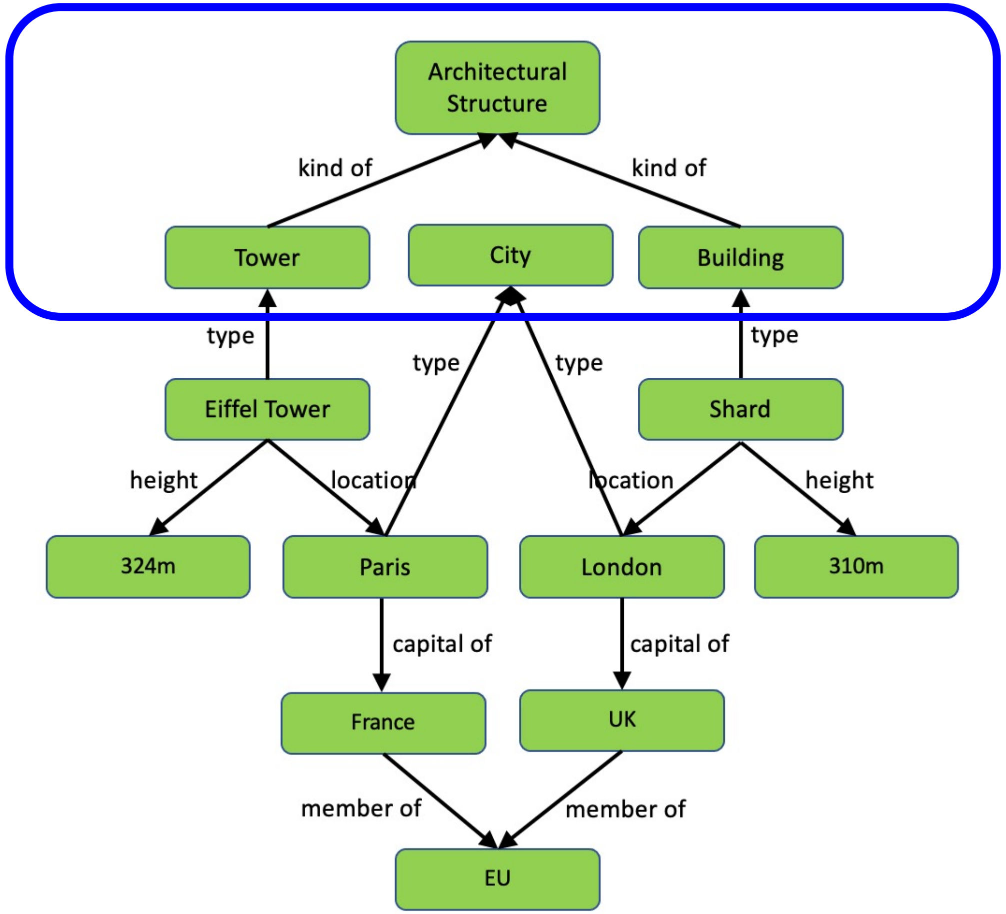
Tower
Name
Eiffel Tower

Building
Name
Shard

City	
Name	Capital Of
Paris	France
London	UK

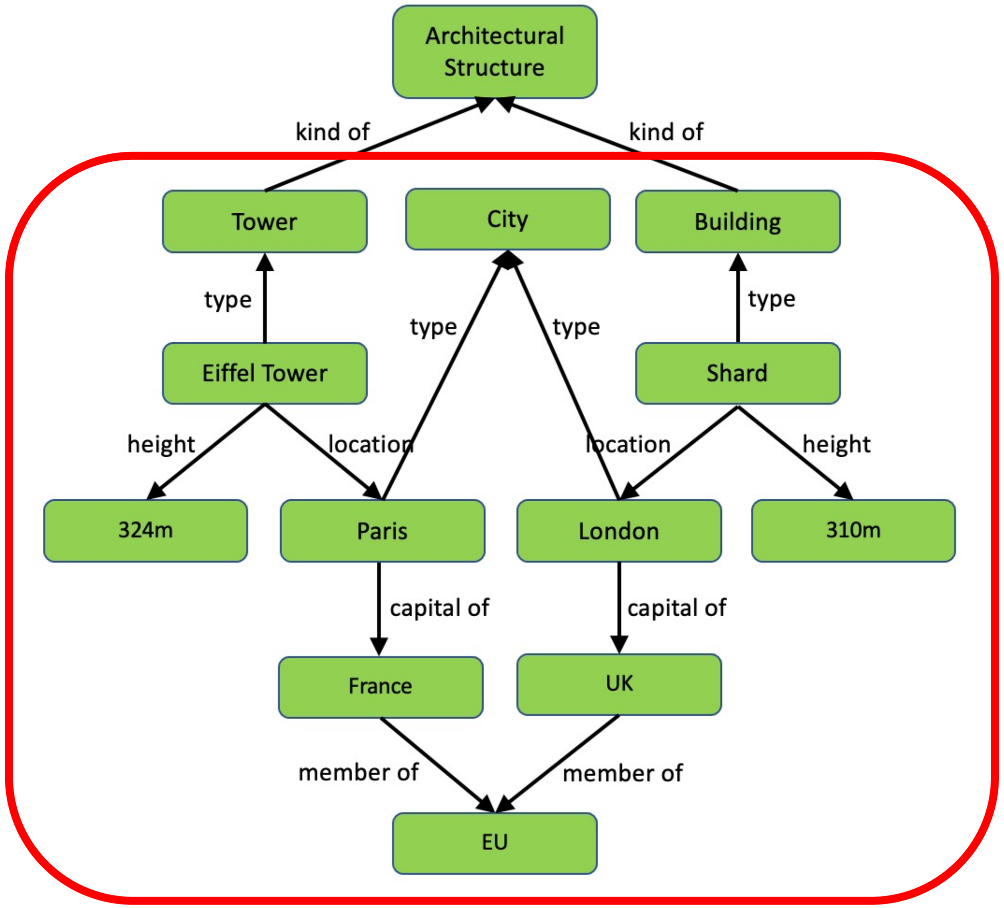
Member	
Country	Union
France	EU
UK	EU

Anatomy of a Knowledge Graph



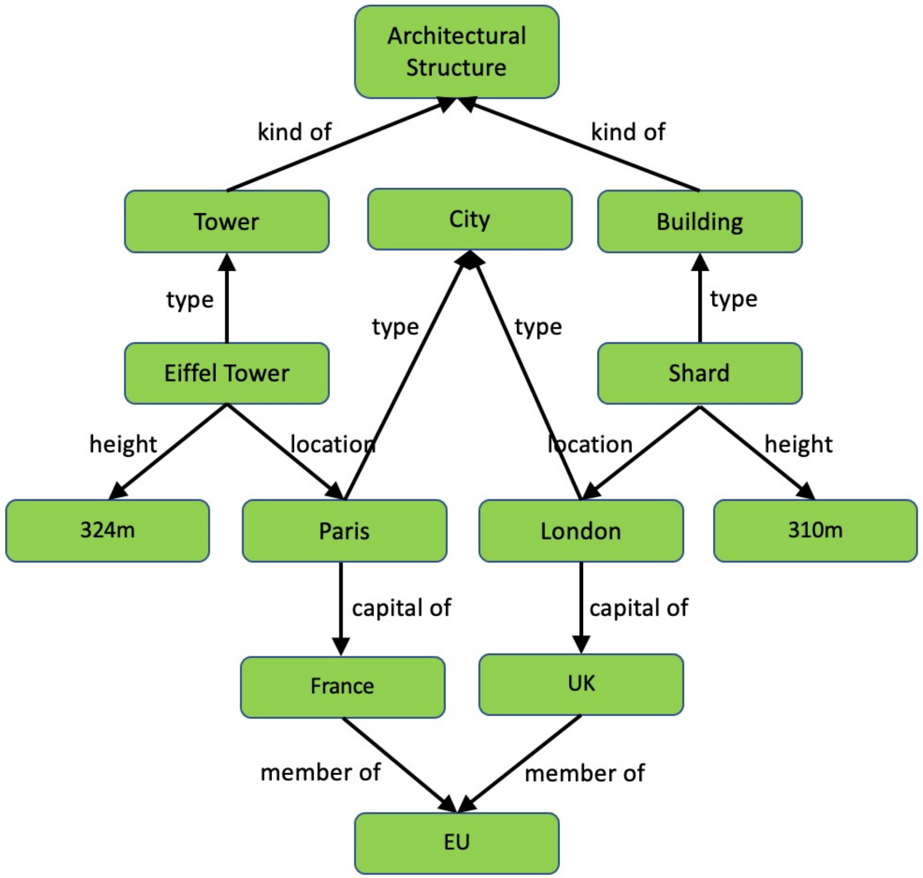
- ✓ Intuitive (e.g., no “foreign keys”)
- ✓ **Data** + **schema (ontology)**

Anatomy of a Knowledge Graph



- ✓ Intuitive (e.g., no "foreign keys")
- ✓ **Data** + **schema (ontology)**

Anatomy of a Knowledge Graph



Architectural Structure		
Name	Height	Location
Eiffel Tower	324	Paris
Shard	310	London

