Université Paris-Saclay LISN CNRS

Knowledge Graphs: an Introduction

Representation, reasoning, validation, Alignment, Linking and Trustworthiness

IA^2 – Autumn Institute of Artificial Intelligence, 19-23, October, 2026

Fatiha Saïs — Professor in Computer Science at Université Paris-Saclay / LISN

Outline

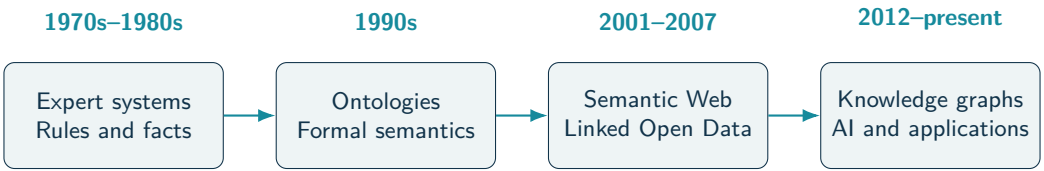
Knowledge Graphs: Fundamentals

Knowledge Graph Sharing and Quality

Knowledge Graphs Integration: some key problems

Temporal Fact Validation in Knowledge Graphs

Knowledge Graphs within Knowledge Representation Area of AI



Historical perspective

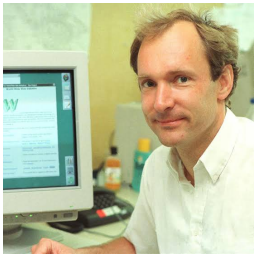
- Knowledge graphs build on classical AI knowledge bases.
- They represent knowledge as connected entities and relationships.
- Linked Open Data made it easier to share and connect knowledge across different sources.

Linked Open Data: Principles

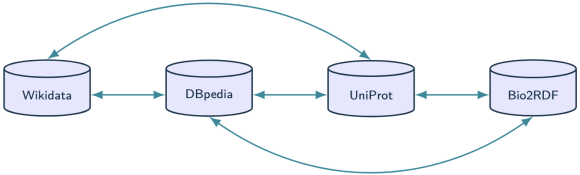
Linked Data Principles

- Use global HTTP IRIs to identify entities
- Describe resources using RDF
- Connect entities through typed links
- Reuse shared vocabularies





Tim Berners Lee



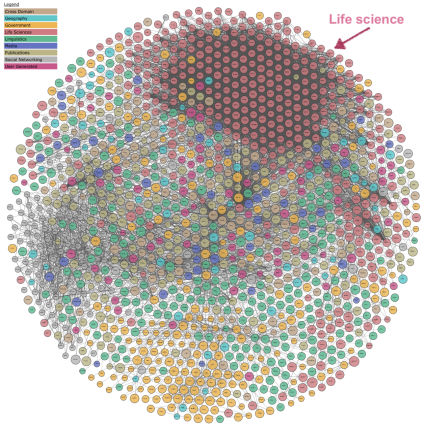
Key idea: Publish structured data that can be connected and reused across the Web.

Linked Open Data: Why It Matters

Why It Matters

- Connect data across different sources
- Reuse public datasets
- Share a common graph-based representation
- Support scientific and enterprise KGs

Key idea: Linked Open Data helped knowledge graphs grow beyond isolated sources.



WHO IS DEVELOPING KNOWLEDGE GRAPHS?

2007

DBpedia



2008

yago
select knowledge



2012

WIKIDATA



2018

ORKG
Open Research Knowledge Graph



2022

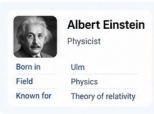
OpenAlex



Open / academic side

2012

Google
Knowledge Graph



2015

Microsoft
Graph



2016

LinkedIn
Economic Graph



2020s

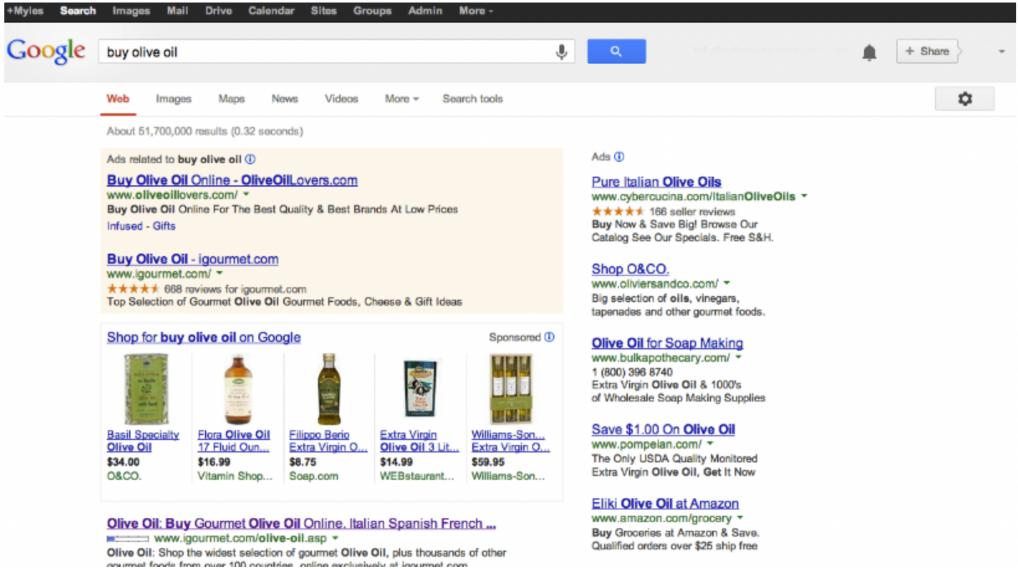
Enterprise knowledge graphs / AI assistants



