- 1950 : Test de Turing
- Logique : modélisation formelle de mécanismes de bon sens, de raisonnement
  ↳ décrire des infos + associer un sens + raisonner dessus
  ↳ syntaxe + sémantique (sens) + mécanisme de déduction

Logique de description :

- Niveau terminologique (Tbox) : concepts (voiture, pizza, haine), rôles (manger, aimer)
- Niveau assertionnel (Abox) : instances de concepts (voiture immatriculé X01), instances de relations (Manger (Pomme#3, Eve))
- concepts primitifs
- concepts définis : par CNS d'appartenance, concepts & rôle
- subsomption (hiérarchie des concepts)

Ex :

Personne ≤ Top
Homme ≤ Personne] concepts primitif
Femme = (and Personne (not Homme))] concept défini] Tbox
Personne (Nicomette)
Equipe = Laas
membre (Laas, Nicomette)
membre (Laas, Dragomiresco)] Assertions (Abox)
↳ inférence : Dragomiresco est une personne.

Tim Berner's Lee

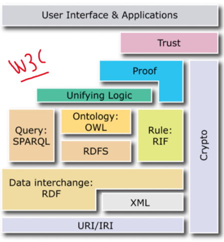
Graphe de connaissances :

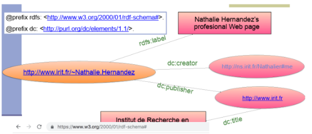
- Descriptions des ressources fondées sur ces ontologies

Ontologies

- Représentations de connaissances
- Favoriser l'interopérabilité.
- Associer une interprétation partagée aux descriptions de ressources

Les URI Uniform Resource Identifier ⤳ IRI : International[...] (en différentes langues)





https://www.w3.org/2000/01/rdf-schema#

```
rdfs:label a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "label" ;
        rdfs:comment "A human-readable name for the subject." ;
        rdfs:domain rdfs:Resource ;
        rdfs:range rdfs:Literal .
```

RDF : Resource Description Framework ⤳ graphes RDF ↗
↳ triplet (sujet, predicat, objet) : (livre#3176, aParAuteur, onfray) + URI
Les ontologies ont des URIs et il est possible d'accéder à leur description RDF.

Vocabulaire RDF

- **Classes**
  - **rdf:Resource** - the class resource, everything
  - **rdf:XMLLiteral** - the class of XML literal values
  - **rdf:Property** - the class of properties
  - **rdf:Statement** - the class of RDF statements
  - **rdf:Alt**, **rdf:Bag**, **rdf:Seq** - containers of alternatives, unordered containers, and ordered containers
  - **rdf:List** - the class of RDF Lists
  - **rdf:nil** - an instance of rdf:List representing the empty list
- **Properties**
  - **rdf:type** - an instance of rdf:Property used to state that a resource is an instance of a class
  - **rdf:first** - the first item in the subject RDF list
  - **rdf:rest** - the rest of the subject RDF list after the first item
  - **rdf:value** - idiomatic property used for structured values
  - **rdf:subject** - the subject of the subject RDF statement
  - **rdf:predicate** - the predicate of the subject RDF statement
  - **rdf:object** - the object of the subject RDF statement

RDFA : RDF sous forme HTML
↘ JSON-LD : RDF sous forme JSON
↘ microdata : similaire à RDFA par les GAFAM

Classes de base

- rdf:Resource, la classe des ressources (tout)
  - exemple : Barack
- rdfs:Class, la classe des classes
  - exemple : les Présidents des USA
- rdf:Property, la classe des propriétés
  - exemple : aGagnéEn
- rdfs:Literal, la classe des littéraux (chaînes, nombres)
  - exemple : 2012, "Yes we can !"
- rdf:Statement, la classe des triplets (Cf. réification)
  - exemple : {Barack, aGagnéEn, 2012}

donnée
langage de description (RDF)

Que mettre dans une ontologie ?