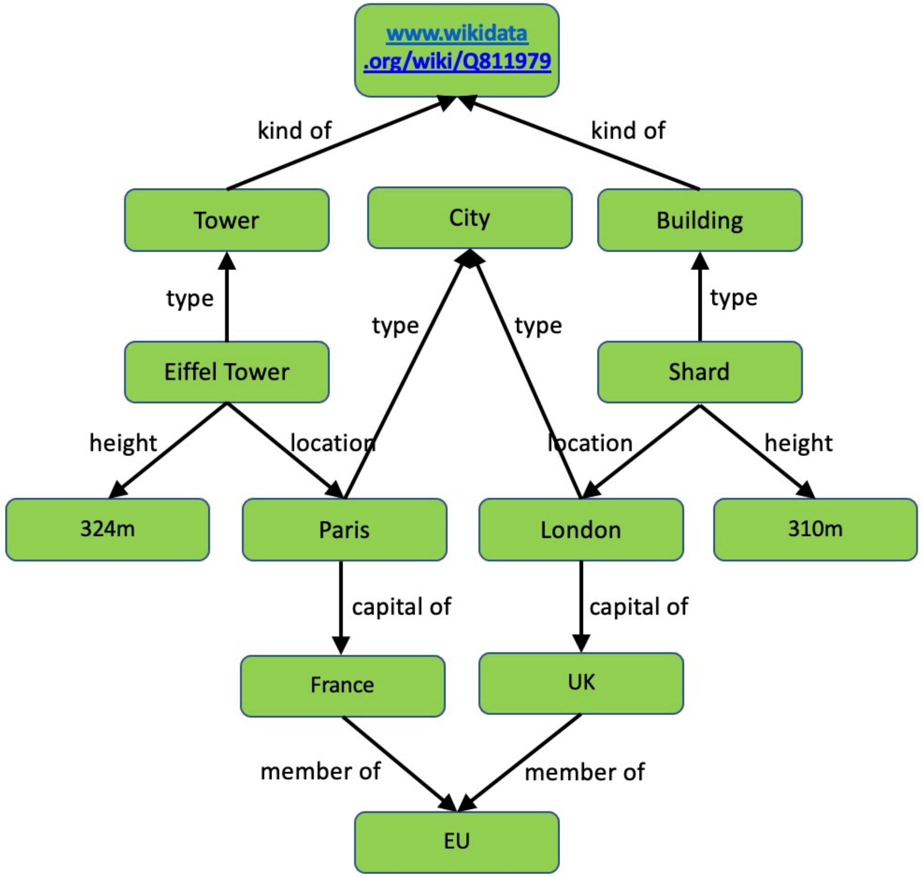
Tower
Name
Eiffel Tower

Building
Name
Shard

City	
Name	Capital Of
Paris	France
London	UK

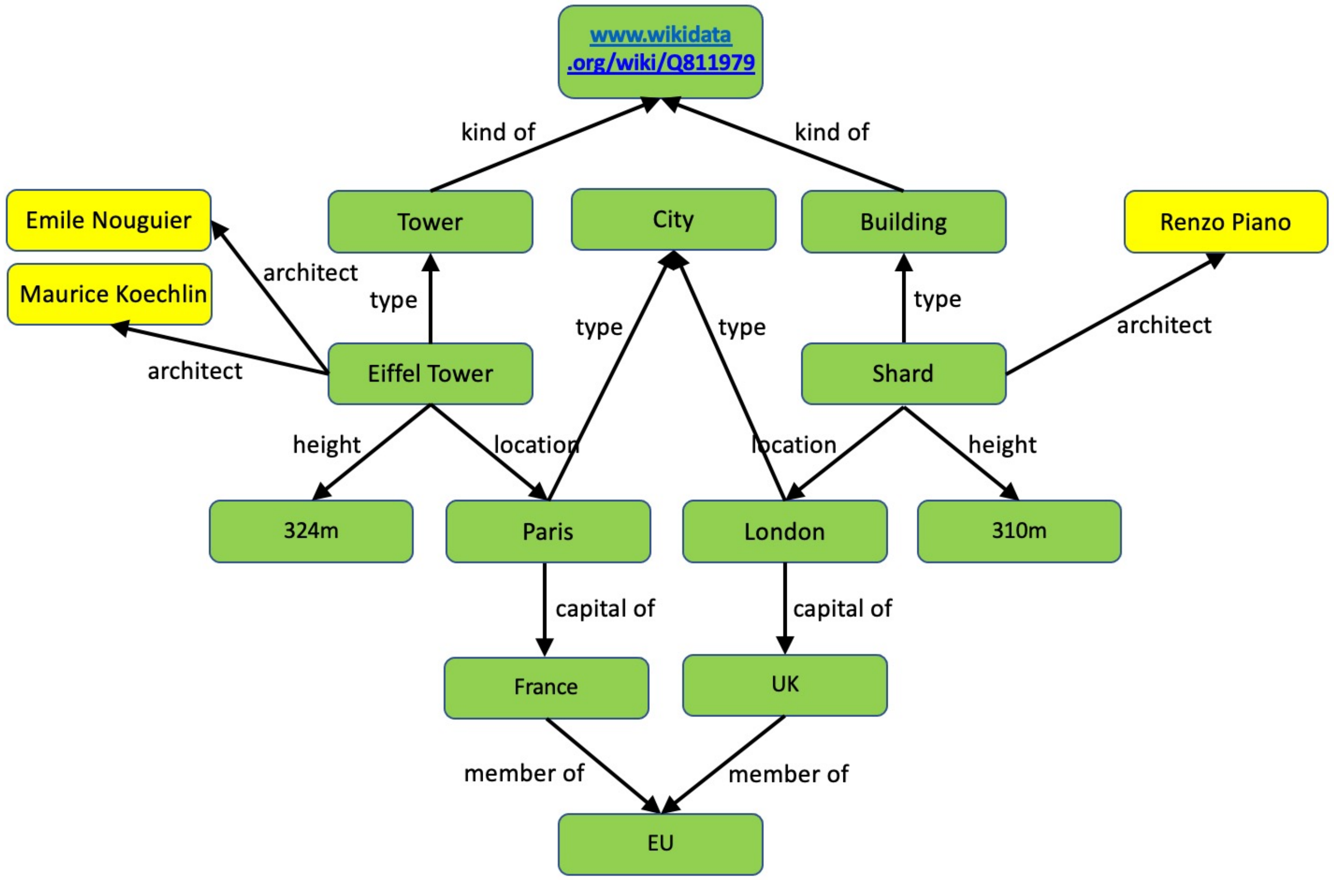
Member	
Country	Union
France	EU
UK	EU

Anatomy of a Knowledge Graph



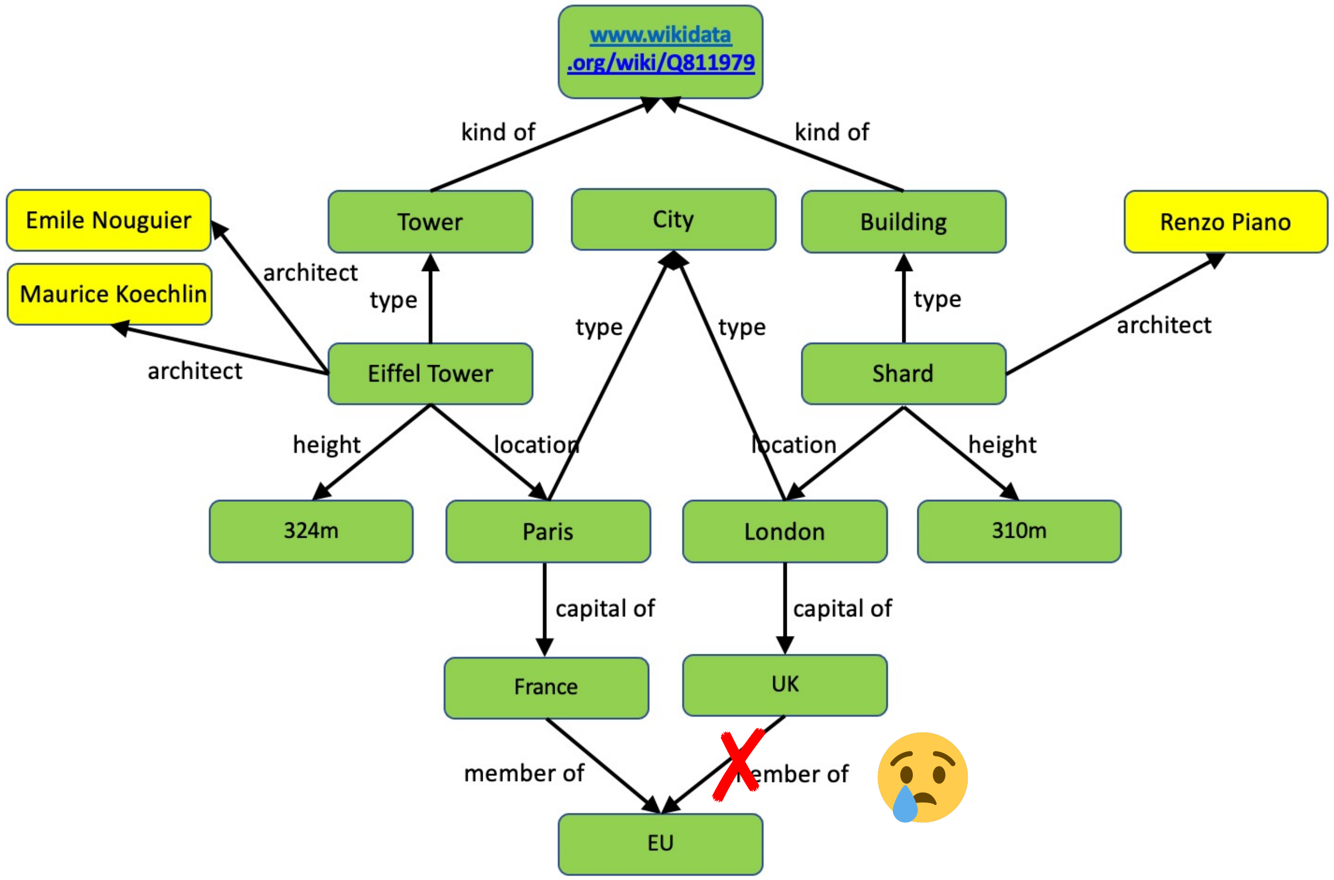
- ✓ Intuitive (e.g., no “foreign keys”)
- ✓ **Data** + **schema (ontology)**
- ✓ URIs not strings

Anatomy of a Knowledge Graph



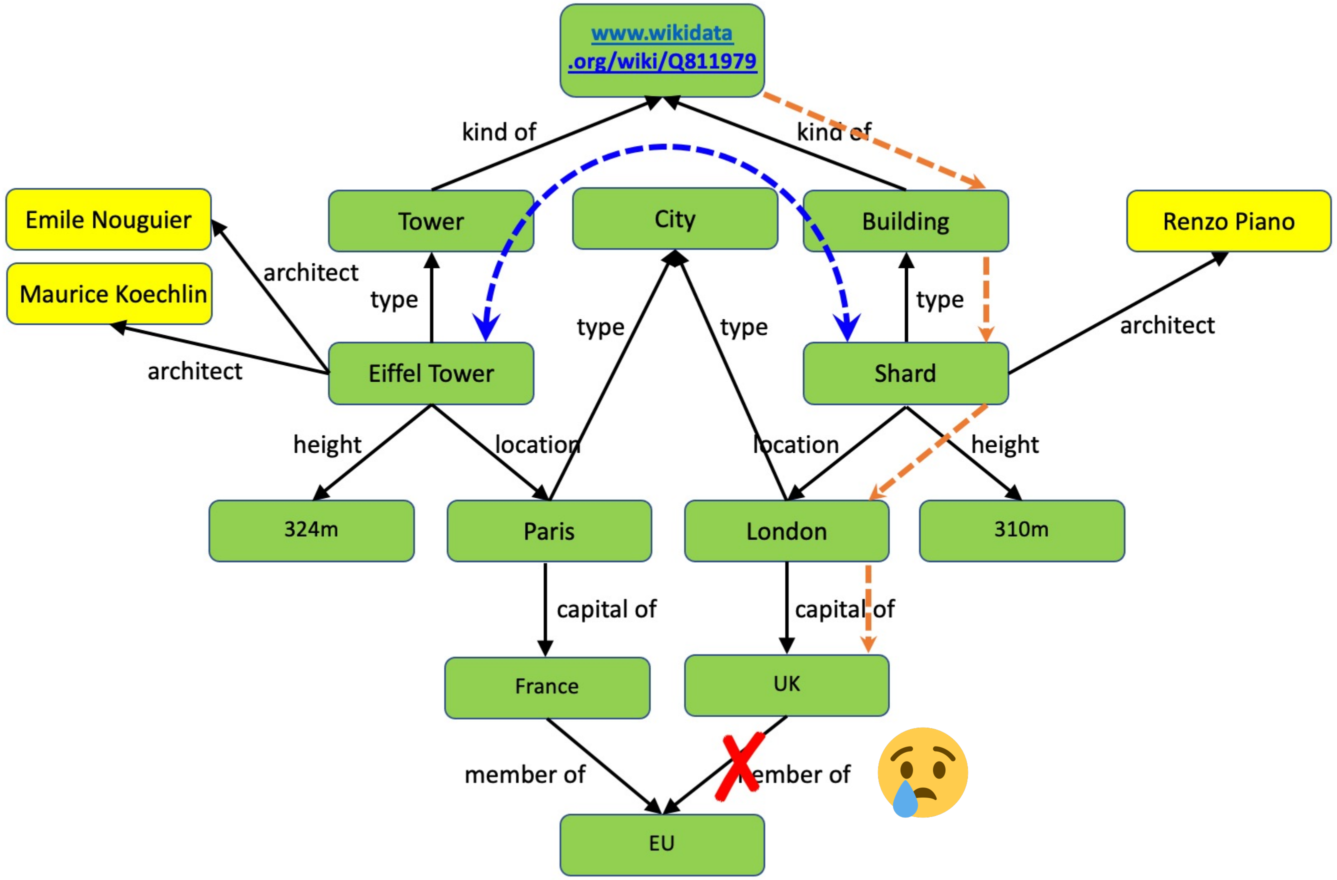
- ✓ Intuitive (e.g., no “foreign keys”)
- ✓ **Data** + **schema (ontology)**
- ✓ URIs not strings
- ✓ Flexible & extensible

Anatomy of a Knowledge Graph



- ✓ Intuitive (e.g., no “foreign keys”)
- ✓ **Data** + **schema (ontology)**
- ✓ URIs not strings
- ✓ Flexible & extensible

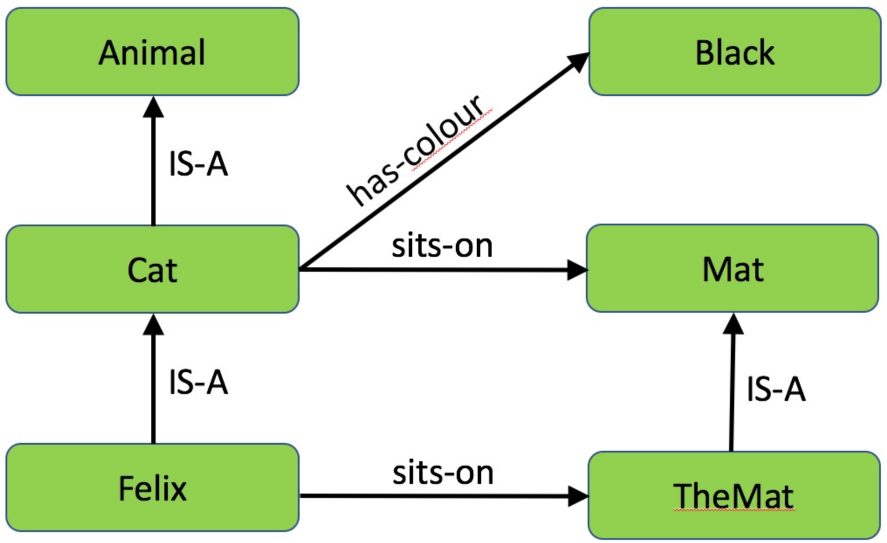
Anatomy of a Knowledge Graph



- ✓ Intuitive (e.g., no “foreign keys”)
- ✓ **Data** + **schema (ontology)**
- ✓ URIs not strings
- ✓ Flexible & extensible
- ✓ Other kinds of query
  - navigation
  - similarity & locality

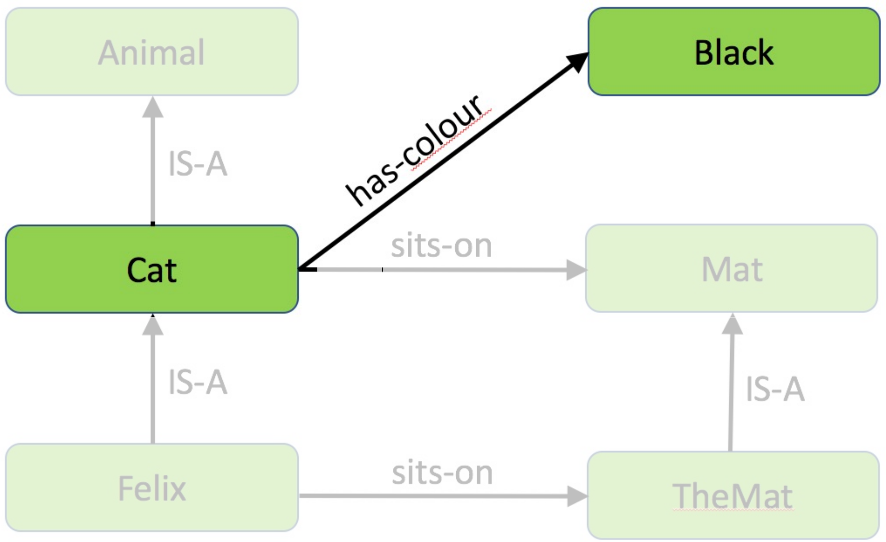
Challenges and Solutions (1)

Vague Semantics



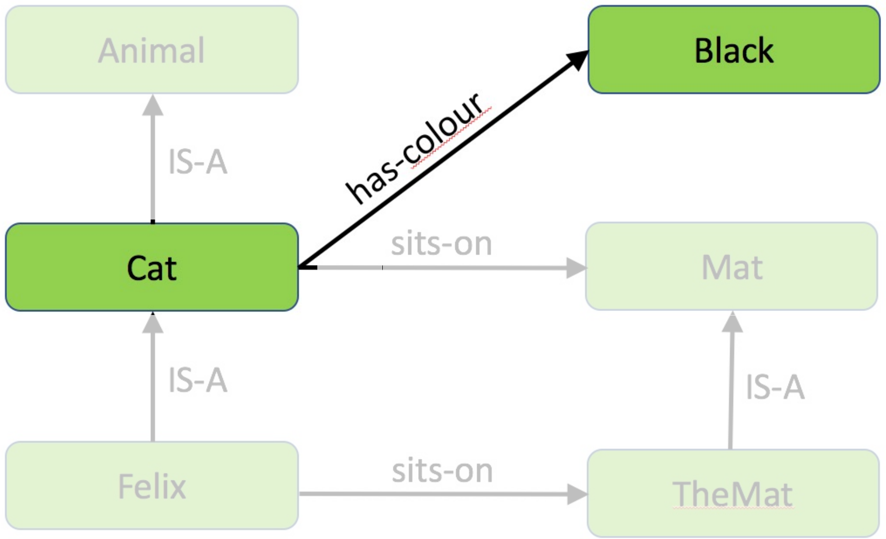
[Quillian, 1967]

Vague Semantics



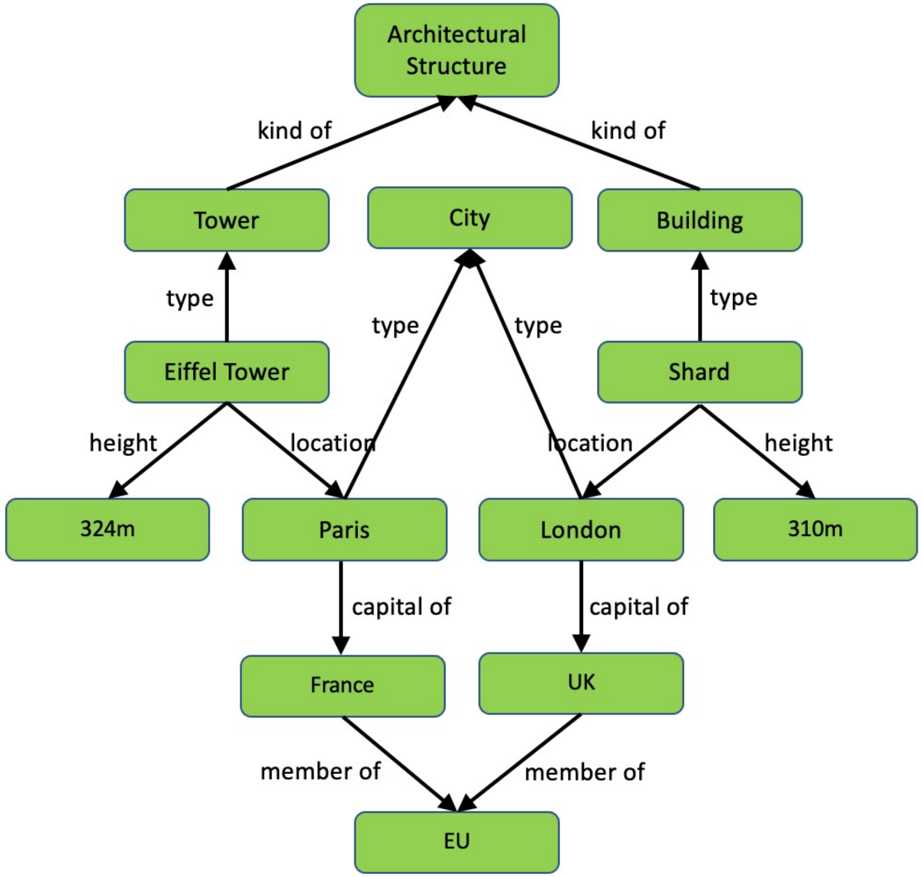
[Quillian, 1967]