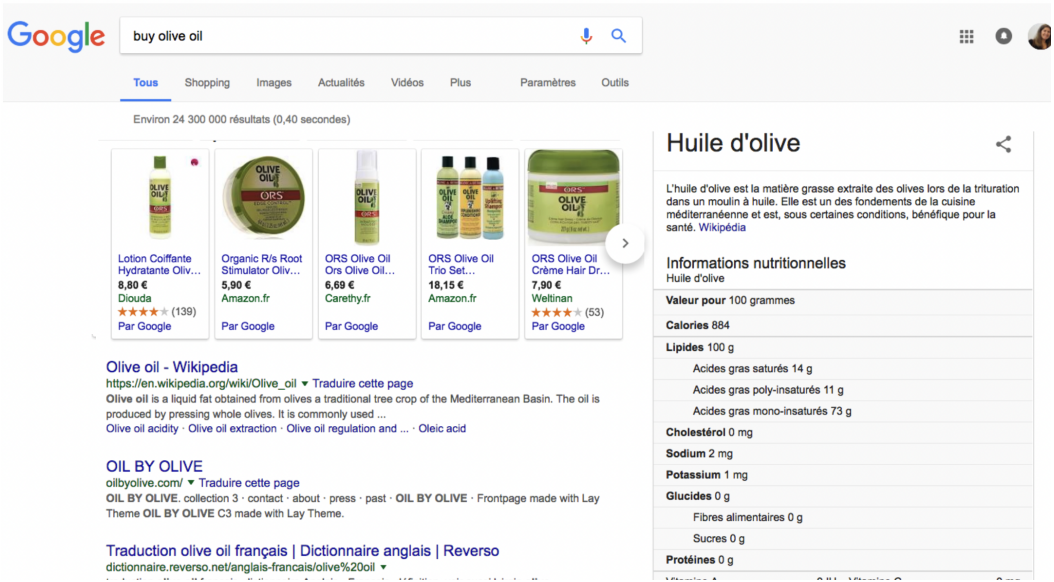
Commercial side

16

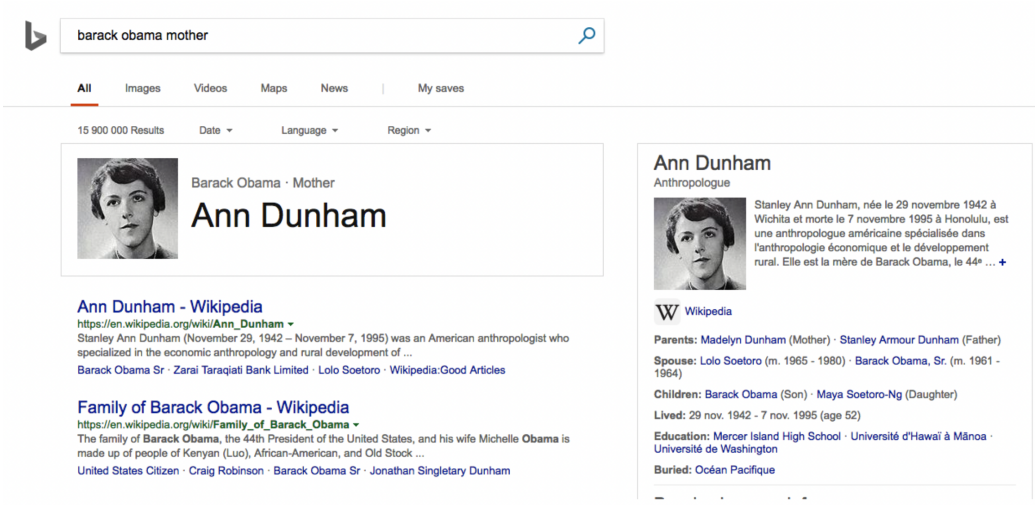
Web Search without Knowledge Graphs



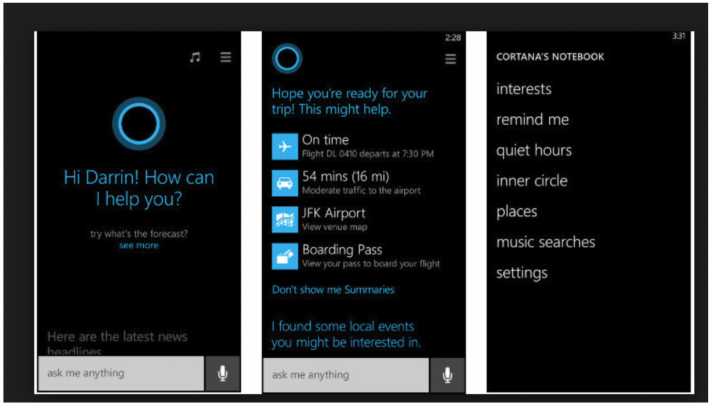
Web Search with Knowledge Graphs



Question Answering with Knowledge Graphs



Building a Knowledge-Powered AI Assistant



- Natural access and storage of knowledge
- Chat bots
- Personalization
- Emotion

Knowledge Graphs: Fundamentals

Université Paris-Saclay LISN CNRS

What is a Knowledge Graph?

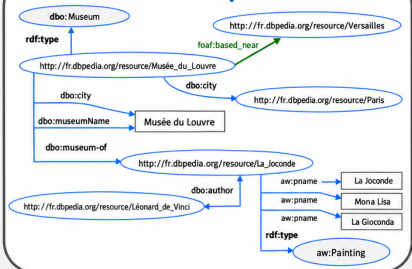
There is no single universally accepted formal definition of a Knowledge Graph.

Knowledge Graph - a definition

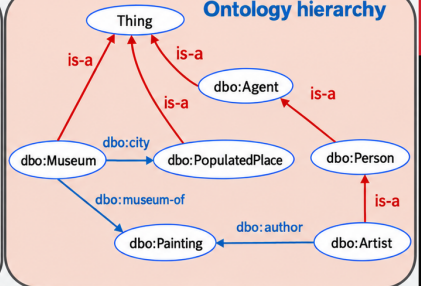
A Knowledge Graph is a graph-based representation of entities, their attributes and relations, enriched with explicit semantics and designed to support integration, querying, inference and/or machine learning.

KNOWLEDGE GRAPH (KG)

RDF Graphs



Ontology hierarchy



Querying (SPARQL)

```
PREFIX dbo: <http://dbpedia.org/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?m ?p
WHERE { ?m rdf:type dbo:Museum . ?m dbo:museum-of ?p .}
```

Reasoners: (Pellet, Fact++, Hermit, etc.)

KG saturation: infer whatever can be inferred from the KG.
KG consistency checking: no contradictions
KG repairing

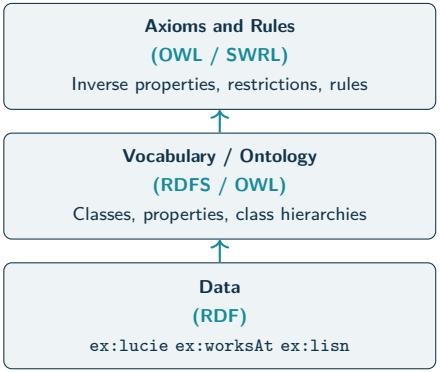
Ontology axioms and rules

```
owl:equivalentClass(dbo:Municipality, dbo:Place)
owl:equivalentClass(dbo:Place, dbo:Wikidata:Q532)
owl:equivalentClass(dbo:Village, dbo:PopulatedPlace)
owl:equivalentClass(dbo:PopulatedPlace,dbo:Municipality)
owl:disjointClass(dbo:PopulatedPlace, dbo:Artist)
owl:disjointClass(dbo:PopulatedPlace, dbo:Painting)
owl:FunctionalProperty(dbo:city)
owl:InverseFunctionalProperty(dbo:museum-of)

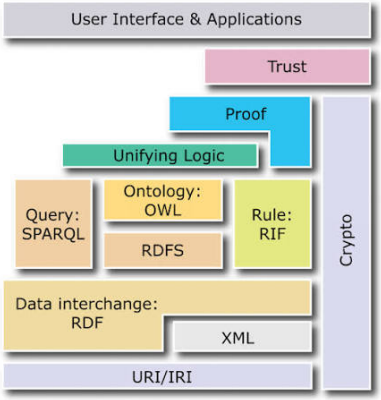
dbo:birthPlace(X, Y) => dbo:citizenOf(X, Y)
dbo:parentOf(X, Y) => dbo:child(Y, X)
```

25

Three Levels of Knowledge Representation



Semantic Web Layer Cake



A knowledge graph may contain facts without explicitly storing all their logical consequences.