↳ concepts
↳ relations entre concepts dont relations de subsomption
↳ règles et axiomes pour raisonner
↳ symbôle (termes) désignant les concepts/relations

Propriétés de base

- rdfs:subClassOf, permet d'associer une classe à l'une de ses super-classes
- rdfs:subPropertyOf, permet d'associer une propriété à l'une de ses super-propriétés

Propriétés pour définir des restrictions sur les propriétés

- rdfs:domain : domaine d'une Propriété
  - classe de toutes les ressources qui peuvent apparaître comme sujet dans un triplet Sujet-Prédicat-Objet
- rdfs:range
  - classe de toutes les ressources qui peuvent apparaître comme valeur d'une propriété dans un triplet Sujet-Prédicat-Objet

donnée + structure de données (RDFS)

RDFS permet de faire le squelette de l'ontologie, il manque la possibilité de faire des classes et des propriétés.

OWL : RDF + RDFS + propriétés (style inférences)
↳ owl:Thing et owl:Nothing (top et bottom)
↳ owl:Class est une sous classe de rdfs:Class

```
<owl:Class rdf:about="Enseignant"/>
```

↳ subsomption entre class

```
<owl:Class rdf:about="MaitreDeConferences">
  <rdfs:subClassOf rdf:resource="Enseignant"/>
</owl:Class>
```

↳ a : prefix pour connaitre rdf:type

```
PREFIX : <http://www.lamaisondumeurtre.fr#>
SELECT ?personne WHERE {
  ?personne a :Personne ;
            :estVivant "true"^^xsd:boolean .
}
```

```
PREFIX : <https://w3id.org/cluedo4KG/onto#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX house_used: <https://w3id.org/cluedo4KG/KG#lmdm>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

1. les pièces
SELECT * WHERE
{
    ?house_used :houseContainsRoom ?room.
}

2. nombre de pièces
SELECT (COUNT(?room) AS ?room_count) WHERE
{
    ?house_used :houseContainsRoom ?room.
}

3. nb de gens dans la pièces
SELECT (COUNT(?person) AS ?person_count) WHERE
{
    ?house_used :houseContainsRoom ?room.
    ?room :roomContainsPerson ?person.
}

4. nom de la victime
SELECT ?name WHERE
{
    ?house_used :houseContainsRoom ?room.
    ?room :roomContainsPerson ?person.
    ?person :isAlive false.
    ?person rdfs:label ?name.
}

5. nom des suspects
SELECT ?name WHERE
{
    ?house_used :houseContainsRoom ?room.
    ?room :roomContainsPerson ?person.
    ?person :isAlive true.
    ?person rdfs:label ?name.
}
```

```
7. est-ce que la room contient quelqu'un d'autre de suspect (vivant)
ASK  WHERE
{
    ?house_used :houseContainsRoom ?room.
    ?room :roomContainsPerson ?victim.
    ?victim :isAlive false.
    ?suspect :isAlive true.
    ?room :roomContainsPerson ?suspect.
}

8. lister les gens dans chaque pièces
SELECT ?name ?roomname WHERE
{
    ?house_used :houseContainsRoom ?room.
    ?room :roomContainsPerson ?person.
    ?person rdfs:label ?name.
    ?room rdfs:label ?roomname.
}

9. combien de personnes dans chaque room
SELECT ?roomLabel (COUNT(?person) AS ?personCount) WHERE {
  ?house_used :houseContainsRoom ?room.
  ?room :roomContainsPerson ?person.
  ?room rdfs:label ?roomLabel.
  FILTER (LANG(?roomLabel) = "fr")
}
GROUP BY ?roomLabel

10. quelles sont les rooms ou y a au moins 2 personnes
SELECT ?roomLabel (COUNT(?person) AS ?personCount) WHERE {
  ?house_used :houseContainsRoom ?room.
  ?room :roomContainsPerson ?person.
  ?room rdfs:label ?roomLabel.
  FILTER (LANG(?roomLabel) = "fr")
}
GROUP BY ?roomLabel
HAVING (COUNT(?person) >= 2)

11. quelles rooms sont vides ?
SELECT ?roomLabel (COUNT(?person) AS ?personCount) WHERE {
  ?house_used :houseContainsRoom ?room.
  ?room :roomContainsPerson ?person.
  ?room rdfs:label ?roomLabel.
  FILTER (LANG(?roomLabel) = "fr")
}
GROUP BY ?roomLabel
HAVING (COUNT(?person) = 0)
```

```
X. exclusions :
SELECT ?name WHERE {
    .... ?name.
    MINUS {
    ... ?name.
    }
}
```