Vague Semantics



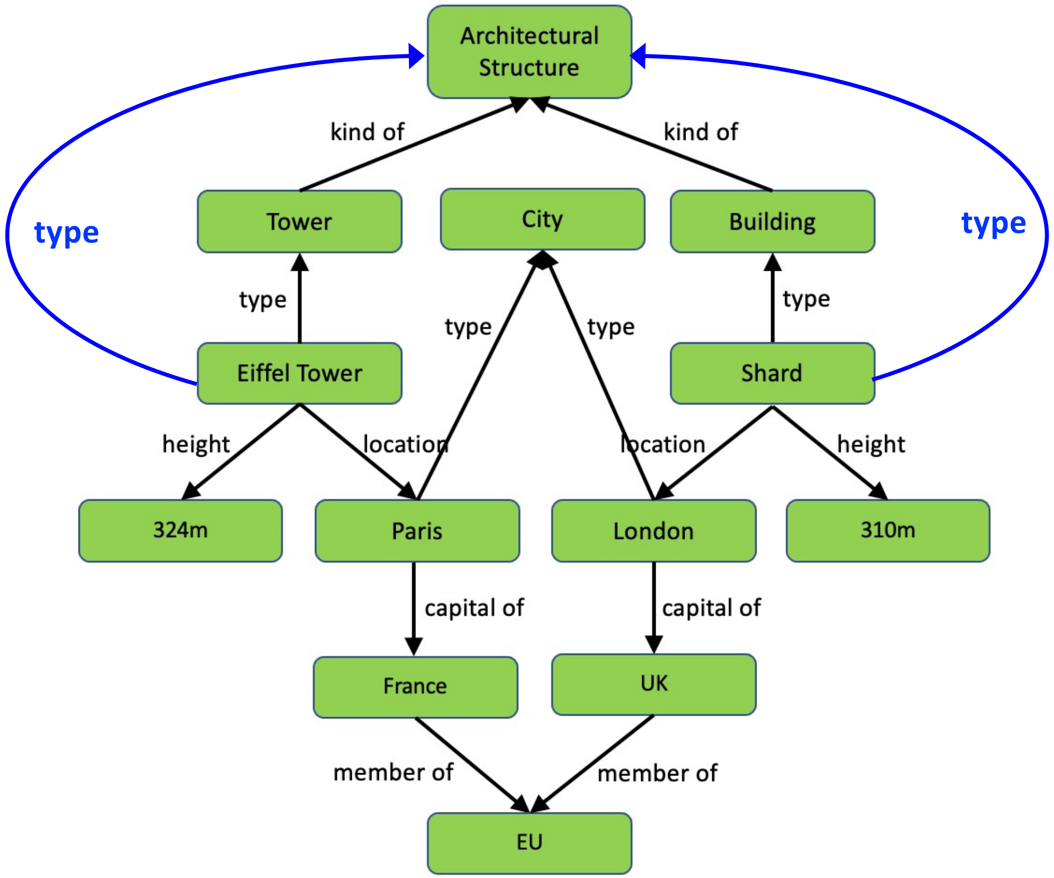
[Quillian, 1967]

Vague Semantics



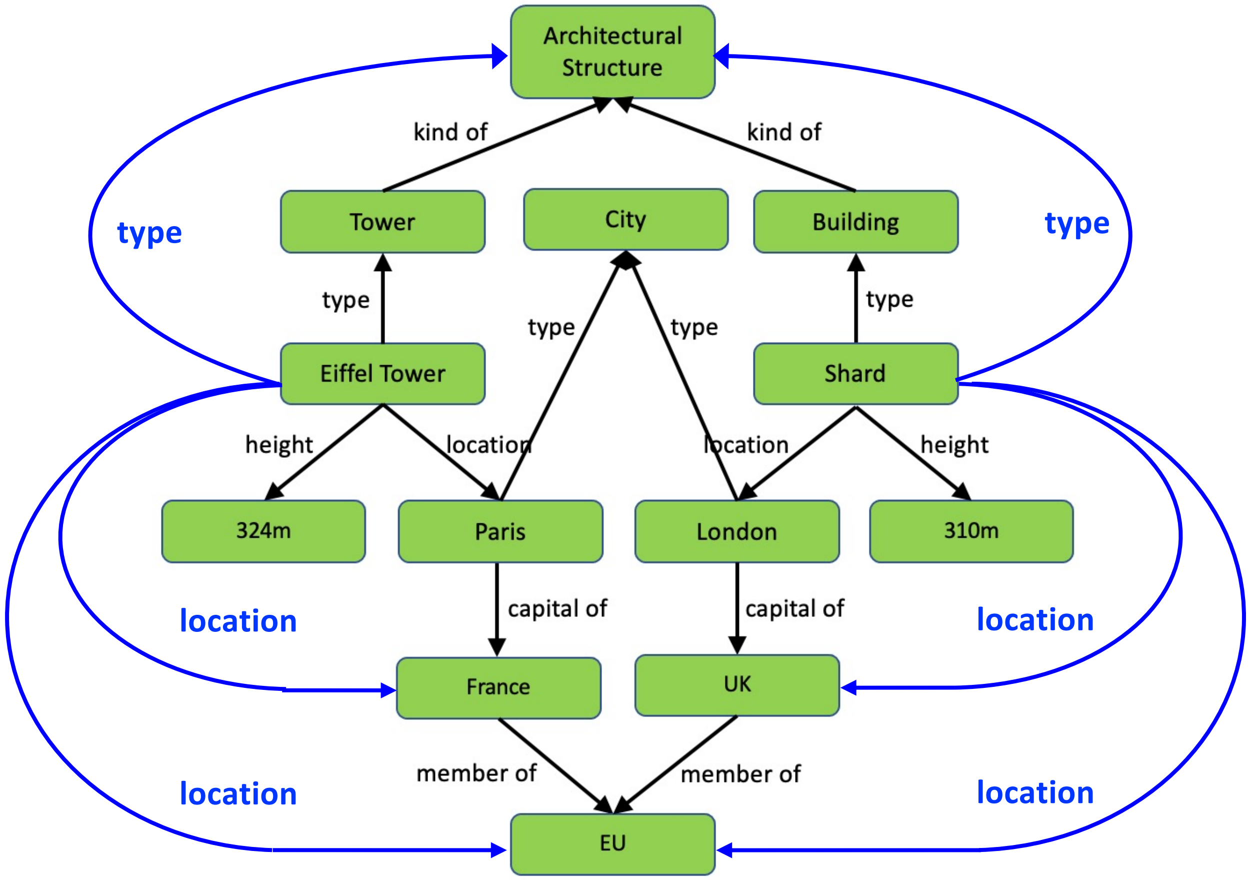
- Architectural Structure with location in the EU?

Vague Semantics



- Architectural Structure with location in the EU?
- Semantics of type and kind of edges?

Vague Semantics

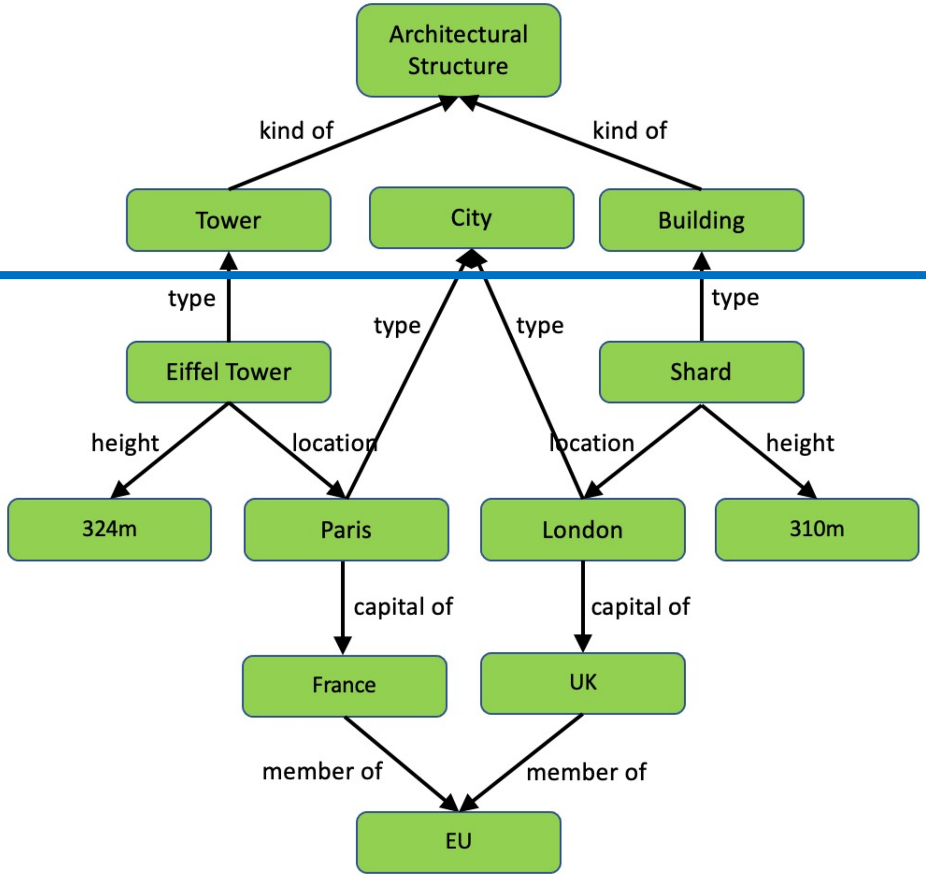


- Architectural Structure with location in the EU?
- Semantics of type and kind of edges?
- Semantics of location + capital of + member of edges?

Solution: Logic!

(OWL) ontology / conceptual schema

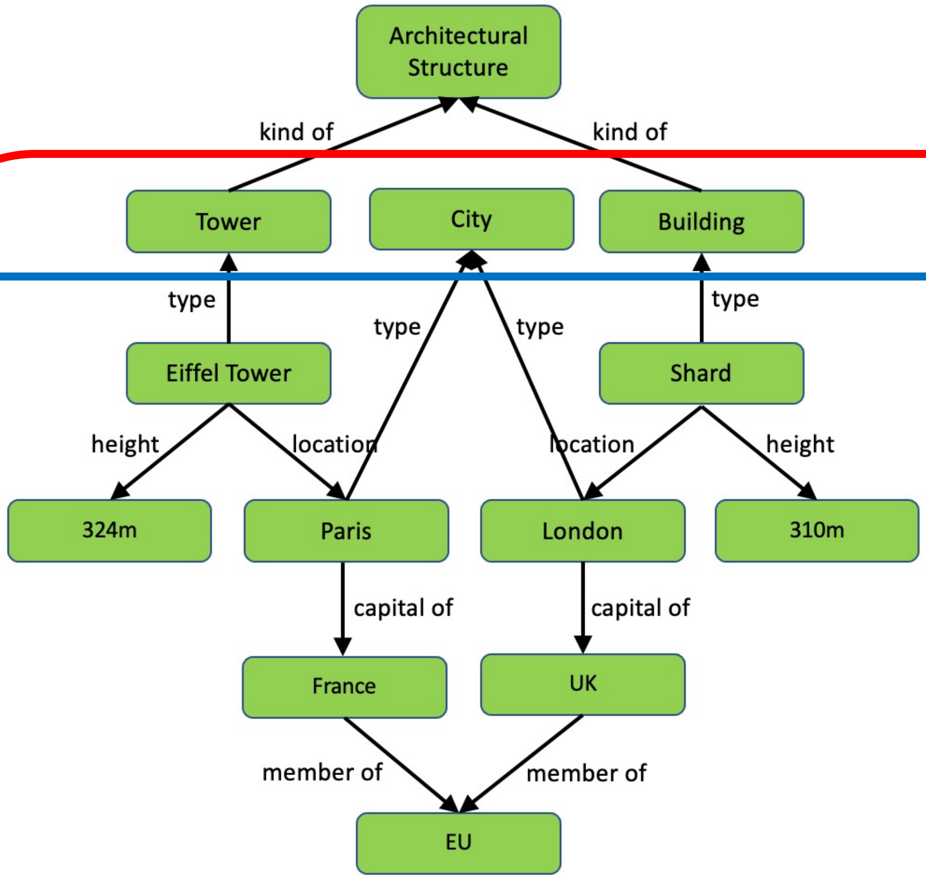
$\forall x\ \text{Tower}(x) \rightarrow \text{ArchitecturalStructure}(x)$
$\forall x\ \text{Building}(x) \rightarrow \text{ArchitecturalStructure}(x)$



Solution: Logic!

(OWL) ontology / conceptual schema

$\forall x\ \text{Tower}(x) \rightarrow \text{ArchitecturalStructure}(x)$
$\forall x\ \text{Building}(x) \rightarrow \text{ArchitecturalStructure}(x)$

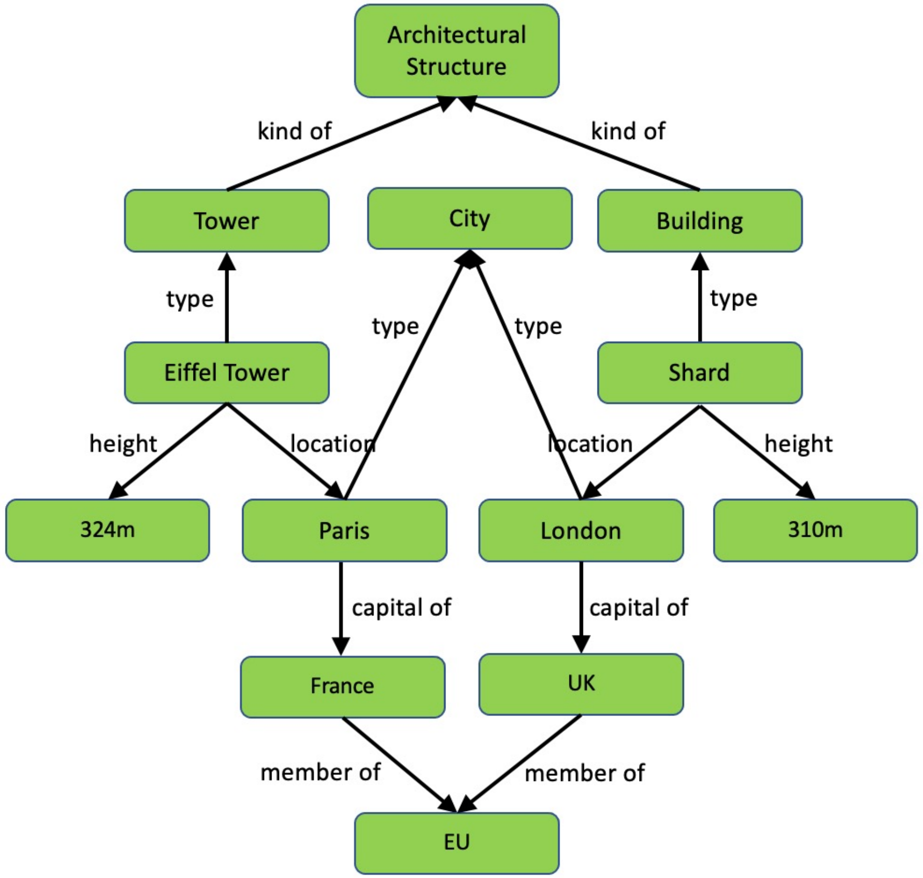


Tower(EiffelTower)	Building(Shard)
City(Paris)	City(London)
location(EiffelTower, Paris)	height(EiffelTower, 324m)
location(Shard, London)	height(Shard, 310m)
capital_of(Paris, France)	capital_of(London, UK)
member_of(France, EU)	member_of(UK, EU)

(RDF) graph / facts / data

Solution: Logic!