Data: RDF – the triple data model

An atomic fact

$$(\textit{subject}, \textit{predicate}, \textit{object}) = (\texttt{ex:lucie}, \texttt{ex:worksAt}, \texttt{ex:lisn})$$



The subject and predicate identify resources; the object is a resource or a literal value.

An **RDF Graph** is a set of triples:

- Its nodes are (labelled by) the subjects and objects appearing in the triples
- Its edges are labelled by the properties (i.e., predicates)

.

Data: RDF serialization

Typical formats: Turtle, N-Triples, RDF/XML, JSON-LD.

Example of an RDF graph - Turtle

```
@prefix ex:  <http://example.org/> .
ex:lucie a ex:PhDStudent ;
   ex:worksAt ex:lisn .
ex:lisn a ex:Laboratory ;
   ex:partOf ex:upsaclay .
ex:upsaclay a ex:University ;
   ex:locatedIn ex:france .
```

- a abbreviates `rdf:type`.
- A full stop ends a statement; a semicolon reuses the same subject.

Data: RDF terms (URIs/IRIs, literals, blank nodes)

Term	Example	Role
URI/IRI	`ex:lucie`	Identify a resource
Literal	`"Lucie"@en`	Represent a value
Blank node	`_:b1`	Represent a locally identified resource without an IRI

Positions in an RDF 1.1 triple

Subject: IRI or blank node; predicate: IRI; object: IRI, blank node or literal.

Need of formal Knowledge

[Artificial Intelligence 47 (1991)]

ON THE THRESHOLDS OF KNOWLEDGE

Douglas B. Lenat
MCC
3500 W. Balcones Center
Austin, TX 78759

Edward A. Feigenbaum
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We articulate the three major fmdings of AI to date: (1) The Knowledge Principle: if a program is to perform complex task well, it must know a great deal about the world in which it operates. (2) A plausible extension of that principle, called the Breadth Hypothesis: there are two additional abilities necessary for intelligent behavior in unexpected situations: falling back on increasingly general knowledge, and analogizing to specific but far-flung knowledge. (3) AI as Empirical Inquiry: we must test our ideas experimentally, on large problems. Each of these three hypotheses proposes a particular threshold to cross, which leads to a qualitative change in emergent intelligence. Together, they determine a direction for future AI research.

opponent is Castling.) Even in the case of having to search

*The knowledge principle: "if a program is to perform a complex task well, **it must know a great deal about the world** in which it operates."*

The Well-Formedness Threshold: For each task, there is some minimum knowledge needed for one to even formulate it.

Ontology: A Definition

Definition

"An ontology is an **explicit, formal specification** of a **shared conceptualization**."

Thomas R. Gruber (1993)

- **Conceptualization:** An abstract model of a domain (concepts and relationships).
- **Specification:** A precise description of the concepts and relationships.
- **Explicit:** Concepts and their meanings are clearly defined.
- **Formal:** Expressed in a language that computers can interpret.
- **Shared:** Based on a common understanding agreed upon by a community.

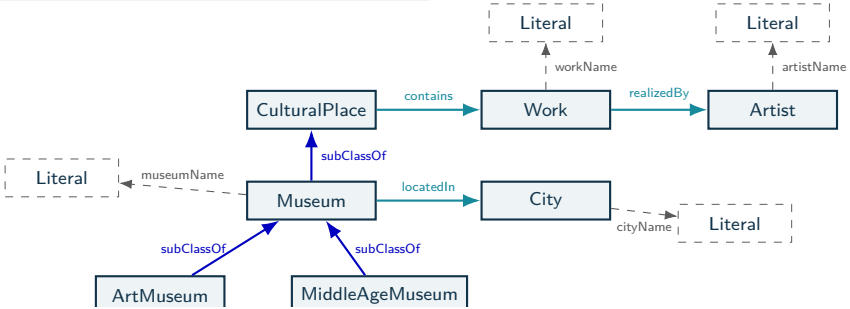
Main Components of an Ontology

Ontology Elements

- **Classes**: concepts or groups of individuals
- **Properties**:
  - Object properties (relations)
  - Data properties (attributes)
- **Individuals**: instances of classes

Axioms

- `rdfs:subClassOf`
- `owl:inverseOf`
- `owl:FunctionalProperty`
- Cardinality restrictions



RDFS vs. OWL

RDFS

- Class and property hierarchies
- Domain and range
- Lightweight inference

Syntax: RDF / Turtle

```
:Researcher
  rdfs:subClassOf :Person .

:worksAt
  rdfs:domain :Researcher ;
  rdfs:range :Laboratory .
```

OWL

- Equivalence and disjointness
- Restrictions and cardinalities
- Property characteristics

Syntax: Description Logic

$$\mathit{Researcher} \sqsubseteq \mathit{Person}$$

$$\mathit{Researcher} \sqsubseteq \exists \mathit{worksAt}.\mathit{Laboratory}$$

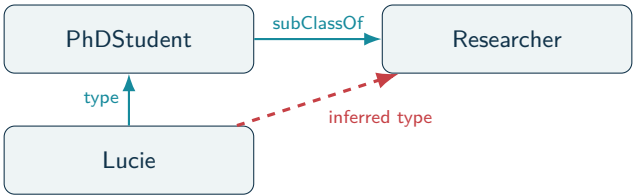
$$\mathit{Researcher} \sqsubseteq (\leq 1\ \mathit{worksAt}.\mathit{Laboratory})$$

Ontology axioms enable inference and consistency checking.

