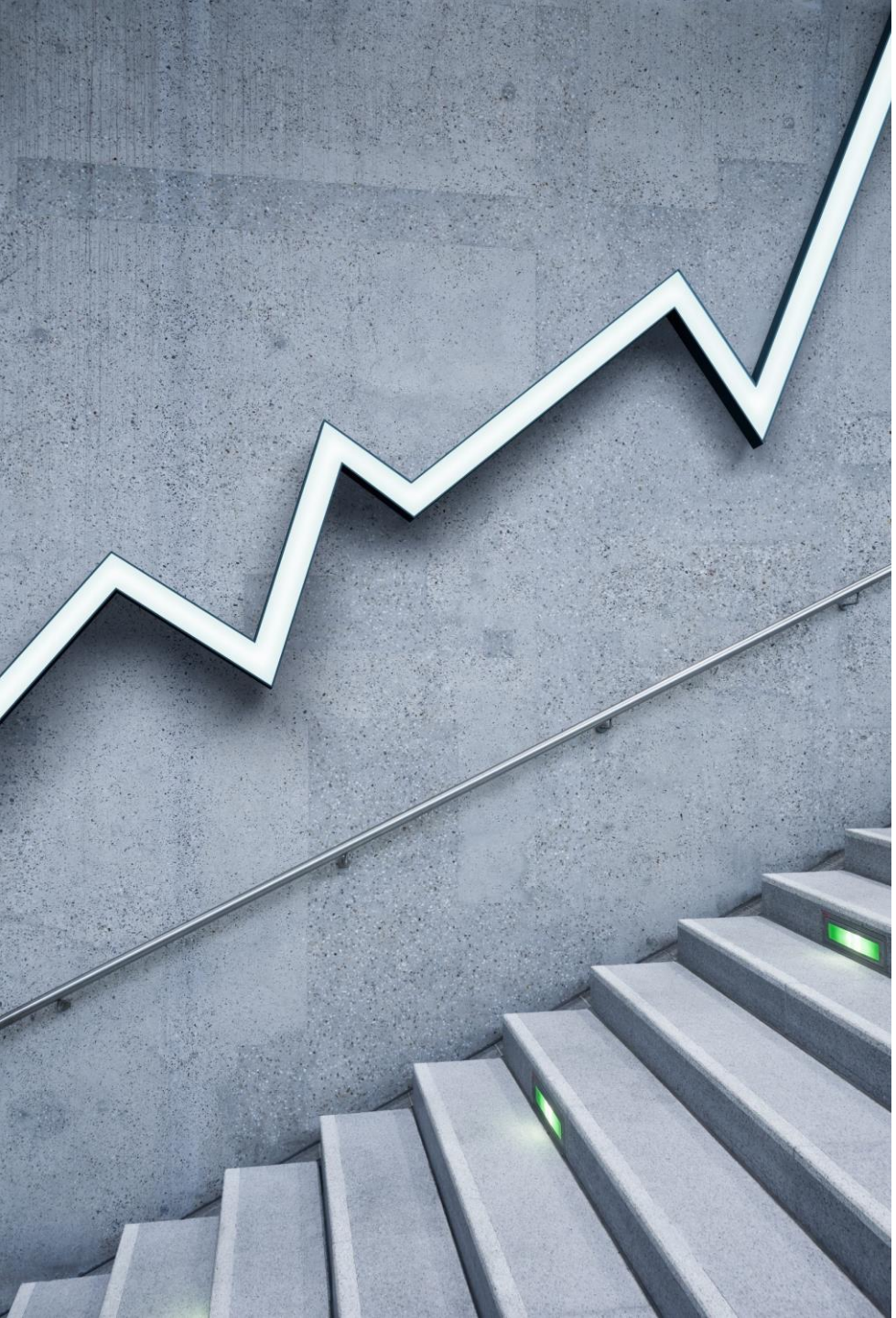


The ontology

ABDELBASSET B

The lack of the current knowledge representation techniques

Let’s imagine that we want to model a road traffic management system for a smart city.