Knowledge base/graph

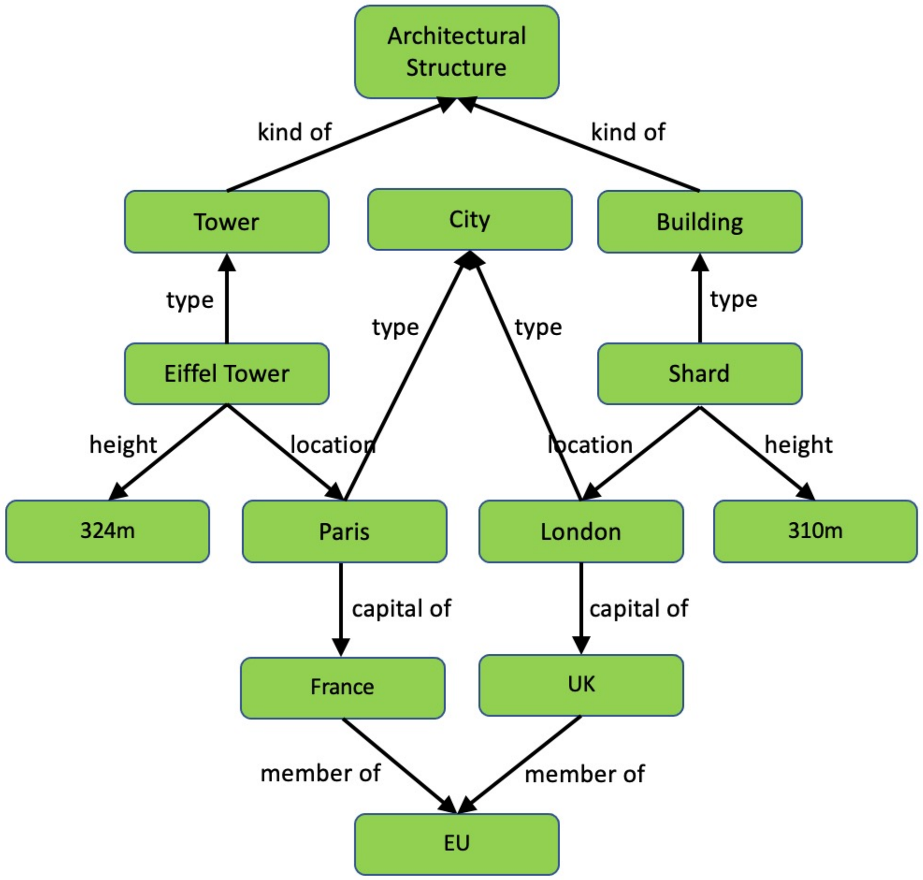


$\forall x\ \mathsf{Tower}(x) \rightarrow \mathsf{ArchitecturalStructure}(x)$
$\forall x\ \mathsf{Building}(x) \rightarrow \mathsf{ArchitecturalStructure}(x)$

$\mathsf{Tower}(\mathsf{EiffelTower})$	$\mathsf{Building}(\mathsf{Shard})$
$\mathsf{City}(\mathsf{Paris})$	$\mathsf{City}(\mathsf{London})$
$\mathsf{location}(\mathsf{EiffelTower}, \mathsf{Paris})$	$\mathsf{height}(\mathsf{EiffelTower}, \mathsf{324m})$
$\mathsf{location}(\mathsf{Shard}, \mathsf{London})$	$\mathsf{height}(\mathsf{Shard}, \mathsf{310m})$
$\mathsf{capital\_of}(\mathsf{Paris}, \mathsf{France})$	$\mathsf{capital\_of}(\mathsf{London}, \mathsf{UK})$
$\mathsf{member\_of}(\mathsf{France}, \mathsf{EU})$	$\mathsf{member\_of}(\mathsf{UK}, \mathsf{EU})$

Solution: Logic!

Knowledge base/graph



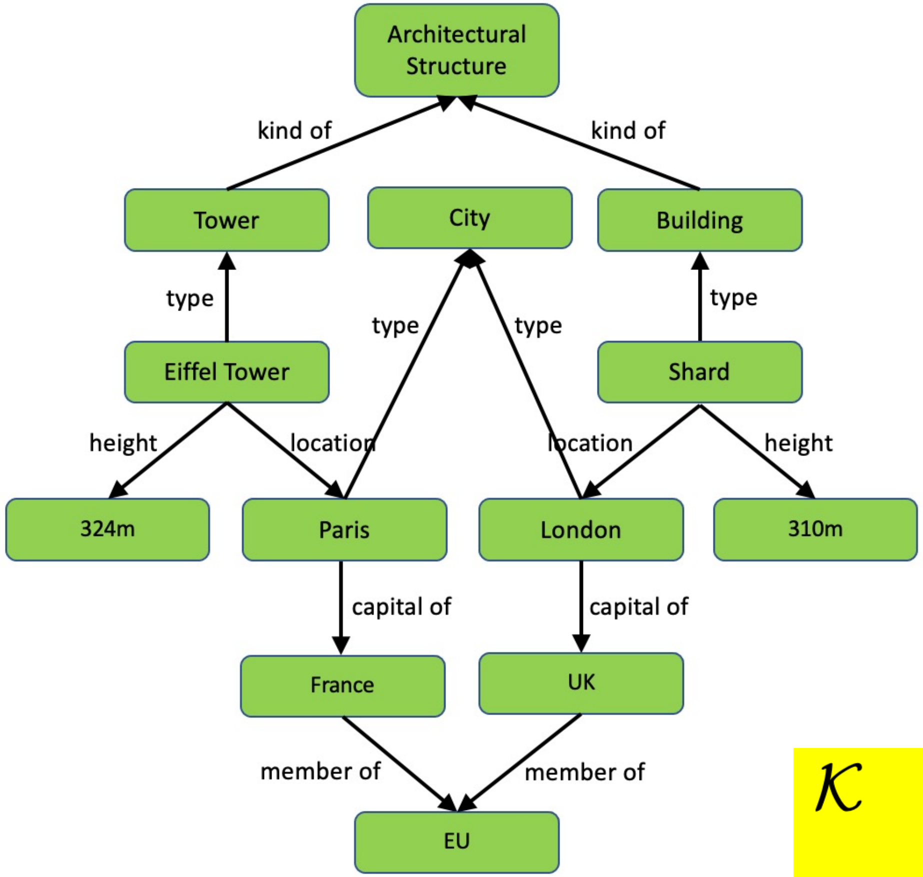
$\forall x$ Tower$(x) \rightarrow$ ArchitecturalStructure(x)
$\forall x$ Building$(x) \rightarrow$ ArchitecturalStructure(x)
$\forall x, y, z$ location$(x, y) \wedge$ capital_of$(y, z) \rightarrow$ location(x, z)
$\forall x, y, z$ location$(x, y) \wedge$ member_of$(y, z) \rightarrow$ location(x, z)

Tower(EiffelTower)
City(Paris)
location(EiffelTower, Paris)
location(Shard, London)
capital_of(Paris, France)
member_of(France, EU)

Building(Shard)
City(London)
height(EiffelTower, 324m)
height(Shard, 310m)
capital_of(London, UK)
member_of(UK, EU)

Solution: Logic!

Knowledge base/graph



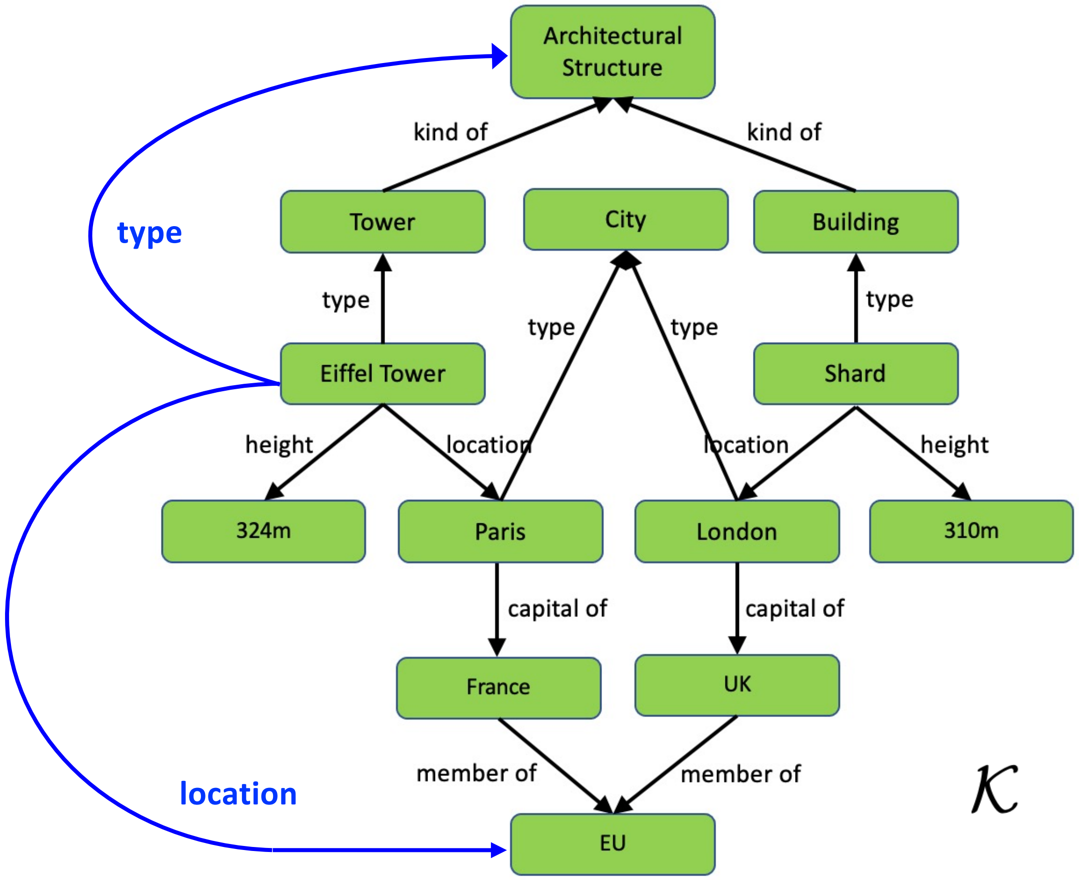
$\forall x\ \text{Tower}(x) \rightarrow \text{ArchitecturalStructure}(x)$
$\forall x\ \text{Building}(x) \rightarrow \text{ArchitecturalStructure}(x)$
$\forall x, y, z\ \text{location}(x, y) \wedge \text{capital\_of}(y, z) \rightarrow \text{location}(x, z)$
$\forall x, y, z\ \text{location}(x, y) \wedge \text{member\_of}(y, z) \rightarrow \text{location}(x, z)$

Tower(EiffelTower)	Building(Shard)
City(Paris)	City(London)
location(EiffelTower, Paris)	height(EiffelTower, 324m)
location(Shard, London)	height(Shard, 310m)
capital_of(Paris, France)	capital_of(London, UK)
member_of(France, EU)	member_of(UK, EU)

$$\mathcal{K} \models \text{ArchitecturalStructure}(\text{EiffelTower}) \wedge \text{location}(\text{EiffelTower}, \text{EU})$$

Solution: Logic!

Knowledge base/graph



$\forall x$ Tower$(x) \to$ ArchitecturalStructure(x)
$\forall x$ Building$(x) \to$ ArchitecturalStructure(x)
$\forall x, y, z$ location$(x, y) \wedge$ capital_of$(y, z) \to$ location(x, z)
$\forall x, y, z$ location$(x, y) \wedge$ member_of$(y, z) \to$ location(x, z)

Tower(EiffelTower)
City(Paris)
location(EiffelTower, Paris)
location(Shard, London)
capital_of(Paris, France)
member_of(France, EU)

Building(Shard)
City(London)
height(EiffelTower, 324m)
height(Shard, 310m)
capital_of(London, UK)
member_of(UK, EU)

$\mathcal{K} \models$ ArchitecturalStructure(EiffelTower) $\wedge$
location(EiffelTower, EU)

Solution: Logic!

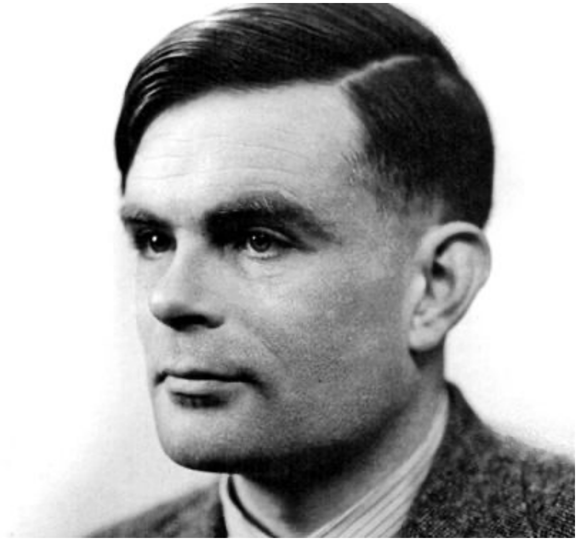
- Identify/devise algorithms that compute query answers
- E.g., using natural deduction rules:

$$\frac{\forall x P(x) \rightarrow Q(x) \quad P(a)}{Q(a)} \qquad \frac{\forall x\, \text{Building}(x) \rightarrow \text{ArchitecturalStructure}(x) \quad \text{Building}(\text{EiffelTower})}{\text{ArchitecturalStructure}(\text{EiffelTower})}$$

- Can check/prove algorithms are sound and complete w.r.t. semantics

Problem Solved?

- Some problems cannot be completely solved using standard computational model
  - halting problem
  - **FOL entailment problem**



- Even if decidable, reasoning might be of inherently high complexity and so take an infeasibly long time

So what to do?

- These are worst case results
  - Even if *logic* is undecidable, some *problems* may still be decidable
  - Even if *logic* is intractable, some *problems* may still be tractable

- **Study** KR languages to find suitable balance of expressive power and computability
- **Design** reasoning algorithms that work well in typical cases
- **Develop** highly optimised implementations

Description Logic

- Family of logic-based KR languages
- Most are decidable subsets of FOPC (usually in C2)
- Provide a range of different constructors
  - Booleans (and, or, not)
  - Restricted forms of quantification (exists, forall)
  - Counting (atmost, atleast)
  - …
- Decidability/complexity and (efficient) algorithms known for many combinations of constructors
- Effective reasoners available for several “sweet-spot” DLs

W3C and the Semantic Web

- Goal: to make web data machine-readable
  - KRR on the web
- Standardized **RDF**
  - Graphical data model for representing facts
- Extended RDF with **OWL**
  - Ontology language based on expressive DL (**SROIQ)**
- Developed **SPARQL** query language
  - Similar to SQL
  - Tailored to graphical data model

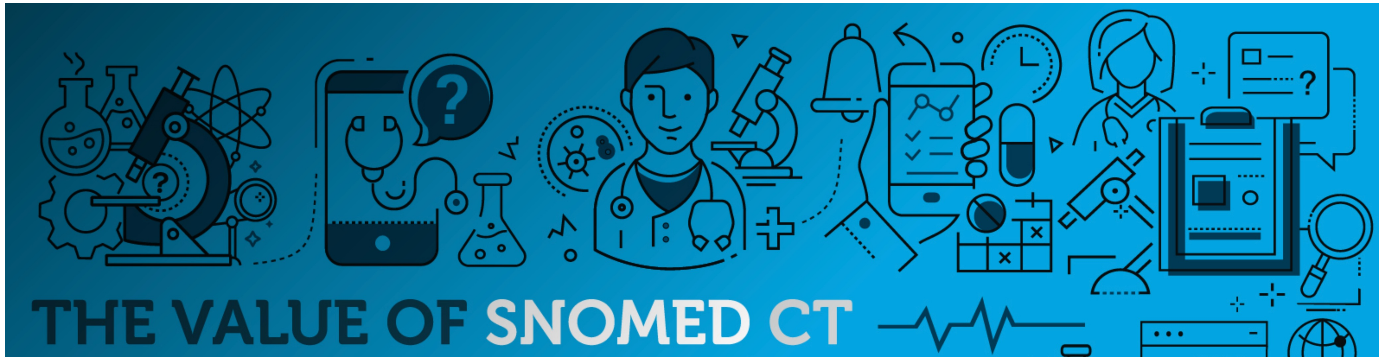
RDF

SPARQL

Challenges and Solutions (2)