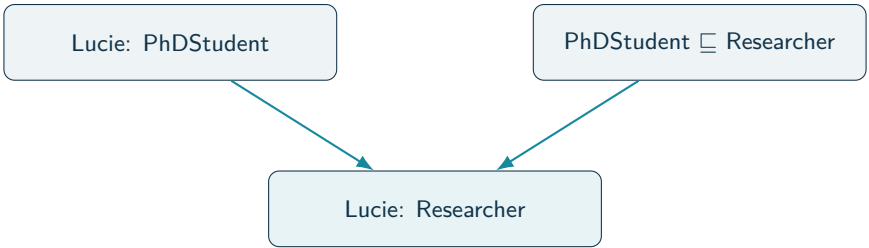
RDFS: subclass inference - Example

Schematic rule

$$x \texttt{ rdf:type } C \ \wedge \ C \texttt{ rdfs:subClassOf } D$$
$$\Longrightarrow \quad x \texttt{ rdf:type } D$$



Explaining inference: a proof chain



- A justification identifies *which facts and axioms* entail a conclusion.
- It can help diagnose inconsistency or source errors.

RDFS: Inferring types from a property - Example

RDFS ontology

```
ex:worksAt rdfs:domain ex:Person ;
        rdfs:range ex:Organization .
```

Asserted fact

```
ex:lucie ex:worksAt ex:lisn .
```

Entailed facts

```
ex:lucie a ex:Person .
ex:lisn a ex:Organization .
```

These types inferred from the ontology even when not explicitly stated in the graph.

OWL: defining classes using restrictions

OWL expressivity includes RDFS and

Description-logic definition

$$\text{UniversityResearcher} \equiv \text{Researcher} \sqcap \exists \text{worksAt}.\text{University}$$

- Lucie is a researcher.
- Lucie works at a resource that is a university.
- A reasoner can infer `UniversityResearcher(Lucie)`.

How to read the formula

$\sqcap$ is class intersection; $\exists R.C$ means "related through R to at least one instance of C."

Description Logics Behind OWL 2

OWL 2 DL: $\mathcal{SROIQ}(D)$

A highly expressive Description Logic supporting:

- Boolean class constructors: $\sqcap$, $\sqcup$, $\neg$
- Existential and universal restrictions: $\exists R.C$, $\forall R.C$
- Qualified cardinalities: $\geq n\, R.C$, $\leq n\, R.C$
- Inverse roles and property chains

Three OWL 2 Profiles

Profile	Main objective	Typical use
OWL 2 EL	Efficient ontology classification	Large biomedical ontologies
OWL 2 QL	Efficient query answering	Ontology-based data access
OWL 2 RL	Rule-based inference	Large RDF knowledge graphs

Key idea: Balance logical expressivity and reasoning efficiency.

OWL 2: Main Axioms and Entailment Patterns

Remark

The patterns below summarize the main **entailment rules** commonly illustrated in OWL 2 (especially in a rule-based view such as OWL 2 RL).

Axiom / pattern	**Entailment rule**
Subclass	$A \sqsubseteq B,\ x{:}A \ \Rightarrow\ x{:}B$
Equivalent classes	$A \equiv B,\ x{:}A \ \Rightarrow\ x{:}B \quad \text{and} \quad A \equiv B,\ x{:}B \ \Rightarrow\ x{:}A$
Intersection	$x{:}(A \sqcap B) \ \Rightarrow\ x{:}A,\ x{:}B$
Existential restriction	$x{:}\exists R.C,\ R(x,y) \ \Rightarrow\ y{:}C$
Universal restriction	$x{:}\forall R.C,\ R(x,y) \ \Rightarrow\ y{:}C$
Domain / range	$\mathrm{dom}(R) = A,\ R(x,y) \ \Rightarrow\ x{:}A \quad ; \quad \mathrm{range}(R) = B,\ R(x,y) \ \Rightarrow\ y{:}B$
Subproperty	$R \sqsubseteq S,\ R(x,y) \ \Rightarrow\ S(x,y)$
Inverse properties	$R \equiv S^{-},\ R(x,y) \ \Rightarrow\ S(y,x)$
Transitive property	$\mathrm{Trans}(R),\ R(x,y),\ R(y,z) \ \Rightarrow\ R(x,z)$
Symmetric property	$\mathrm{Sym}(R),\ R(x,y) \ \Rightarrow\ R(y,x)$
Functional property	$\mathrm{Func}(R),\ R(x,y),\ R(x,z) \ \Rightarrow\ y = z$
Inverse-functional property	$\mathrm{InvFunc}(R),\ R(x,z),\ R(y,z) \ \Rightarrow\ x = y$

OWL 2 adds expressive axioms whose consequences can derive new facts, types, relations, and equalities.

SPARQL: The RDF Query Language

- SPARQL is the standard language for querying graph data represented as RDF triples.
  - **SPARQL**: SPARQL Protocol and RDF Query Language
  - One of the core standards of the Semantic Web, alongside RDF and OWL
  - Became a W3C standard in January 2008
  - SPARQL 1.1 became a W3C Recommendation in March 2013

Types of SPARQL Queries

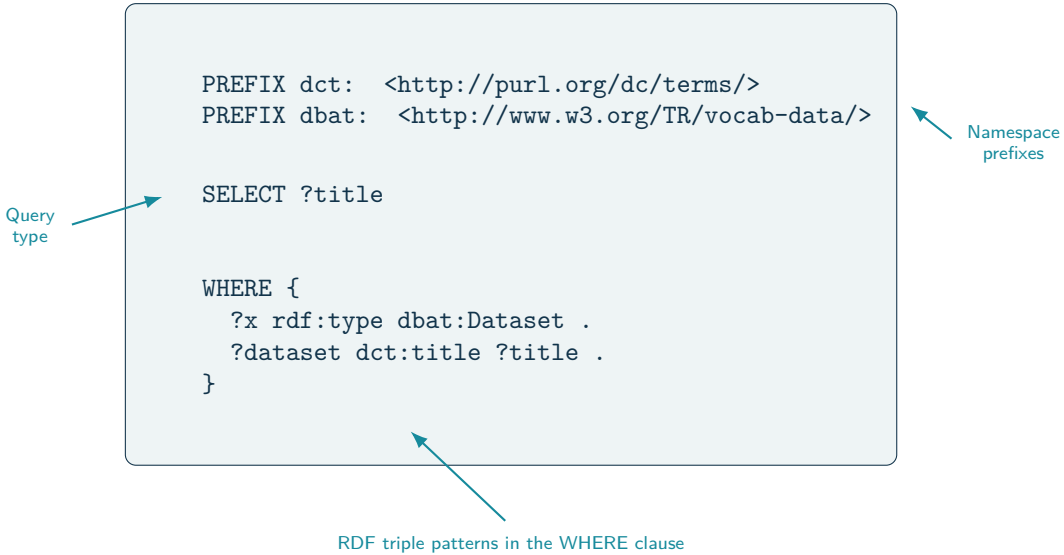
SELECT Returns a table of variable bindings that satisfy the query conditions.

CONSTRUCT Uses matched variables to generate RDF statements according to a specified template, producing a graph.

DESCRIBE Returns RDF statements that describe specified resources.

ASK Returns whether there is at least one solution satisfying the query conditions.