Several applications must cooperate:

1. a traffic monitoring system
2. a parking management system
3. an emergency services system

The lack of the current knowledge representation techniques

We could have two systems A and B that manipulate different concepts

Car: speed, lane
Road: length
Sensor: count

System A

Vehicle: velocity
Street: size
Detector: flow

System B

Each application must implement its own rules:

```
If (type.equals("ambulance")) {
    givePriority();
}
```

```
If (type.equals("EmergencyCar")) {
    givePriority();
}
```

The lack of the current knowledge representation techniques

Problems appear:

- Rules are duplicated
- Semantics are hard-coded
- Any change requires modifying all programs
- Systems cannot evolve easily

The Ontology

Ontology Definitions

1. Neches et al. (1991)

"An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary."

Ontology Definitions

2. Gruber (1993)

“An ontology is an explicit specification of a conceptualization.”

Ontology Definitions

3. Borst (1997)

“An ontology is a formal specification of a shared conceptualization.”

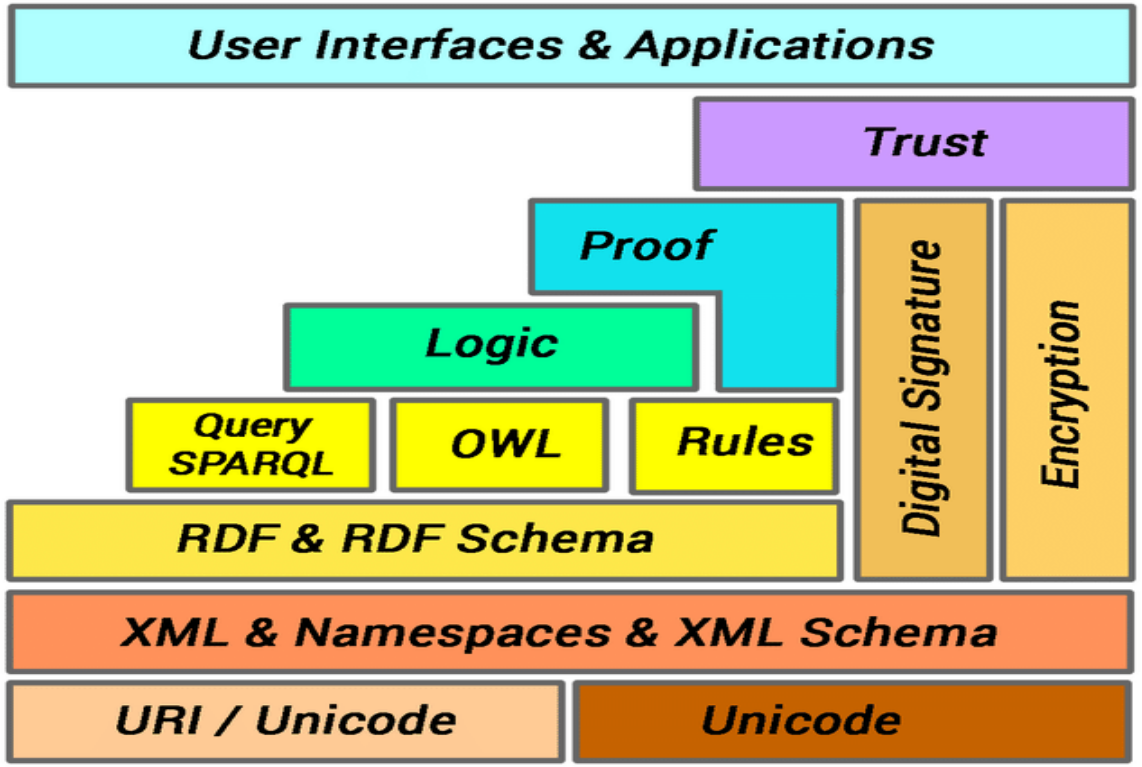
The Semantic Web

Tim Berners-Lee (W3C) – Classic Definition

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

— *Tim Berners-Lee, James Hendler, Ora Lassila (2001)*

The Semantic Web



Description Logic (DL)

Why not simply use FOL for ontologies?

With FOL you can do everything, but...

> You could also program everything with assembler instead of higher programming languages.