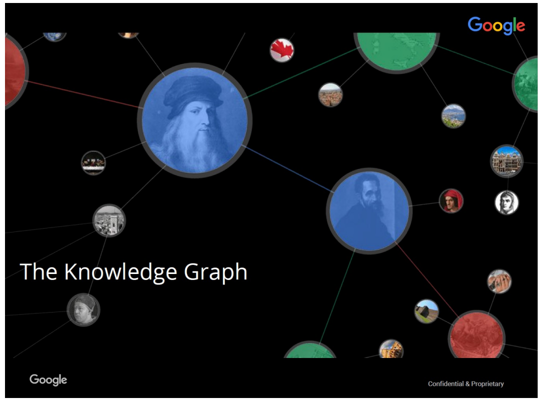
Ontology-centric Applications

- Development of large/complex ontologies
  - Class axioms (usually $<10^6$ classes) with few or no facts
  - Main reasoning task is consistency/subsumption
- OWL/DL reasoners such as **HermiT** and **ELK** used
  - to identify errors and inconsistencies
  - to compute class hierarchy (classification)
- Widely used in medicine and life sciences
  - Bioportal (900+ ontologies)
  - **SNOMED CT**
  - …



Data-centric Applications

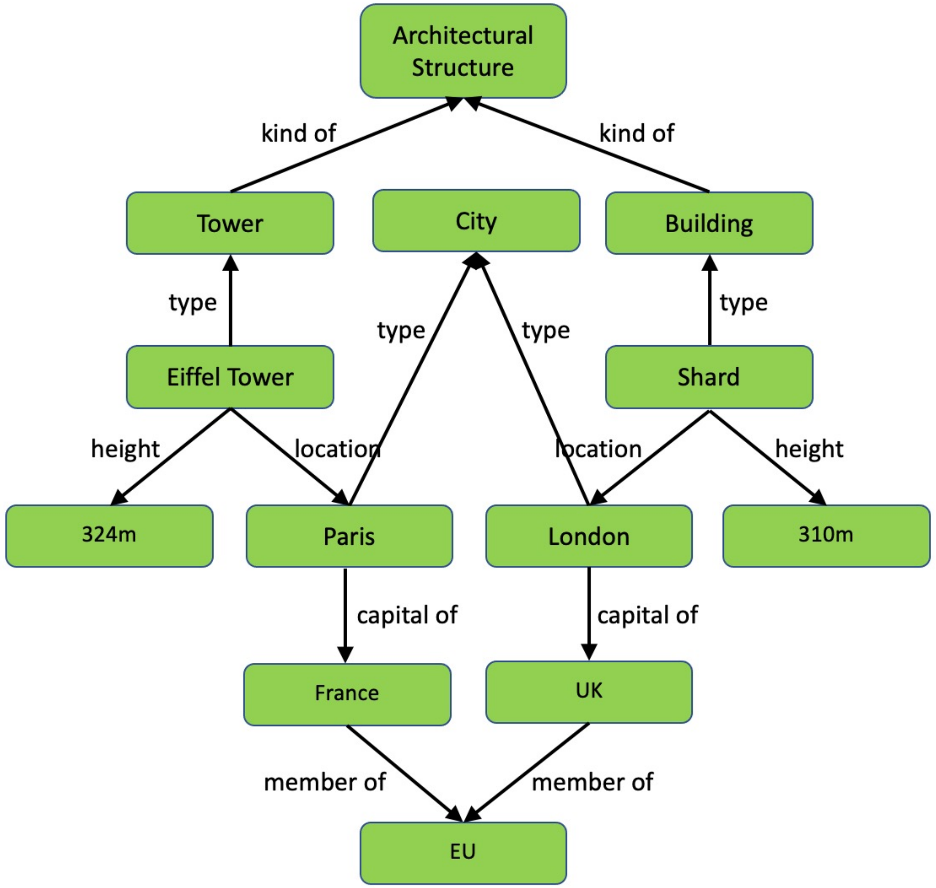
- Development and deployment of large knowledge graphs
  - Ontology/rules plus large number of facts (can be $>10^9$ edges)
  - Main reasoning task is (SPARQL) query answering
- OWL/DL reasoners don't scale well to this task
  - Query answering reduces to multiple entailment checks
  - Number of checks is polynomial in size of graph
  - Each such check can be costly



OWL 2 Profiles

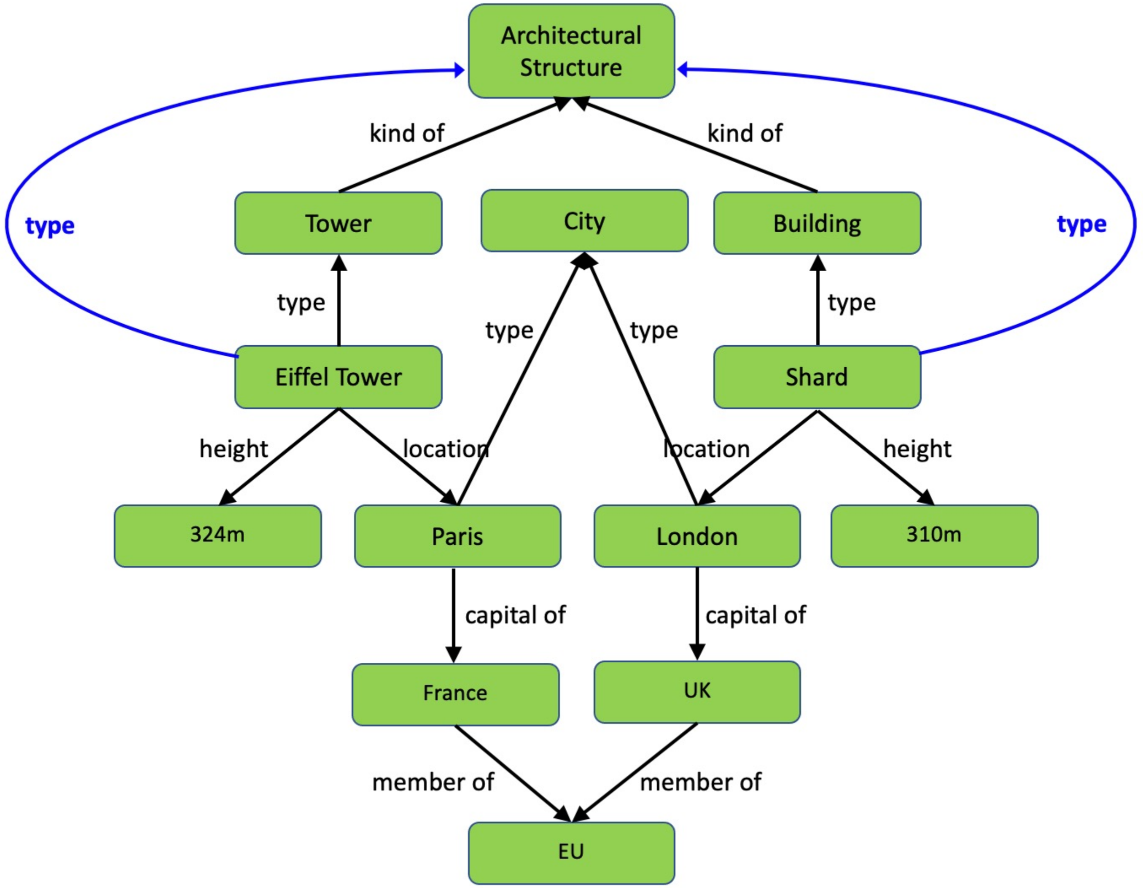
- OWL 2 is based on powerful but still decidable DL (SROIQ)
- OWL 2 also introduced three “profiles” based on tractable subsets
  - **QL**: based on the DL-Lite description logic
  - **EL**: based on the EL description logic
  - **RL**: based on the DL fragment of Datalog (aka DLP)
- Profiles allow for algorithmic techniques suited to query answering
  - **Query rewriting** for QL
  - **Materialisation** for RL
  - **Combined approach** for EL

Materialisation for OWL RL



$$Q(x) \leftarrow \text{Architectural\_Structure}(x)$$

Materialisation for OWL RL

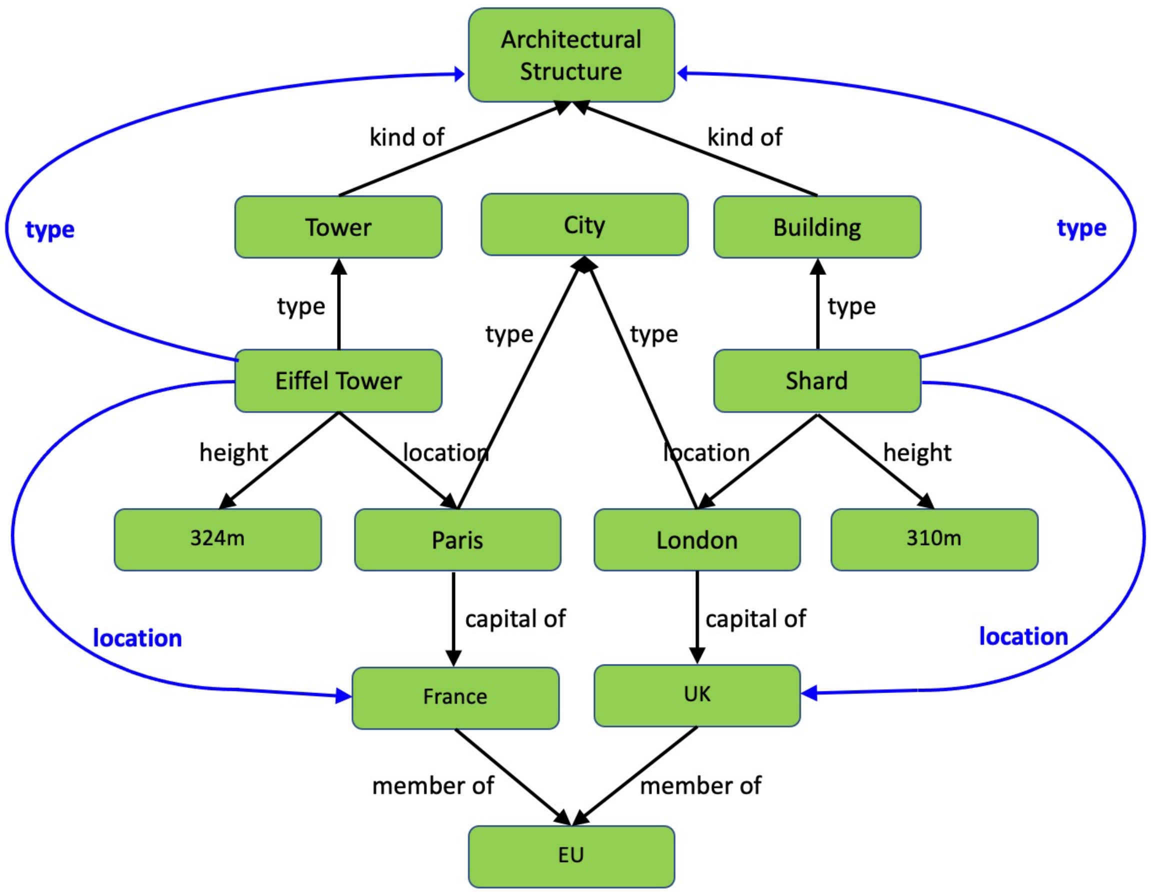


$$Q(x) \leftarrow \text{Architectural\_Structure}(x)$$

$$\text{Tower} \sqsubseteq \text{Architectural\_Structure}$$
$$\text{Building} \sqsubseteq \text{Architectural\_Structure}$$

$$\text{Tower}(x) \rightarrow \text{Architectural\_Structure}(x)$$
$$\text{Building}(x) \rightarrow \text{Architectural\_Structure}(x)$$

Materialisation for OWL RL



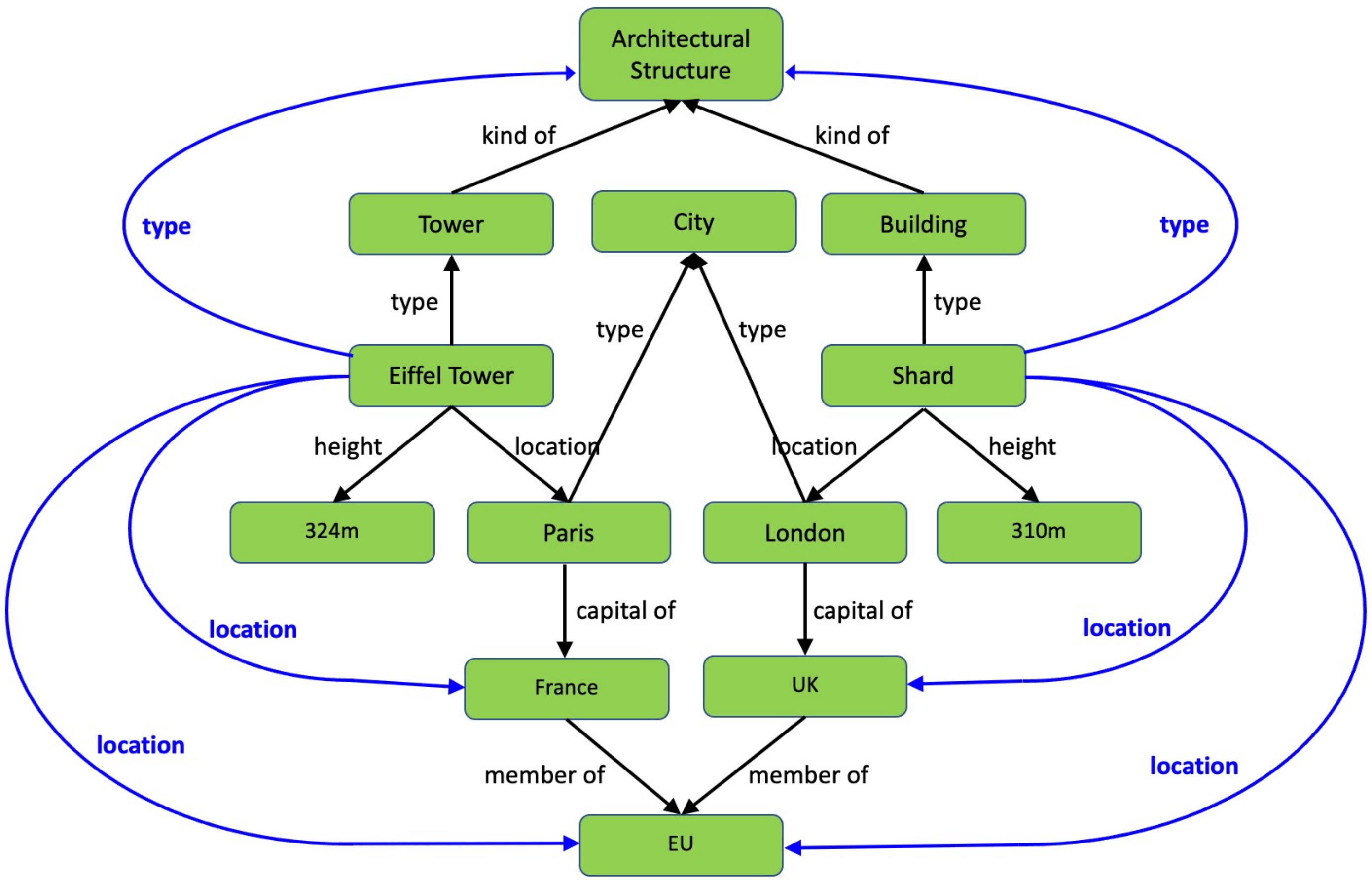
$$Q(x) \leftarrow \text{Architectural\_Structure}(x)$$

$$\text{Tower} \sqsubseteq \text{Architectural\_Structure}$$
$$\text{Building} \sqsubseteq \text{Architectural\_Structure}$$

$$\text{Tower}(x) \rightarrow \text{Architectural\_Structure}(x)$$
$$\text{Building}(x) \rightarrow \text{Architectural\_Structure}(x)$$

$$\text{location}(x, y) \wedge \text{capital\_of}(y, z) \rightarrow \text{location}(x, z)$$