Structure of a SPARQL Query



SPARQL Example

Question: Who was born in Europe and received a Nobel Prize?

SPARQL query

```
SELECT ?x WHERE {
  ?y :hasName ?x ;
    :hasWonPrize :NobelPrize ;
    :bornIn :Europe .
}
```

RDF Triple Store

:Marie	:hasName	"Marie Curie"
:Marie	:hasWonPrize	:Prize1
:Prize1	a	:NobelPrize
:Brian	:hasName	"Brian P. Schmidt"
:Brian	:hasWonPrize	:Prize2
:Prize2	a	:NobelPrizeInPhysics

Highlighted match: Marie Curie's name, prize and birthplace.

SPARQL: Without Reasoning

Question: Who has received a Nobel Prize?

SPARQL Query

```
SELECT ?name WHERE {
  ?p :hasName ?name ;
     :hasWonPrize ?prize .
  ?prize a :NobelPrize .
}
```

RDF Triple Store

:Marie	:hasName	"Marie Curie"
:Marie	:hasWonPrize	:Prize1
:Prize1	a	:NobelPrize
:Brian	:hasName	"Brian P. Schmidt"
:Brian	:hasWonPrize	:Prize2
:Prize2	a	:NobelPrizeInPhysics

Answer:

Marie Curie

Brian P. Schmidt is not returned: his prize is not explicitly typed as NobelPrize.

SPARQL: More Answers Through Reasoning

Question: Who has received a Nobel Prize?

Same SPARQL Query

```
SELECT ?name WHERE {
  ?p :hasName ?name ;
     :hasWonPrize ?prize .
  ?prize a :NobelPrize .
}
```

Answers:

- **Marie Curie**
- **Brian P. Schmidt** (inferred)

Ontology and Reasoning

Ontology:

$$\texttt{:NobelPrizeInPhysics} \sqsubseteq \texttt{:NobelPrize}$$

Explicit fact:

$$\texttt{:Prize2 a :NobelPrizeInPhysics}$$

RDFS inference:

$$\texttt{:Prize2 a :NobelPrize}$$

Same query + RDFS reasoning = additional answers

Complementary purposes: OWL, SWRL and SHACL

	Purpose	Main question
RDFS/OWL	Derive consequences and check logical consistency	What follows from the axioms?
SWRL	Express conditional rules over individuals	If these facts hold, what follows?
SHACL	Validate a graph against constraints	Does the data meet the requirements?

SWRL: if–then rules on knowledge graphs

General syntax

$$A_1 \wedge \cdots \wedge A_n \longrightarrow B_1 \wedge \cdots \wedge B_m$$

Example: researcher affiliated with a university

```
Researcher(?x) ∧ worksAt(?x, ?l)
∧ partOf(?l, ?u) ∧ University(?u)
⟶ UniversityResearcher(?x)
```

- Variables (`?x`, `?l`, `?u`) range over individuals.
- The conclusion follows when the premises are satisfied.

SHACL: specifying expected data shapes

Application requirement

Every researcher must have a name and at least one affiliation to an organization.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:ResearcherShape a sh:NodeShape ;
  sh:targetClass ex:Researcher ;
  sh:property [ sh:path ex:name ;
                sh:minCount 1 ; sh:datatype xsd:string ] ;
  sh:property [ sh:path ex:worksAt ;
                sh:minCount 1 ; sh:class ex:Organization ] .
```

SHACL: interpreting a validation report

Graph under validation

```
ex:lucie a ex:Researcher .