FOL has high expressivity.
FOL is too bulky for modeling.
FOL is not appropriate for finding consensus in modeling.
FOL proof theoretically is very complex (semi-decidable).

Description Logic (DL)

Basic idea:
Look for an appropriate fragment of FOL...
...and then make it a vocabulary for RDF(S)

Description Logics (DLs)

- DLs are fragments of FOL (a compromise between expressivity and scalability)
- A DL models concepts, roles, individuals, and their relationships.
- In DLs, simpler descriptions are created with the help of constructors.
- DLs differ in the constructors applied (expressivity)

DL Knowledge Base

DL Knowledge Base (KB) normally separated into two parts:
KB= {Tbox, Abox}

TBox (Terminological Box) is a set of axioms in the form of ($C \sqsubseteq D$, $C \equiv D$) describing structure of domain (i.e., schema),

Example:

$\text{Elephant} \sqsubseteq \text{Animal} \sqcap \text{Large} \sqcap \text{Grey}$

ABox (Assertion Box) is a set of axioms in the form of (C(a), R(a, b)) describing a concrete situation (data)

Example:

Father (Med) hasChild(Med, Ali)

ALC: Attributive Language with Complements DL

ALC is the most fundamental and widely studied description logic (DL), often considered the "standard" or "prototypical" DL. It forms the basis for more expressive description logics and corresponds to a fragment of first-order logic.

ALC: Attributive Language with Complements DL

Core Constructors of ALC

Atomic concepts (e.g., Human, Animal, A, B….0)

Top concept : T

Bottom concept: ⊥

Atomic roles (e.g., hasChild, writes)

ALC constructors:

Conjunction (⊓): Human ⊓ Male (human and male)

Disjunction (⊔): Doctor ⊔ Lawyer (doctor or lawyer)

Negation (¬): ¬Student (not a student)

Existential restriction (∃) ∃ R.C : ∃hasChild.Doctor (has at least one child who is a doctor)

Universal restriction (∀) **∀ R.C** : ∀hasChild.Doctor (all children are doctors)

ALC: Attributive Language with Complements DL

In addition to that the Tbox of ALC DL contains the two axioms

Class Inclusion

Novel $\sqsubseteq$ Book // Every novel is also a book

equals FOL $(\forall x)$ (Novel(x) $\rightarrow$ Book(x))

Class Equivalence

Mother $\equiv$ Woman $\sqcap$ Parent //a mother is a woman and a parent in the same time

equals FOL $(\forall x)$ (Mother(x) $\leftrightarrow$ (Woman(x) and Parent(x))

Example Domain: University

We model a university with students, courses, and teachers.

Identifying the Vocabulary

Atomic Concepts

- Person
- Student
- Teacher
- Course
- GraduateStudent

Roles

- teaches
- enrolledIn

TBox (Terminological Box) (Terminological Knowledge)

$\text{Student} \sqsubseteq \text{Person}$	Every student is a person
$\text{Teacher} \sqsubseteq \text{Person}$	Every teacher is a person

$\text{Student} \sqsubseteq \exists\ \text{enrolledIn}.\text{Course}$

Every student is enrolled in at least one course

$\text{Teacher} \sqsubseteq \forall\ \text{teaches}.\text{Course}$

Teachers only teach courses

Example Domain: University

We model a university with students, courses, and teachers.

Identifying the Vocabulary

Atomic Concepts

- Person
- Student
- Teacher
- Course
- GraduateStudent

Roles

- teaches
- enrolledIn

ABox (Assertion Box)

Individuals

Med
Aya
KRR

Basic assertions

Student(Med)
Teacher(Aya)
Course(KRR)

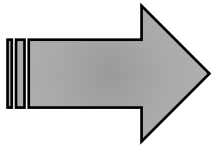
Role assertions

enrolledIn(Med, KRR)
teaches(Aya, KRR)

Example Domain: University

Inferred Types

Student(Med)
Student $\sqsubseteq$ Person



Person(Med)

Example Domain: University

Examples of Increasing Complexity

Teaching Student (new concept)

A student who teaches at least one course

$TeachingStudent \equiv Student \sqcap \exists\ teaches.Course$

Non-Teaching Student (new concept)