Materialisation for OWL RL



$$Q(x) \leftarrow \text{Architectural\_Structure}(x)$$

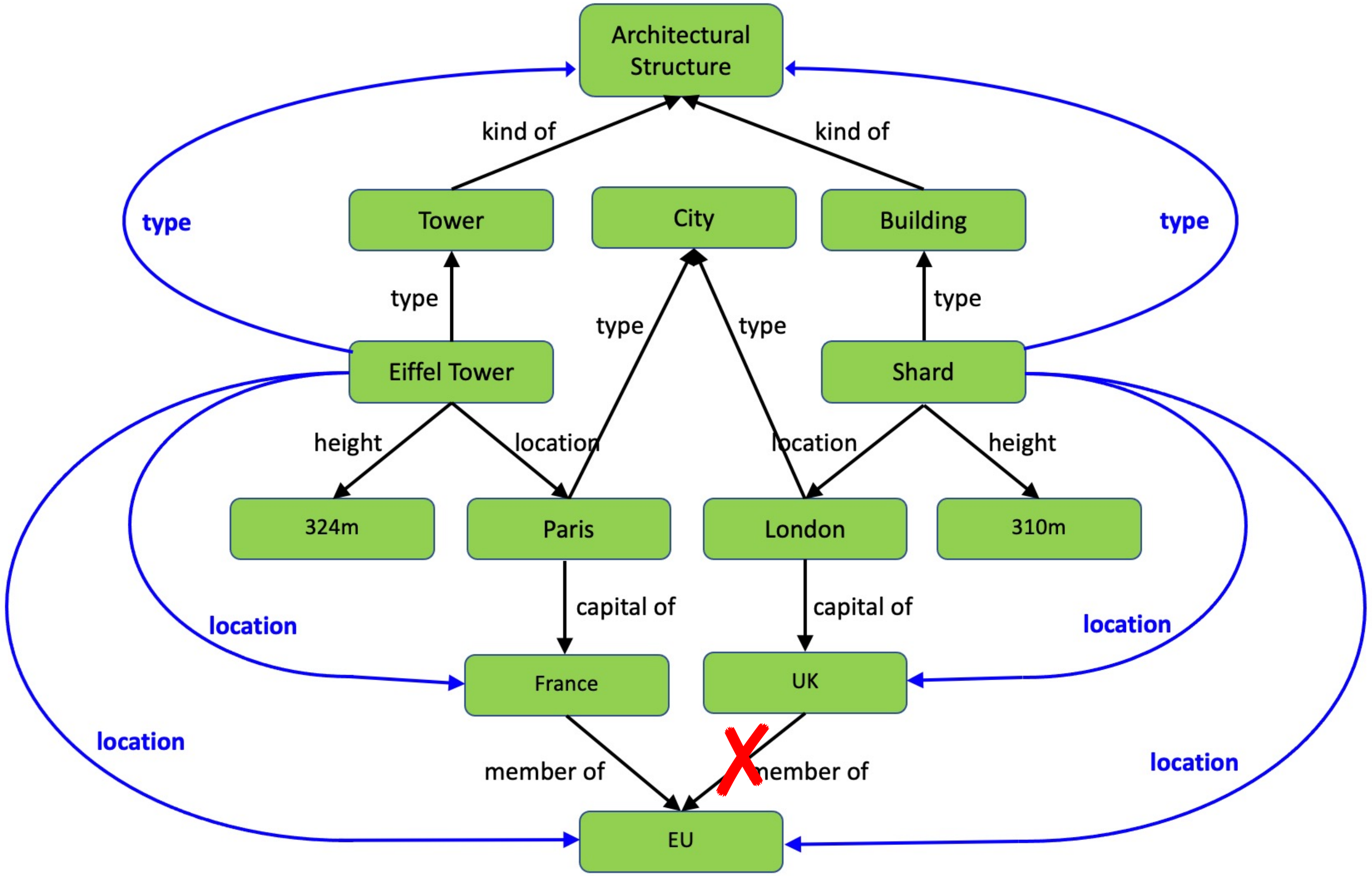
$$\text{Tower} \sqsubseteq \text{Architectural\_Structure}$$
$$\text{Building} \sqsubseteq \text{Architectural\_Structure}$$

$$\text{Tower}(x) \rightarrow \text{Architectural\_Structure}(x)$$
$$\text{Building}(x) \rightarrow \text{Architectural\_Structure}(x)$$

$$\text{location}(x, y) \wedge \text{capital\_of}(y, z) \rightarrow \text{location}(x, z)$$

$$\text{location}(x, y) \wedge \text{member\_of}(y, z) \rightarrow \text{location}(x, z)$$

Materialisation for OWL RL



$$Q(x) \leftarrow \text{Architectural\_Structure}(x)$$

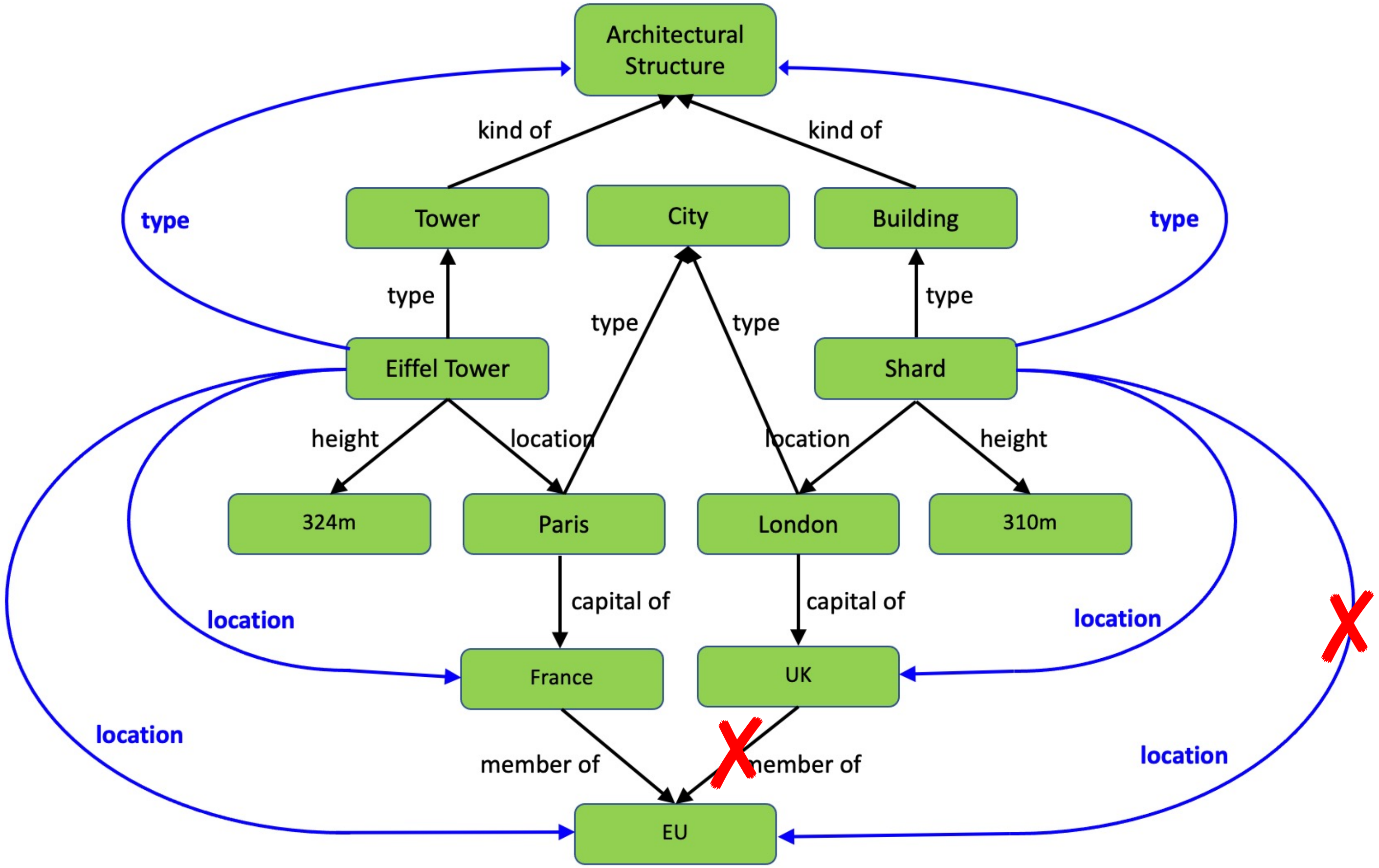
$$\text{Tower} \sqsubseteq \text{Architectural\_Structure}$$
$$\text{Building} \sqsubseteq \text{Architectural\_Structure}$$

$$\text{Tower}(x) \rightarrow \text{Architectural\_Structure}(x)$$
$$\text{Building}(x) \rightarrow \text{Architectural\_Structure}(x)$$

$$\text{location}(x, y) \wedge \text{capital\_of}(y, z) \rightarrow \text{location}(x, z)$$

$$\text{location}(x, y) \wedge \text{member\_of}(y, z) \rightarrow \text{location}(x, z)$$

Materialisation for OWL RL



$$Q(x) \leftarrow \text{Architectural\_Structure}(x)$$

$$\text{Tower} \sqsubseteq \text{Architectural\_Structure}$$
$$\text{Building} \sqsubseteq \text{Architectural\_Structure}$$

$$\text{Tower}(x) \rightarrow \text{Architectural\_Structure}(x)$$
$$\text{Building}(x) \rightarrow \text{Architectural\_Structure}(x)$$

$$\text{location}(x, y) \wedge \text{capital\_of}(y, z) \rightarrow \text{location}(x, z)$$

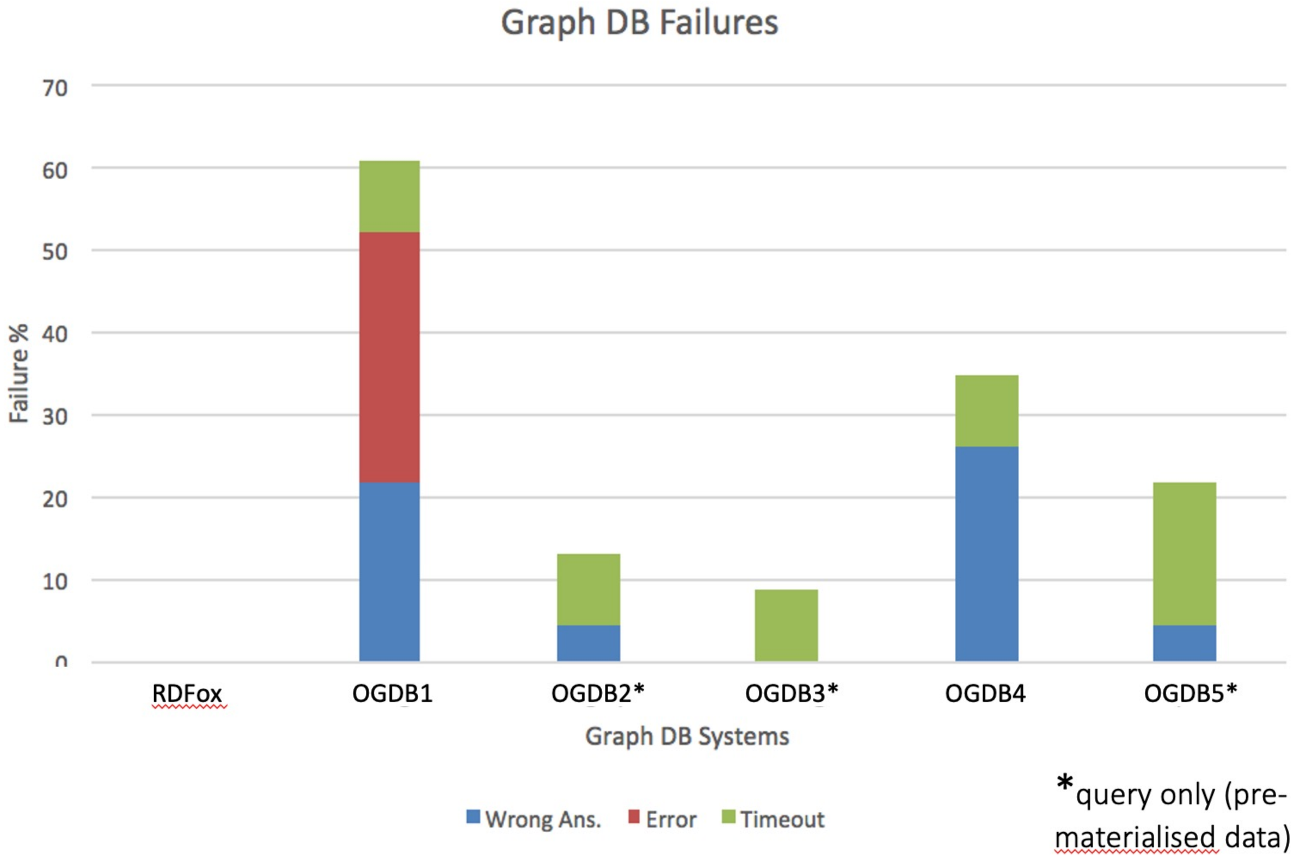
$$\text{location}(x, y) \wedge \text{member\_of}(y, z) \rightarrow \text{location}(x, z)$$

RDFOX

- Materialization reasoning seems ideal for data-centric applications
  - Can support expressive ontology languages
  - Fast query answering over very large graphs
- Challenges
  - Materialisation can be costly in time and memory
  - Materialisation may need to be repeated if data changes
- Solution: RDFox
  - Optimised materialization exploiting modern multi-core architectures
  - Incremental maintenance as data changes