ex:lucie ex:name "Lucie" .
```

No affiliation is provided.

SHACL result

Non-conformant: `sh:minCount 1` is violated for `ex:worksAt`.

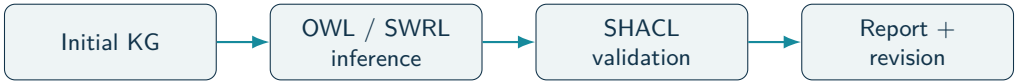
The report identifies the focus node, path and violation.

Contrast with OWL

Under the open-world assumption, a missing affiliation does not prove that Lucie has none. But SHACL can still flag **missing required data** in the graph being checked.

When should we use each formalism?

1. **RDFS/OWL:** classify entities, derive implicit facts and check logical consistency.
2. **SWRL:** express domain rules spanning multiple relationships.
3. **SHACL:** enforce structural and application-level requirements for available data.



This is one possible workflow; validation may also run before inference or directly on asserted triples.

Knowledge Graph Sharing and Quality

Université Paris-Saclay LISN CNRS

Sharing: From Linked Data to FAIR data

Linked Data asks:

- Can I identify it?
- Can I dereference it?
- Can I connect it?

FAIR asks:

- Is it Findable?
- Accessible?
- Interoperable?
- Reusable?

The two views are complementary: Linked Data gives Web architecture; FAIR emphasizes sustainable scientific/data-management practice.

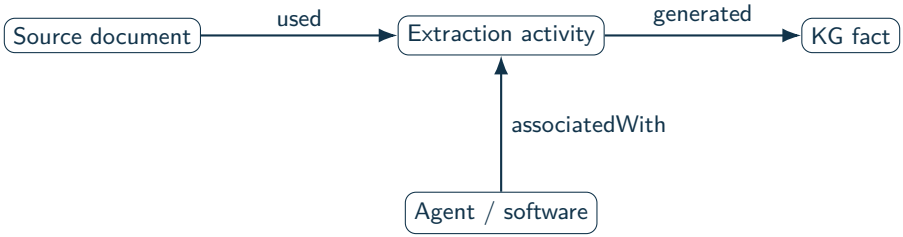
What should accompany a published KG?

- **Metadata**: title, creator, description, distributions.
- **License**: what may users legally do?
- **Provenance**: where did each dataset/fact come from?
- **Version**: what changed and when?
- **Quality documentation**: scope, known limitations, completeness and update policy.
- **Persistent identifiers**: can references survive system migration?

Useful W3C vocabularies include DCAT[1] for catalogs and PROV-O[2] for provenance.

[1] `https://www.w3.org/TR/vocab-dcat-3/`
[2] `https://www.w3.org/TR/prov-o/`

PROV-O: provenance as graph data



Provenance becomes crucial when KG content is automatically extracted, fused or generated by AI systems.

Data quality is not one dimension

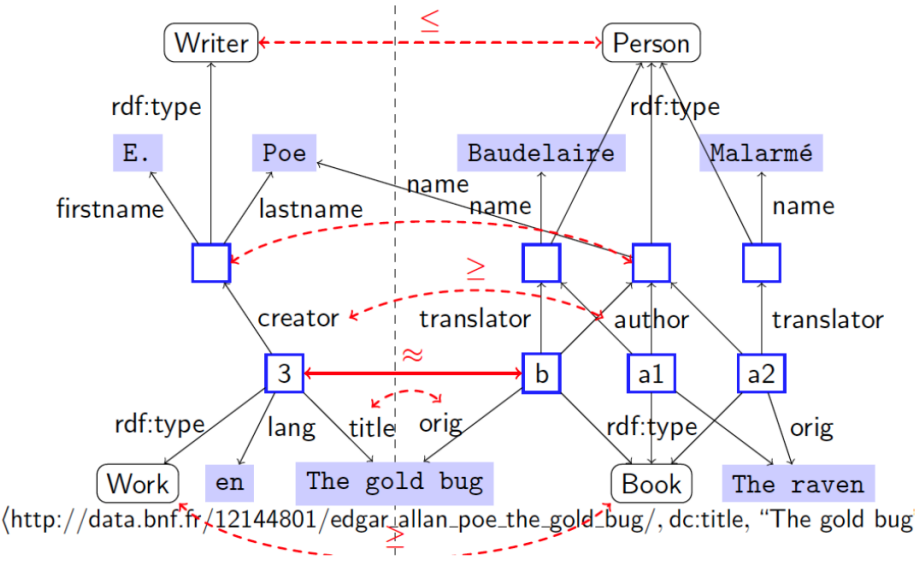
- **Syntactic validity**: can the data be parsed?
- **Semantic consistency**: do facts contradict domain axioms or known constraints?
- **Completeness**: are expected facts missing?
- **Timeliness**: are facts still valid now?
- **Trust / provenance**: can evidence be traced?
- **Interlinking quality**: are identity and semantic links correct?

This distinction prepares the transition to ontology alignment, link invalidation and temporal fact validation.

Knowledge Graphs Integration: some key problems

Université Paris-Saclay LISN CNRS

Entity and Ontology Alignment: the core problem



Ontology alignment problem

Given two ontologies O_1 and O_2, discover a set of correspondences

$$A = \{\langle e_1, e_2, r, c\rangle\}$$

where:

- $e_1 \in O_1$, $e_2 \in O_2$ are entities,
- r is a relation (e.g., equivalence, subsumption),
- c is a confidence score.

Alignment is therefore not only similarity computation: it also includes candidate generation, relation typing and mapping selection.

Sources of heterogeneity — with examples

Heterogeneity	Ontology O_1	Ontology O_2 / issue
Lexical	`Artwork`	`WorkOfArt`: synonym, different surface form
Abbreviation / language	`dateOfBirth`	`birthDate`, or `Musée` vs. `Museum`
Structural	`Exhibition` $\sqsubseteq$ `Event`	`Show` $\sqsubseteq$ `CulturalActivity`
Modeling choice	`hasCreator(Artwork,Person)`	intermediate node `CreationEvent` linking work and artist
Conceptual	`Museum`=organization	`Museum`=physical building
Granularity	one class `Researcher`	`PhDStudent`, `Postdoc`, `Professor`

Consequence

Ontology alignment must distinguish **different names for the same meaning** from **related but genuinely different meanings**.

Foundational taxonomy: Rahm & Bernstein, VLDB Journal, 2001; Euzenat & Shvaiko, Ontology Matching.

Evidence used by classical matchers — examples

Evidence	Example	What it suggests
Lexical	`birthDate` / `dateOfBirth`	token overlap after normalization
Dictionary / thesaurus	`Artwork` / `Work of art`	synonymy despite weak string overlap
Internal structure	both properties have range `xsd:date`	compatible datatype / constraints
External structure	parents `Event` / `CulturalEvent` already match	children become stronger candidates
Instances	two classes contain many already-linked instances	class meanings may overlap
Background KG	Wikidata/WordNet identifies two labels as same concept	external semantic evidence

Classical matching is already multi-signal: modern systems change the representations, not the need to combine evidence.

Cupid: Madhavan, Bernstein & Rahm, VLDB 2001; COMA: Do & Rahm, VLDB 2002.

Foundational matching systems and ideas

Work	Main signal	Foundational idea
Cupid (2001)	lexical + structural	combine names, datatypes, constraints and schema structure
Similarity Flooding (2002)	graph structure	propagate similarity over a product graph until a fixpoint
GLUE (2002)	instances + ML	learn concept mappings from class instances and taxonomic information
COMA (2002)	matcher combination	reusable library of matchers and strategies for combining scores

These works established several ideas still visible today: **multiple evidence sources, propagation, learning, and ensemble combination**.

Madhavan et al., VLDB 2001; Melnik et al., ICDE 2002; Doan et al., WWW 2002; Do & Rahm, VLDB 2002.

OAEI: evaluation remains essential

The Ontology Alignment Evaluation Initiative (OAEI) runs recurring evaluation campaigns across multiple tracks and domains.

OAEI 2025 campaign and results.

- common datasets and reference alignments,
- precision, recall and F-measure,
- robustness and runtime considerations,
- interactive, biomedical, KG and complex matching tracks.

Benchmarks are crucial because different methods excel on different heterogeneity profiles.

OAEI 2025: a changing ecosystem