A student who does not teach any course

$NonTeachingStudent \equiv Student \sqcap \neg\exists\ teaches.Course$

RDF: Resource Description Framework

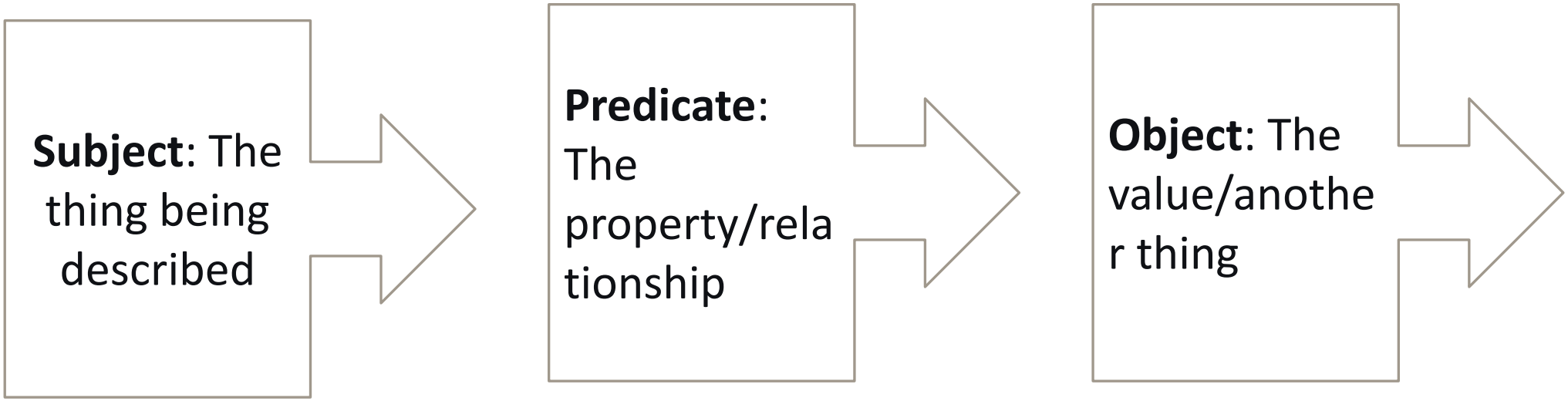
Core Concept: The "Lego of Data"

RDF is the fundamental **data model** for the Semantic Web. Think of it as **universal connectors** that let you link any piece of information to any other piece, creating a giant global graph of data.

RDF: Resource Description Framework

Key Metaphor: "Subject-Predicate-Object" Statements

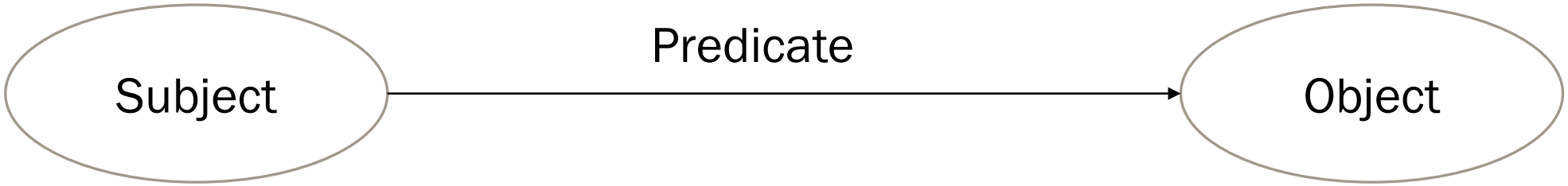
RDF works like **simple sentences** that both humans and machines can understand: The RDF triple