RDFOX

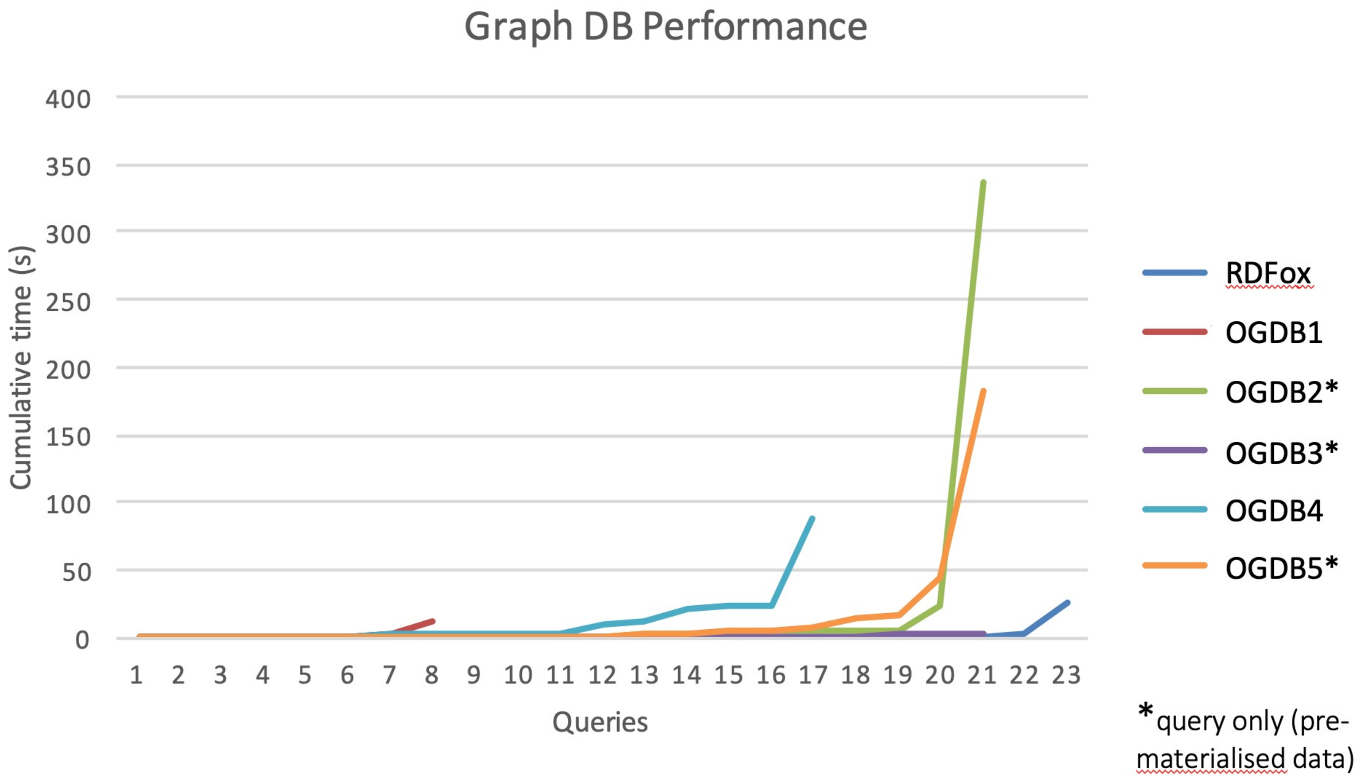
- **Novel algorithms developed at Oxford**
  - Proven correctness



*query only (pre-materialised data)

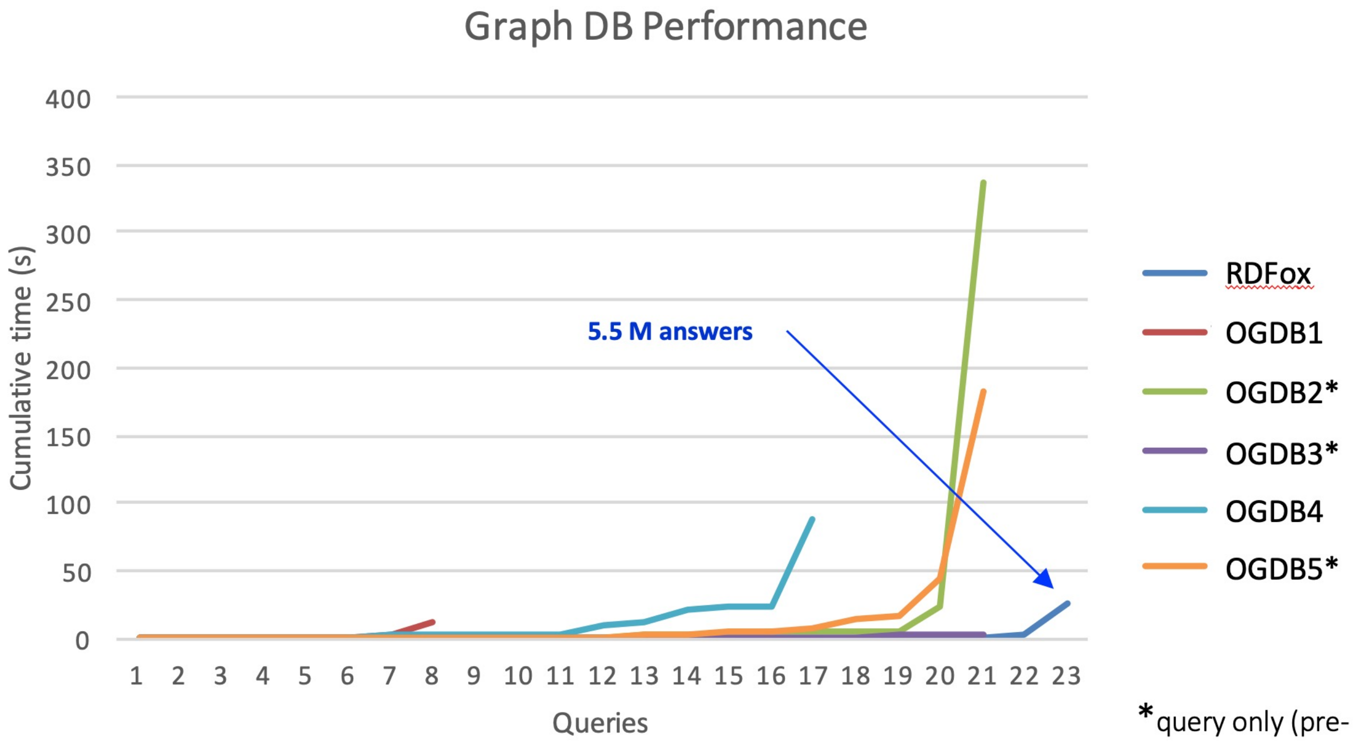
RDFOX

- **Novel algorithms developed at Oxford**
    - Proven correctness



RDFOX

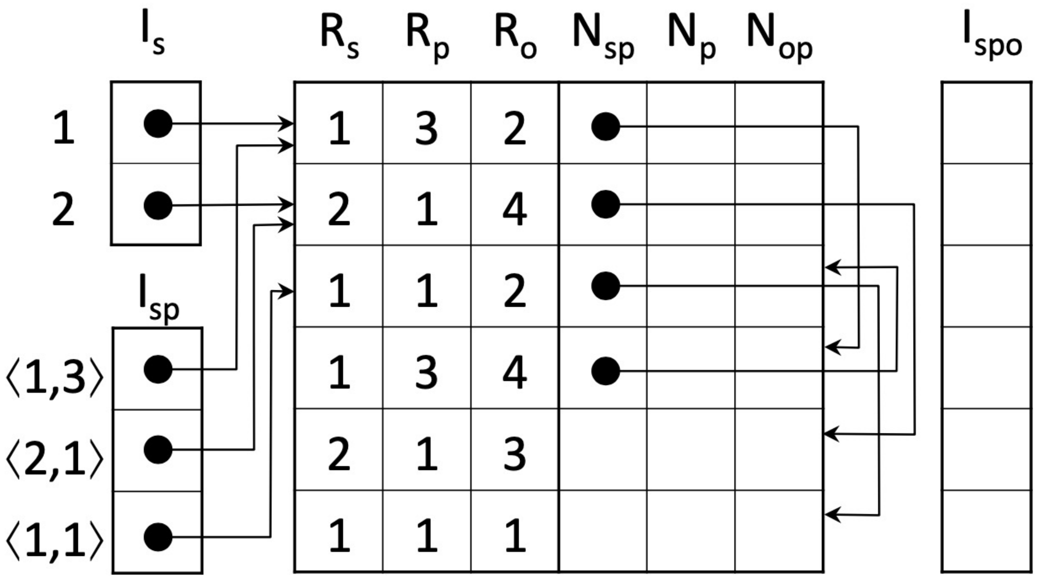
- **Novel algorithms developed at Oxford**
  - Proven correctness



*query only (pre-materialised data)

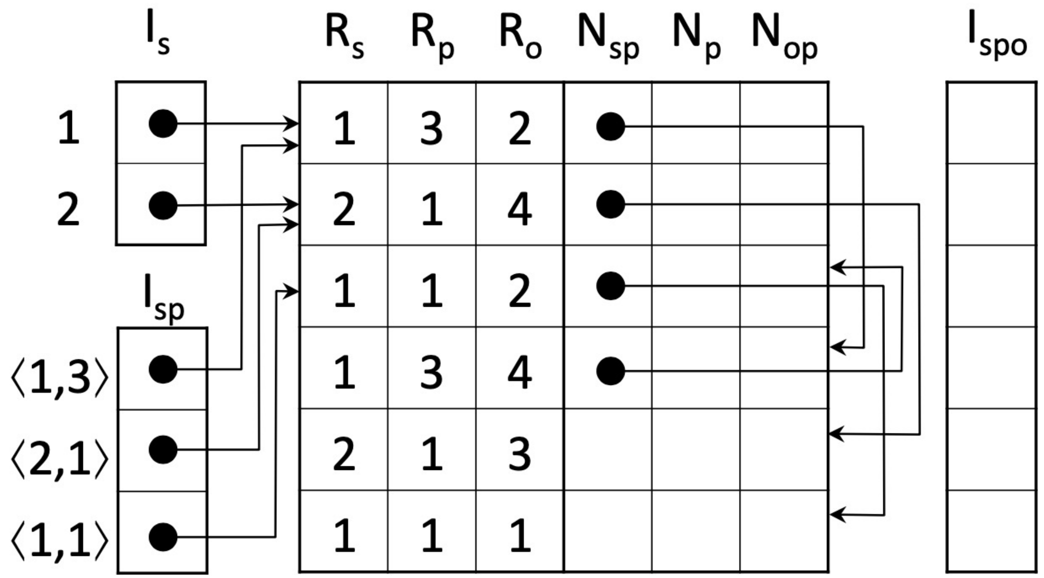
RDFOX

- **Novel algorithms developed at Oxford**
  - Proven correctness
- **Optimized in-memory data structures**
  - >10^9 triples on 128 Gb entry level server
  - >10^{10} triples on 1 Tb server



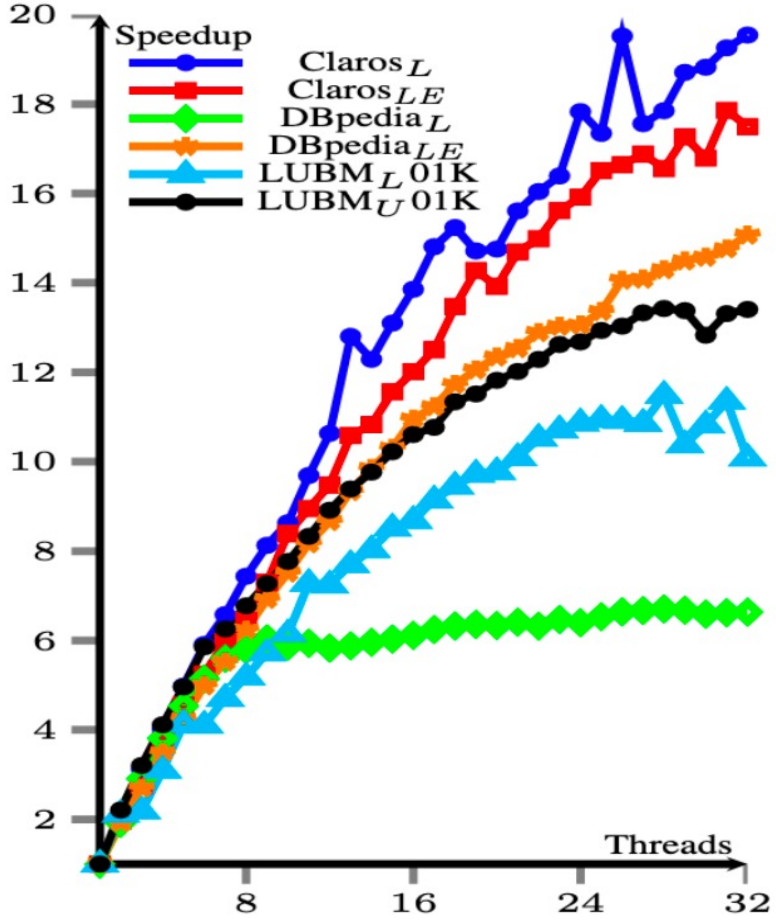
RDFOX

- **Novel algorithms developed at Oxford**
  - Proven correctness
- **Optimized in-memory data structures**
  - $>10^9$ triples on 128 Gb entry level server
  - $>10^{10}$ triples on 1 Tb server
- **Parallelised materialisation**
  - Dynamic distribution of workload
  - Mostly lock-free data structures



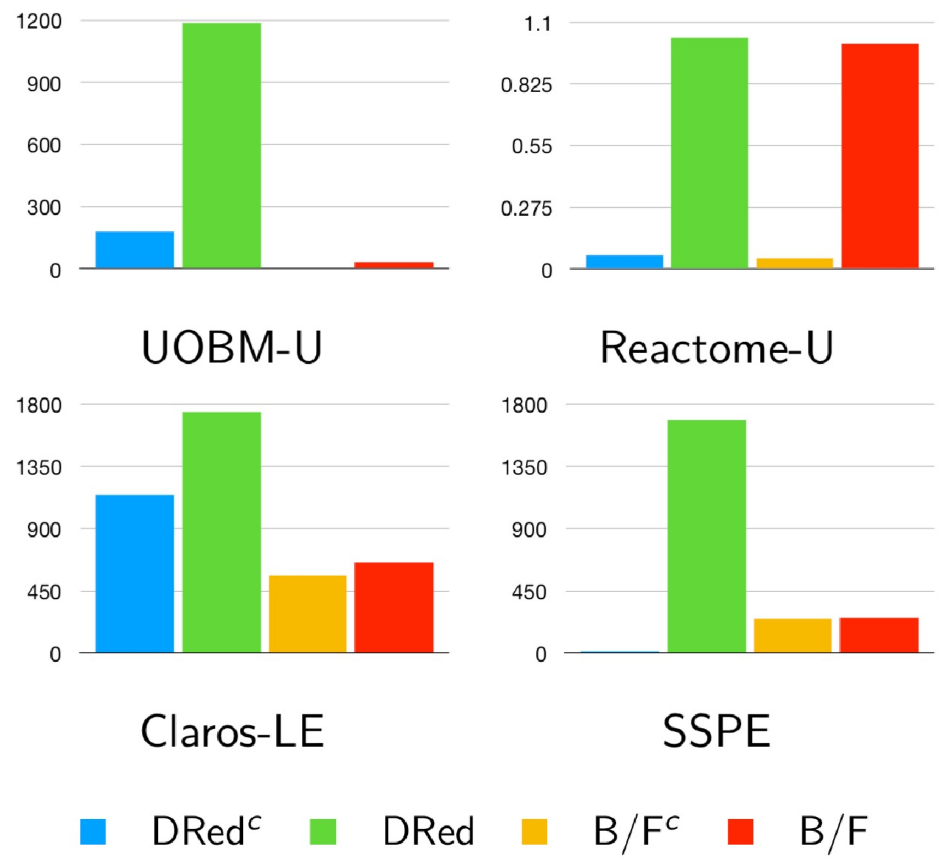
RDFOX

- **Novel algorithms developed at Oxford**
  - Proven correctness
- **Optimized in-memory data structures**
  - >10^9 triples on 128 Gb entry level server
  - >10^{10} triples on 1 Tb server
- **Parallelised materialisation**
  - Dynamic distribution of workload
  - Mostly lock-free data structures