The 2025 campaign includes long-standing logic-based systems and newer learning/LLM systems, including Agent-OM and LogMapLLM on several tracks.

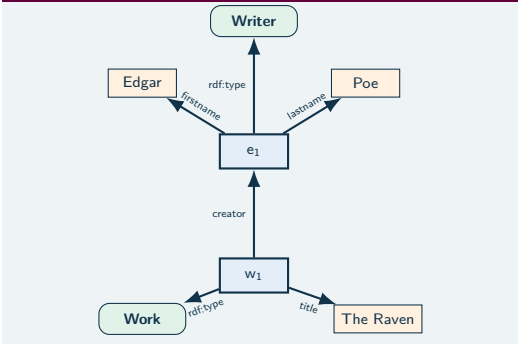
OAEI 2025 results.

Important lesson

There is no evidence that "LLM-only" matching simply replaces established matchers. Modern systems are increasingly **hybrid**: retrieval + language models + symbolic constraints + repair.

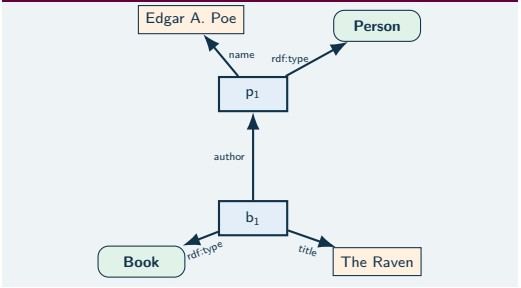
Entity Alignment Across Heterogeneous Knowledge Graphs

KG_1: BnF-like description



$\approx$

KG_2: DBpedia-like description



Alignment task

Identify correspondences between classes, properties and entities:

$$\texttt{Writer} \sqsubseteq \texttt{Person}, \quad \texttt{creator} \equiv \texttt{author}, \quad \texttt{title} \equiv \texttt{title}, \quad e_1 \equiv p_1, \quad w_1 \equiv b_1.$$

Entity alignment: identifying what facts are about

Problem

Identity link detection consists in deciding whether two descriptions of resources refer to the same real-world entity.

Examples

- Barack Hussein Obama II vs. Barack Obama
- President Obama vs. Barack Obama
- Harvard Law vs. Harvard Law School

Trust issue

Before validating a fact, we must know which entity the fact is about.

The same problem has many names

Database / data integration

- record linkage
- entity resolution
- duplicate detection
- object coreference

Knowledge Graphs / Semantic Web

- data linking
- entity alignment
- link discovery
- instance matching
- identity link detection

Terminology changes with communities; the decision problem is essentially the same.

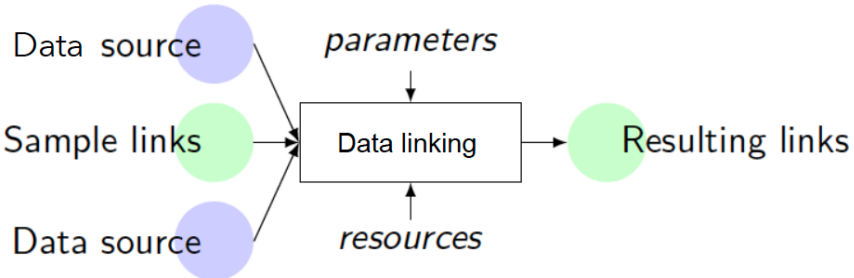
Record linkage: Newcombe et al., Science 1959; Entity Resolution tutorials: Getoor et al.

What evidence can a linker exploit?

- **Identifiers**: DOI, ISBN, ORCID, IATA, authority IDs
- **Attributes**: names, dates, addresses, descriptions
- **Rules / keys**: identifying or discriminative properties
- **External resources**: VIAF, Wikidata, GeoNames, domain KGs
- **Graph structure**: neighbors, relations, paths
- **Ontology knowledge**: disjointness, functionality, domains/ranges
- **Learned representations**: embeddings
- **Language-model evidence**: descriptions + contextual reasoning

Modern systems differ mainly in how they represent and combine these signals.

Entity Alignment Process



Identifiers, keys and rules

Strong identifiers

- DOI for publications
- ISBN for books
- ORCID for researchers
- IATA for airports

Rule example

$$homepage(x, u) \wedge homepage(y, u) \Rightarrow sameAs(x, y)$$

Atencia et al., EKAW 2012; Pernelle et al., JWS 2013; SAKey, ISWC 2014; VICKEY, ISWC 2017.

Key intuition

- one property may identify an entity;
- a combination may form a key;
- almost-keys can still be highly discriminative;
- keys can support blocking as well as matching.

String and attribute similarity

Typical ingredients:

- normalization, tokenization, stemming;
- Levenshtein / edit distance;
- Jaro–Winkler for short names;
- Jaccard / token overlap;
- cosine similarity for text vectors;
- numeric and date distances.

A structured matcher often aggregates property-level similarities:

$$Sim(x, y) = F\big(w_1 sim_1(x, y), \ldots, w_k sim_k(x, y)\big).$$

A good entity matcher usually compares compatible properties rather than flattening everything into one string.

Families of entity alignment approaches

Approach	Principle
Instance-based	Compare datatype properties and literal values.
Graph-based	Use object properties and similarity propagation in the graph.
ML-based	Use embeddings, GNNs, transformers or LLM-based matching and ranking.
Rule-based	Use logical patterns and key-like constraints to infer identity links.

Need

Alignment decisions should not be black boxes: they should be transparent and explainable.

Keys (rules) for entity alignment

Definition

A key is a set of properties that uniquely identifies every instance of a class in the data.

Entity	First name	Last name	Birth date
Person 1	Anne	Thompson	15/02/1988
Person 2	Marie	Thompson	02/09/1975
Person 3	Marie	David	15/02/1985
Person 4	Vincent	Solgar	06/12/1990

FirstName is not a key; **LastName** is not a key; **FirstName** + **LastName** may be a key.

Key discovery: different kinds of keys

Homogeneous graphs - almost-keys [Symeonidou et al. 2014, 2017]]

$$\forall b_1, b_2, a, t \left(Book(b_1) \wedge Book(b_2) \wedge author(b_1, a) \wedge author(b_2, a) \wedge title(b_1, t) \wedge title(b_2, t) \rightarrow b_1 = b_2 \right)$$

Heterogeneous graphs - linkkeys [Atencia et al. 2020]

$$LK(Book_1, Work_2) = \{\langle author_1, creator_2 \rangle, \langle title_1, title_2 \rangle\}$$

Other criteria

Keys may also differ on how they consider multi-valued properties and the absence of information in the KG.

Key discovery: a complex problem

Search space

Finding all minimal keys requires exploring combinations of properties.

- Exponential number of candidates
- Need for efficient pruning
- Incomplete and erroneous RDF data

Monotonicity

If a set of properties is a key, then all its supersets are also keys.

Minimal key

A key that stops being a key when any property is removed.

SAKey: scalable almost-key discovery

Motivation

Real KGs are incomplete and noisy: strict keys may be too restrictive.

Almost key

A set of properties that is not a perfect key only because of a limited number of exceptions.

Contribution

SAKey discovers almost keys in large RDF datasets using efficient pruning strategies.

For linking

Almost keys often improve recall while keeping high precision.

Joint work: Symeonidou, Armant, Pernelle and Saïs, ISWC 2014.

From keys to linkkeys [Atencia et al. 2020, Abbas et al. 2023, 2024]

Data interlinking across heterogeneous sources

When two KGs use different ontologies, alignment may require pairs of properties rather than a single common schema.