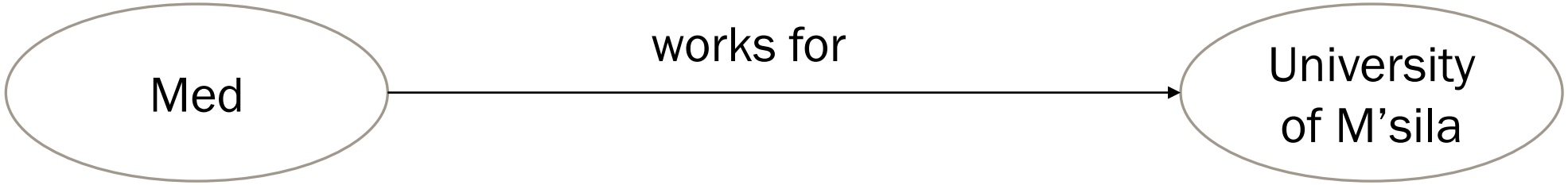
Example in plain English: Med works for the University of M’sila

RDF: Resource Description Framework

Key Metaphor: Directed graphs with labeled edges

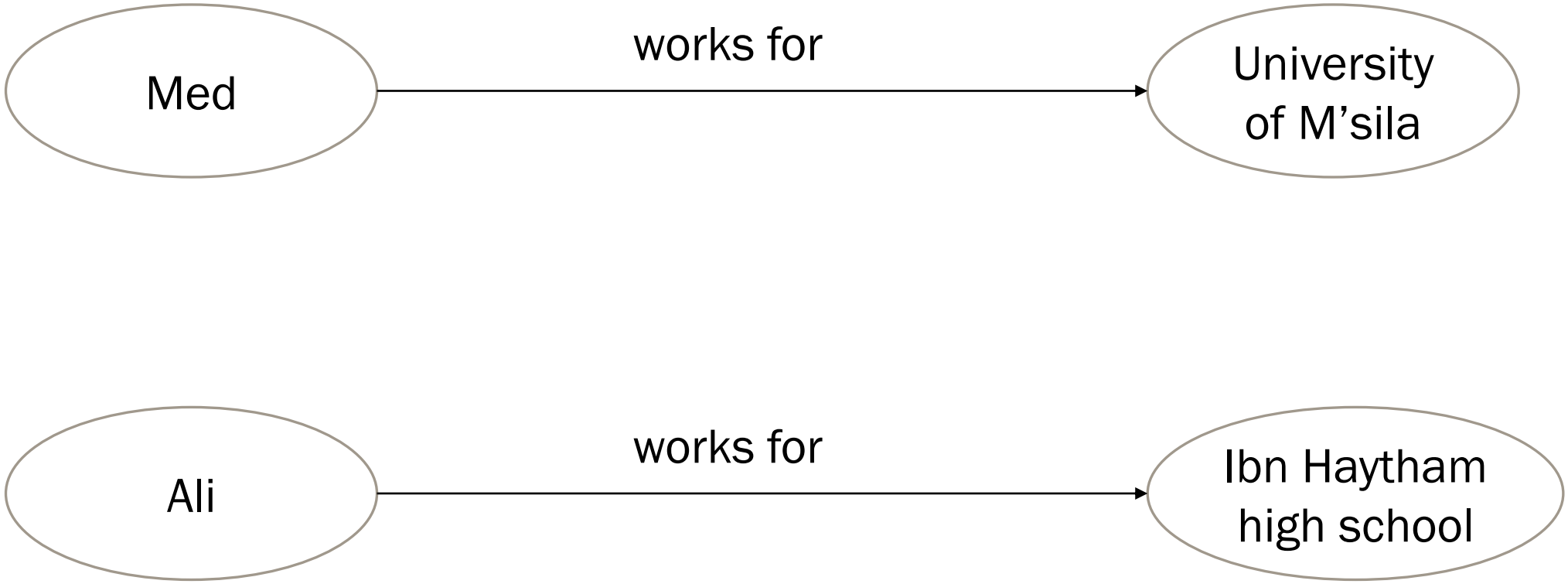


Example in plain English: Med works for the University of M’sila



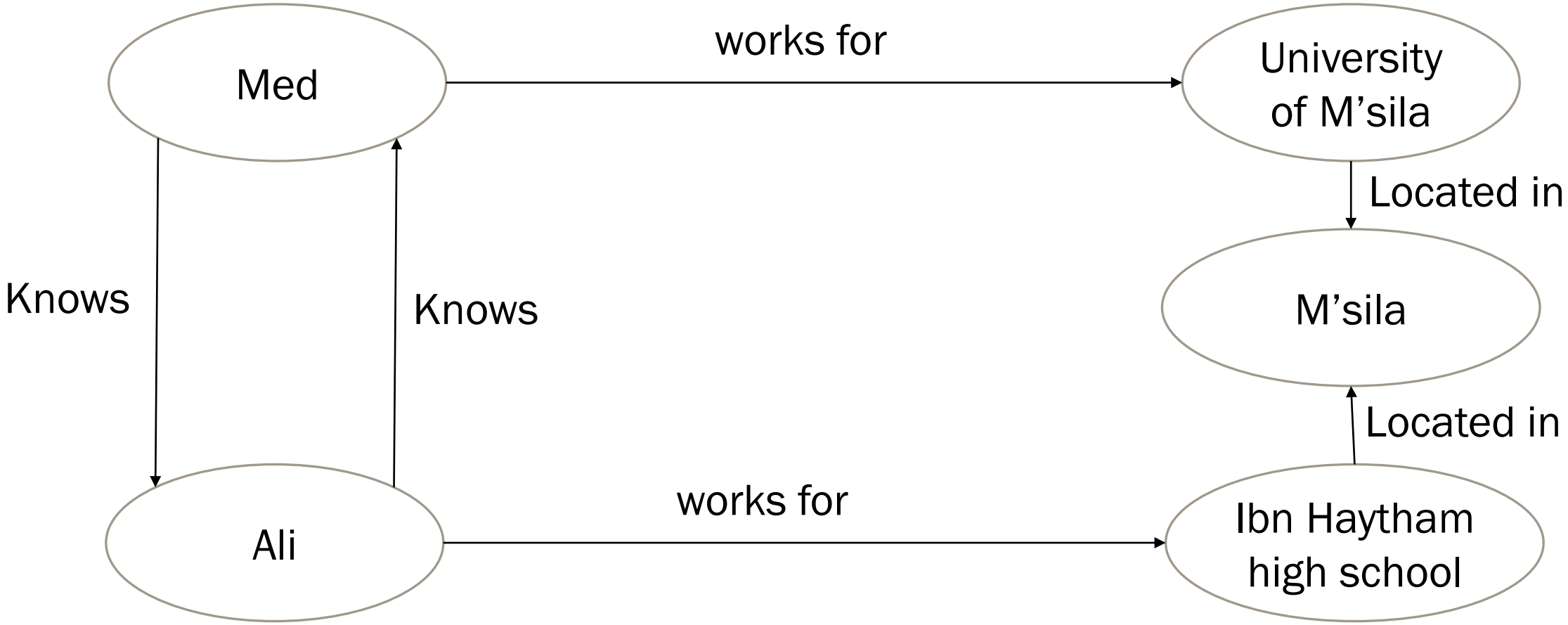
RDF: Resource Description Framework

A knowledge graph consists of multiple sentences



RDF: Resource Description Framework

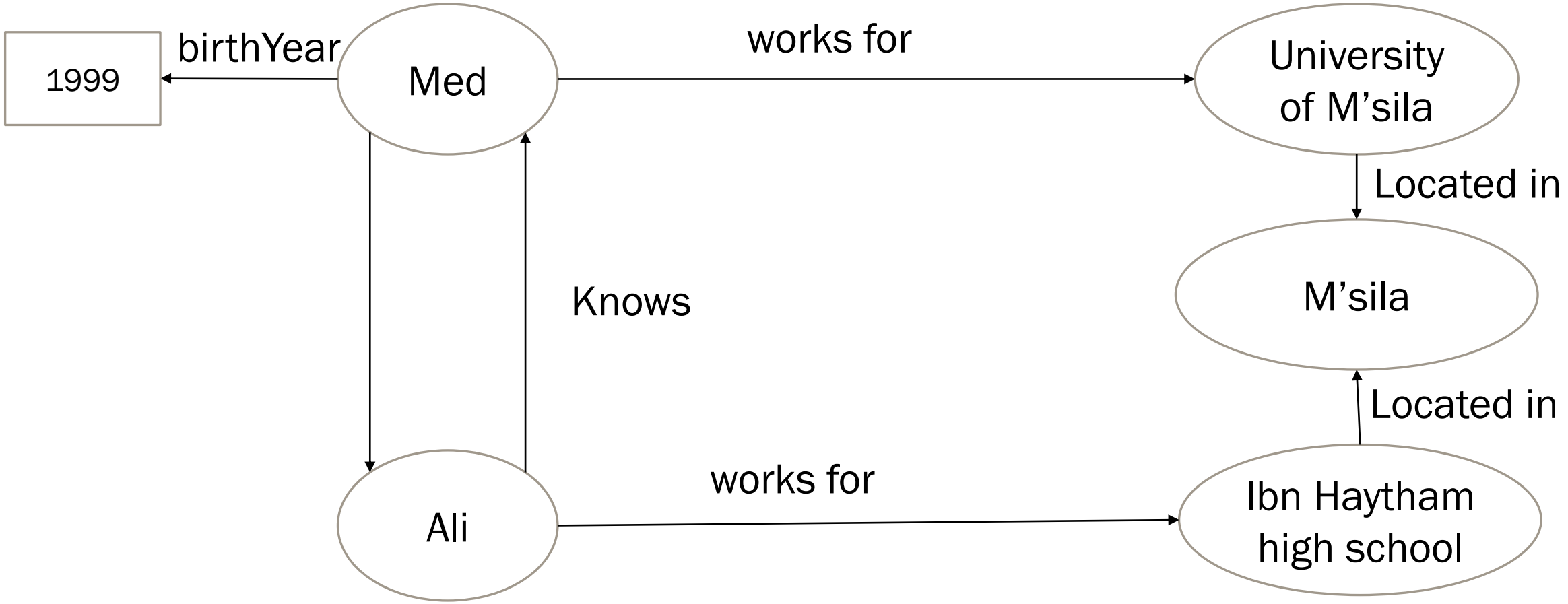
A knowledge graph consists of multiple sentences



– Objects of one statement become subjects of another

RDF: Resource Description Framework

A **literal** is a **self-contained value** that represents **data but not identity**, and can only appear as the **object** in an RDF triple, never as the subject.



– Objects of one statement become subjects of another

Turtle RDF syntax

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

# Person resources
:Med a :Person ;