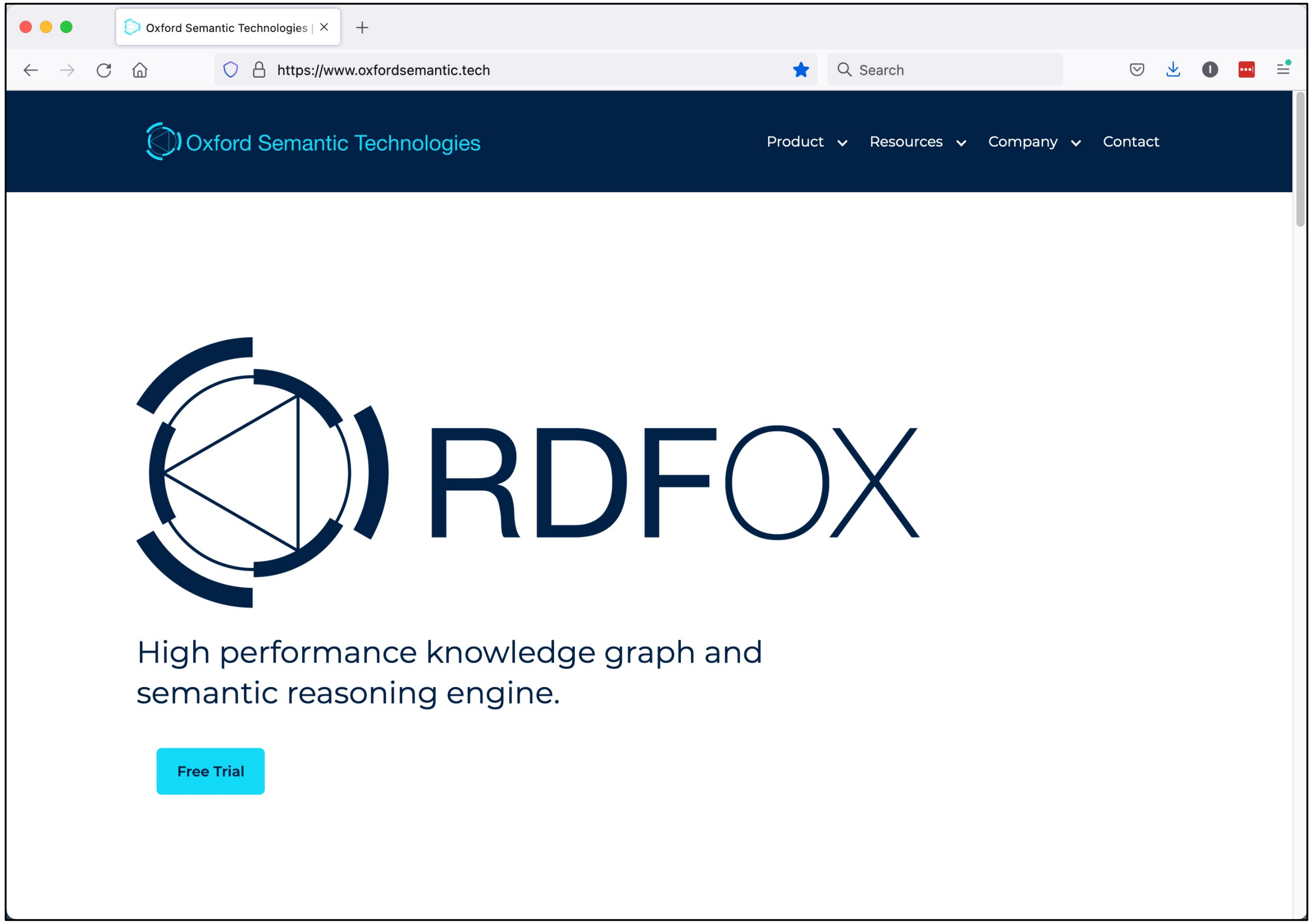
RDFOX

- **Novel algorithms developed at Oxford**
  - Proven correctness
- **Optimized in-memory data structures**
  - $>10^9$ triples on 128 Gb entry level server
  - $>10^{10}$ triples on 1 Tb server
- **Parallelised materialisation**
  - Dynamic distribution of workload
  - Mostly lock-free data structures
- **Incremental addition and retraction**
  - Novel B/F materialisation maintenance algorithm



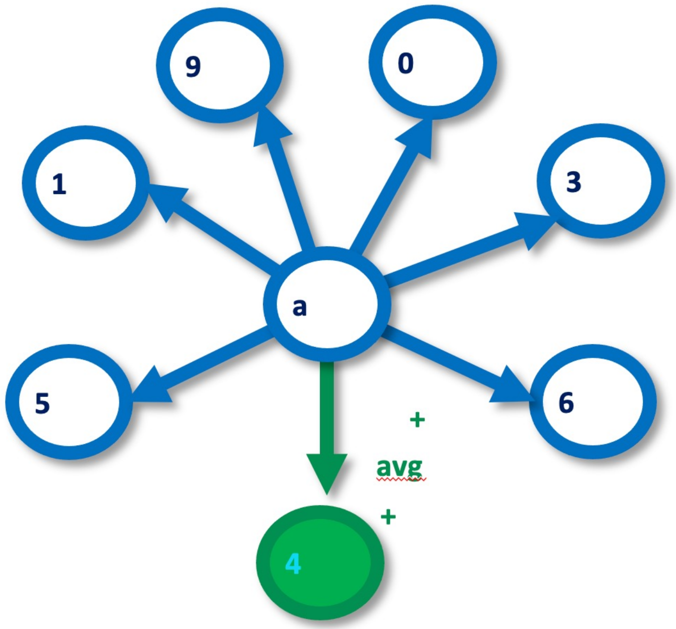
Challenges and Solutions (3)

Oxford Semantic Technologies

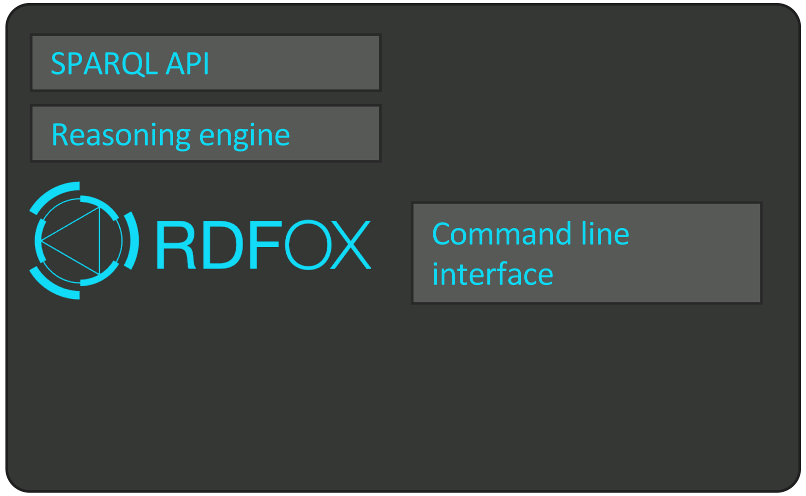


Extensions

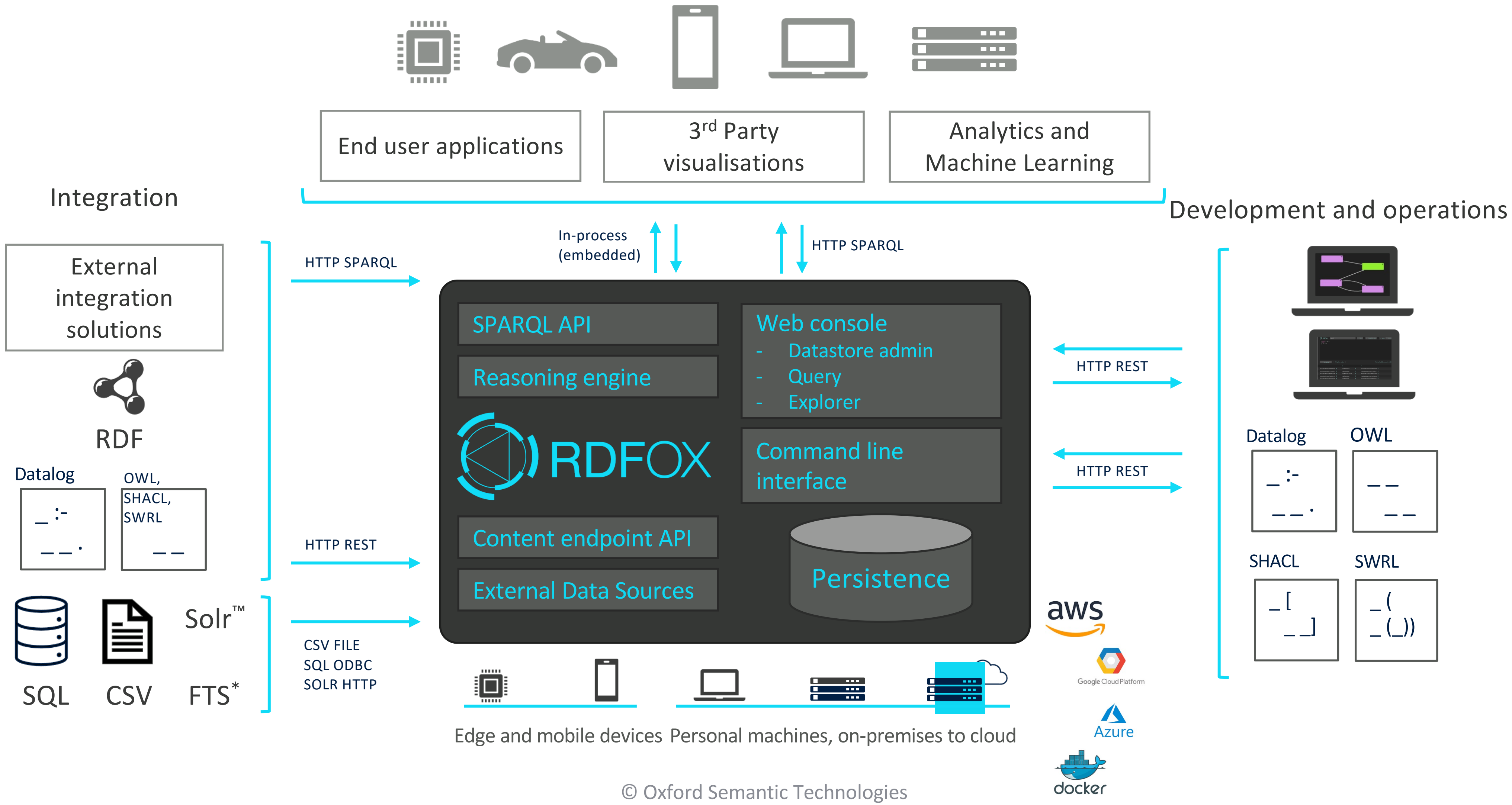
- Arbitrary rules
  - No restriction to OWL RL (tree-shaped) rules
- Data types and values
  - Numbers, strings, dates, …
  - Built in functions and aggregation
- Value invention
  - Add new (possibly computed) values to graph
  - Add new URI nodes to graph
- Constraints and negation as failure
  - SHACL+



System Architecture



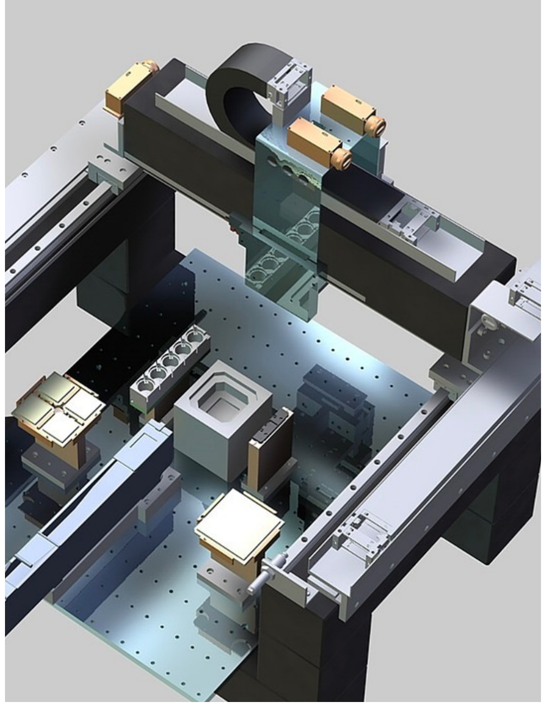
System Architecture

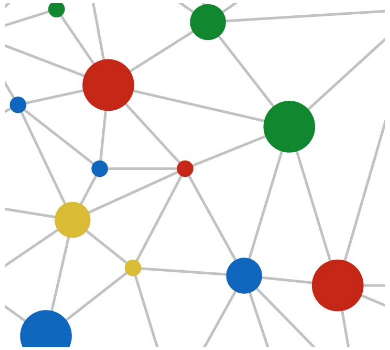


Knowledge Graph Use Cases

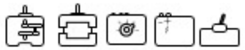
Configuration Management

FESTO

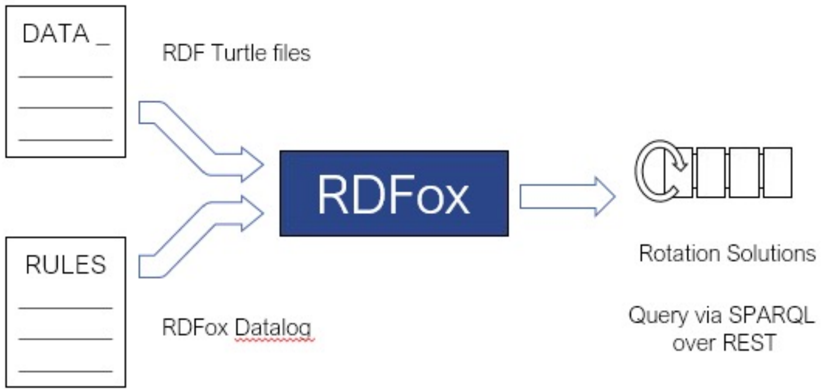


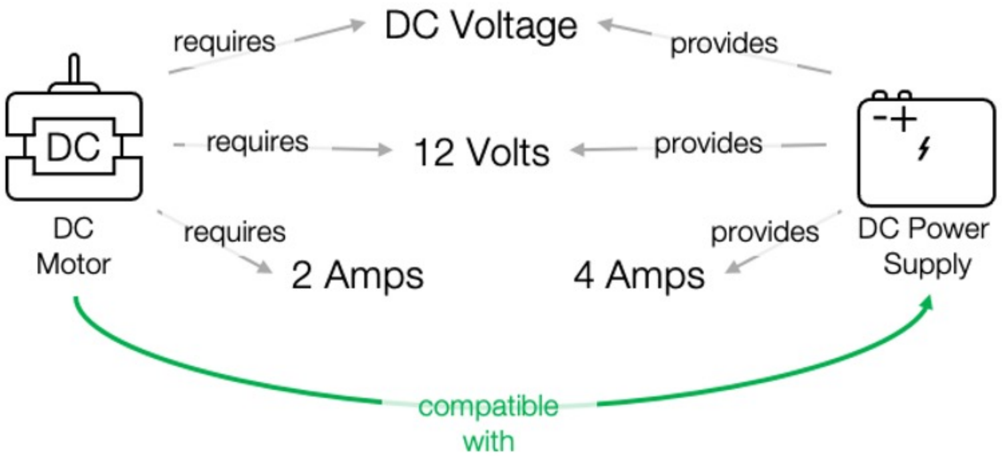


- Components
- Their attributes & constraints



- Definitions of compatibility &
- Valid configurations





Oxford Semantic Technologies

Wrap-up

Summary

- **KGs are powerful tool** for representing & reasoning about knowledge
- **Many applications**: configuration, data integration, compliance, …
- **Technical challenges**: complexity, scalability, extensions, systems, …
- **Solutions** based on foundational research + systems engineering

Thanks to Colleagues Collaborators and Funders

EPSRC
Engineering and Physical Sciences Research Council

EU

DARPA

SAMSUNG

SIEMENS

ebay

eDF

ORACLE

Thanks for Listening
Any Questions?

RDFOX : **www.oxfordsemantic.tech**

Background reading:

- **Description Logic:** Baader, Horrocks, Lutz, and Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.
- **OWL:** Horrocks, Patel-Schneider, and van Harmelen. *From SHIQ and RDF to OWL: The Making of a Web Ontology Language*. J. of Web Semantics, 1(1):7-26, 2003.
- **RDFox algorithms & data structures:** Motik, Nenov, Piro, Horrocks, and Olteanu. *Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems*. AAAI 2014.
- **Incremental maintenance:** Motik, Nenov, Robert Piro, and Horrocks. *Maintenance of datalog materialisations revisited*. Artificial Intelligence, 269:76-136, 2019.