Idea

Linkkeys are maximal sets of property pairs that can link instances of two classes across two datasets.

Referring expressions for entity alignment

Definition

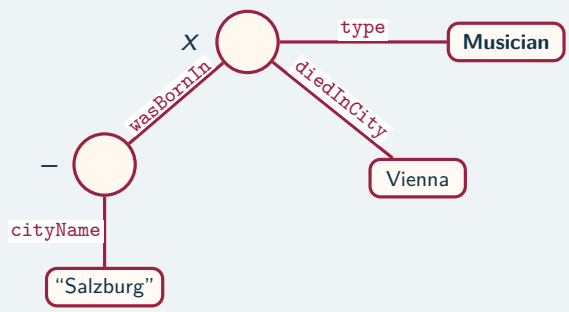
A referring expression is a description that uniquely identifies an instance in a given context.
They can explain why a description uniquely identifies an entity.

Simple example

The 44th President of the USA

$\Rightarrow$ Barack Obama

Graph-based example



RE-Miner: discovering minimal referring expressions

Input and output

- **Input:** a knowledge graph G and a class C
- **Output:** minimal referring expressions valid within C

1. Discover maximal non-key sets for C using SAKey.
2. Group candidate property sets by cardinality.
3. Construct candidate referring expressions and keep valid ones.
4. Increase the depth of the graph pattern when needed.
5. Replace IRIs by instantiations of minimal key properties in post-processing.

Joint work: Khajeh Nassiri, Pernelle, Quercini and Saïs, 2020–2023.

Data linking with referring expressions: protocol and result

Experimental hypothesis

If a description uniquely identifies an entity in one KG, it is likely that the same description identifies the same entity in another KG.

Protocol

- Generate REs for entities in YAGO.
- Test whether the same description is fulfilled by a unique DBpedia entity.
- Compare with keys
- Use strict string equality for literals.

Result

The approach produces identity links supported by explicit descriptions and was ranked first on SPIMBENCH at IM-OAEI 2020.

The explanation is the referring expression itself.

Temporal Fact Validation in Knowledge Graphs

Université Paris-Saclay LISN CNRS

From Knowledge Graphs to Temporal Knowledge Graphs

Standard fact

$\langle Obama, workedAt, SidleyAustinLLP \rangle$

Temporal fact

$\langle Obama, workedAt, SidleyAustinLLP, [1991, 1991$

Why it matters

Many facts are not timeless:

- jobs and affiliations
- political positions
- regulations and policies
- scientific claims and collaborations

Temporal question

Is a fact valid now, valid in the past, outdated or temporally inconsistent?

Temporal reasoning in KGs

Example question

Who is the Prime Minister of Nepal?

- Correct answers may change over time.
- Sources may be outdated, contradictory or temporally underspecified.

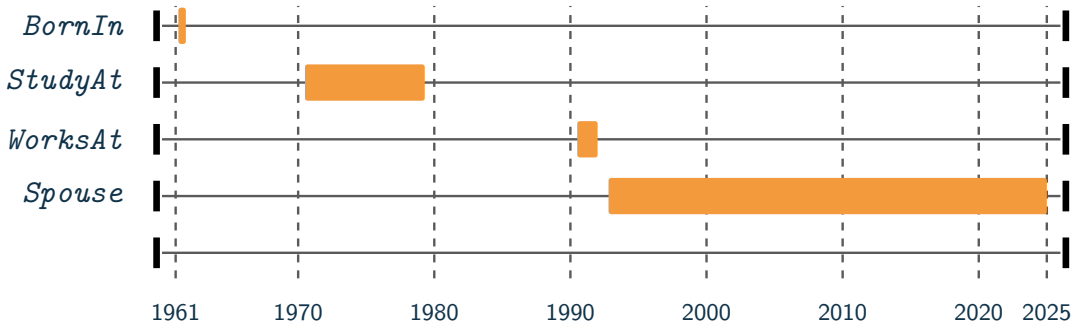
Message

Poor temporal reasoning can produce wrong answers even when the entity and relation look correct.

Temporal validity is a key dimension of trust.

Temporal Constraint Discovery: An Entity Timeline View

BarackObama:



Intuition

Temporal facts can be compared through their validity intervals to detect inconsistency.

Temporal constraints for fact validation [Soulard et al. 2025]

Goal

Discover constraints that capture how temporal relations between facts should behave.

Simple constraints

- birth before death
- study before diploma
- employment before affiliation end

Complex constraints

- overlapping intervals
- temporal exclusion
- temporal ordering between roles

Contribution

Temporal constraints provide interpretable evidence for accepting, rejecting or flagging facts.

Temporal constraint discovery: from entities to classes

Entity-level view

Observe temporal relations between facts for a given entity.

Class-level generalization

Generalize recurring temporal patterns to classes and relations.



Applying temporal constraints: symbolic decision

Input fact

$\langle Obama, servedAs, U.S.Senator \rangle$

with a queried time interval

Decision

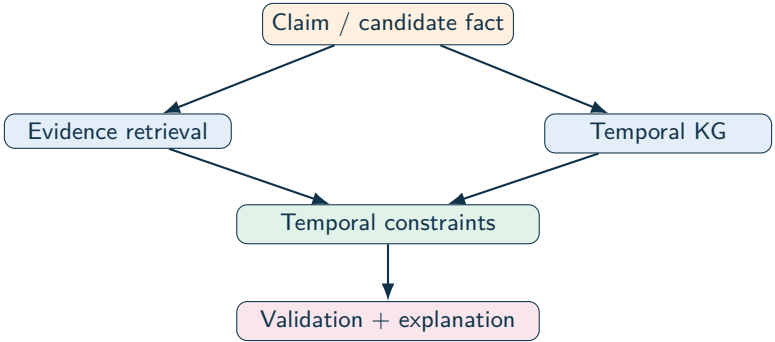
- valid
- invalid
- uncertain
- temporally underspecified

Explanation

The decision can be justified by the violated or satisfied temporal constraints and by the facts used as evidence.

Joint work: Soulard et al., temporal constraint discovery and fact validation.

Hybrid temporal fact validation



Message

Hybrid approaches combine evidence retrieval, temporal KGs and symbolic constraints for explainable validation.

Temporal constraint application: experiments

Experimental setting

Temporal constraints are applied to facts to validate and encoded as features for decision making.

Value	Meaning
0	Satisfied temporal constraint
1	Violated temporal constraint
2	Temporal constraint not about the fact to validate
3	Sequence is not temporally functional
4	No data on the entity or relation

Result

The hybrid approach is more effective than the pure symbolic decision algorithm.

Temporal constraint learning: first results

Constraint acquisition

Temporal constraints are modeled as a temporal constraint network using Allen's interval algebra and empirical confidence scores.

- Many temporal relations are weakly supported or never observed before propagation.
- Dataset density strongly impacts inference quality: denser classes support higher-confidence propagation.
- These first results support hybrid and constraint-based approaches to temporal fact validation.

Message

Constraint learning and propagation extend temporal coverage beyond observed facts while preserving interpretability.

Take-Home Messages

Knowledge Graphs: a foundation for Knowledge-driven AI

Knowledge Graphs provide an explicit, structured and semantically rich representation of knowledge, supporting integration, querying and reasoning.

What Knowledge Graphs Enable

- Connect heterogeneous data sources
- Represent entities and relationships
- Infer implicit knowledge
- Support semantic querying
- Enable knowledge reuse

What Makes Them Reliable

- Ontologies and logical axioms
- Rules and constraints
- Entity and ontology alignment
- Data quality and validation
- Provenance and explainability

The future of AI lies in combining learning capabilities with explicit, structured and verifiable knowledge.

Thank you! `sais@lisn.fr`