    :birthYear "1999"^^xsd:integer ;
    :worksFor :UniversityOfMsila .

:All a :Person ;
    :worksFor :IbnHaythamHighSchool .

# Organization resources
:UniversityOfMsila a :Organization ;
                :name "University of M'sila" ;
                :locatedIn "M'sila" .

:IbnHaythamHighSchool a :Organization ;
                    :name "Ibn Haytham high school" ;
                    :locatedIn "M'sila" .
```

There are also other syntax as RDF/XML and Jason

RDFS (RDF Schema)

RDFS (RDF Schema) is an extension of RDF (Resource Description Framework) that provides a basic **vocabulary for defining semantic schemas** for RDF data. It's used to describe the structure, relationships, and constraints of RDF data in a machine-readable way.

RDFS allows us to:

- Define **classes** and **subclasses** of resources.
- Define **properties** and **subproperties**.
- Specify **domains** and **ranges** for properties.
- Enable **basic inference** (e.g., class membership, hierarchy propagation).

RDFS (RDF Schema)

Key RDFS Vocabulary

Classes & Hierarchy

- rdfs:Class → Defines a class of resources.
- rdfs:Resource → The class of all resources.
- rdfs:Literal → The class of literal values (strings, numbers).
- rdfs:subClassOf → Establishes a subclass relationship between classes.

Properties & Typing

- rdfs:Property → The class of RDF properties.
- rdfs:subPropertyOf → Declares a property as a subproperty of another.
- rdfs:domain → Specifies the class of subjects for a property.
- rdfs:range → Specifies the class or datatype of values for a property.

Annotations & Documentation

- rdfs:label → Human-readable label for a resource.
- rdfs:comment → Descriptive text about a resource.
- rdfs:seeAlso → Reference to additional information.

RDFS Example using Turtle syntax

```
# Define classes with proper hierarchy
:Person a rdfs:Class .
:Student rdfs:subClassOf :Person .
:Professor rdfs:subClassOf :Person .

# Define property with clearer semantics
:hasSupervisor a rdfs:Property ;
               rdfs:domain :Student ;
               rdfs:range :Professor .

# Instance examples
:Med a :Student ;
     :hasSupervisor :Aya .
:Aya a :Professor .
```

RDFS Example using RDF/XML syntax

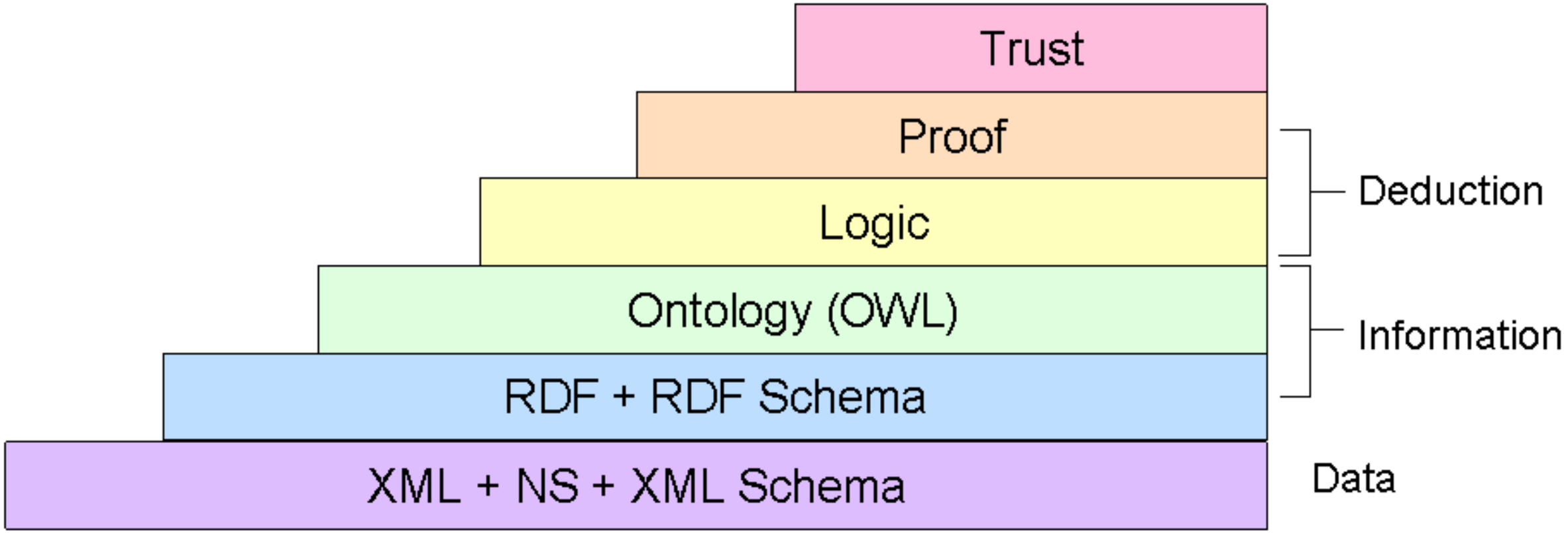
```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ex="http://example.org/">
  <!-- Define classes with proper hierarchy -->
  <rdfs:Class rdf:about="http://example.org/Person">
    <rdfs:label>Person</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:about="http://example.org/Student">
    <rdfs:label>Student</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://example.org/Person"/>
  </rdfs:Class>
.....
</rdf:RDF>
```

RDFS Example using Jason syntax

```
{
  "@context": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "ex": "http://example.org/",
    "Person": "ex:Person",
    "Student": "ex:Student",
    "Professor": "ex:Professor",
    "hasSupervisor": "ex:hasSupervisor"
  },

  "@id": "ex:Person",
  "@type": "rdfs:Class",

  "@graph": [
    {.....
```


Trust
Proof
Logic
Ontology (OWL)
RDF + RDF Schema
XML + NS + XML Schema
Deduction
Information
Data