Semantic Web: Ontologies

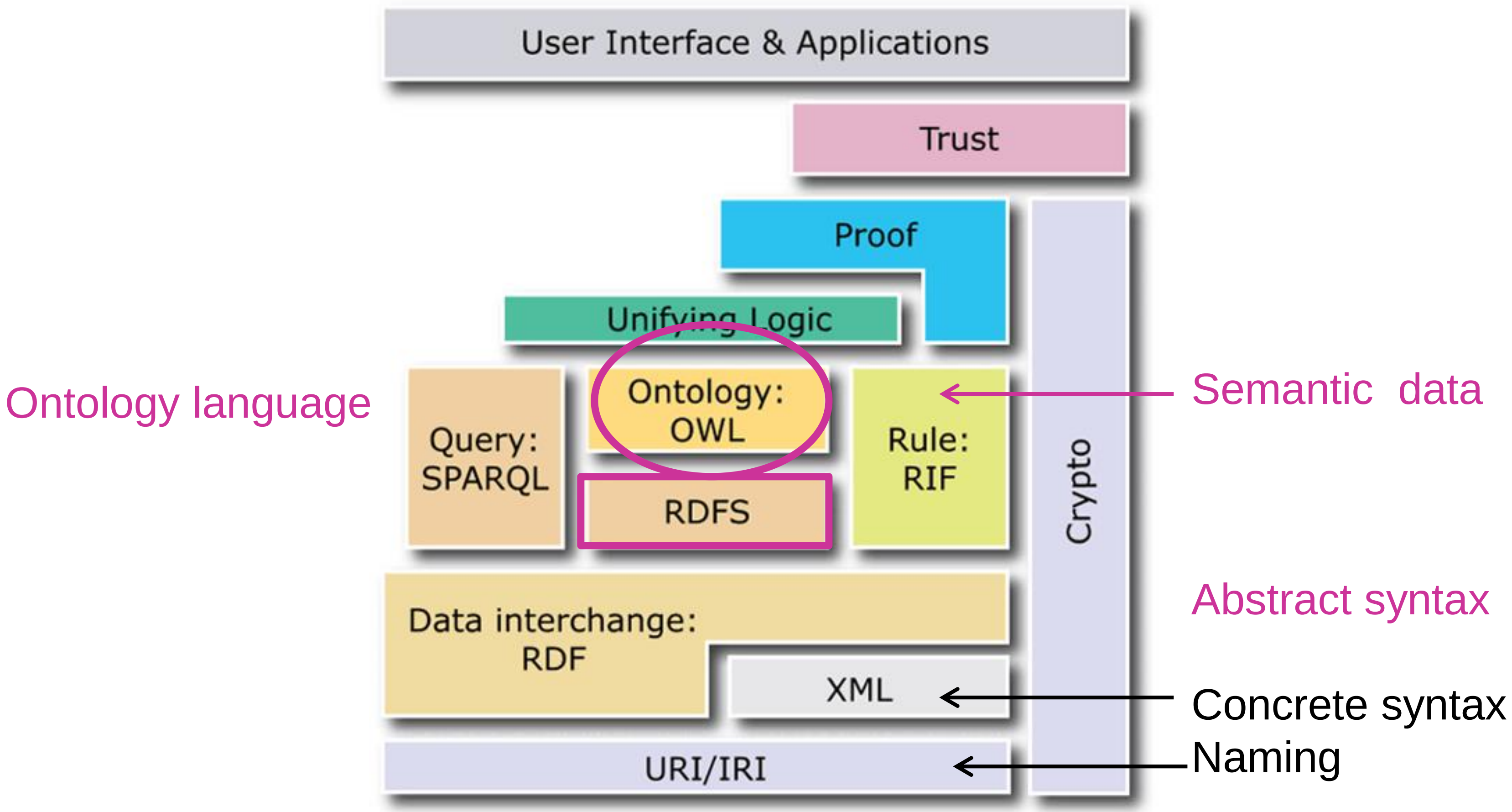
N. Aussenac-Gilles
IRIT- CNRS
aussenac@irit.fr
MELODI group
http://www.irit.fr/-Equipe-MELODI-

Tutorials about OWL and ontologies

- L. Vieu, O. Haemmerlé, Master M2R UPS
- https://www.irit.fr/~Andreas.Herzig/Cours/CDescrLogic/LDescr.pdf
- Mooc Web semantique d'INRIA

  https://www.fun-mooc.fr/courses/inria/41002S02/session02/26a7ae9651d745fc998d2ec72ae37535/

The Semantic Web layer cake (2010)



SPARQL: structure of a query (reminder)

- Namespace declaration
  PREFIX pref: <http://www.exemple.com/ressources#> **...**

- Expected result
  SELECT / ASK/ CONSTRUCT

- Pattern query definition with searched criteria: graph pattern
  WHERE {
  …
  }

- Browsing / filtering results
  FILTER … (in WHERE)
  ORDER BY ... (after WHERE)

Exercice 1

1. ?x dbpedia-owl:child ?y (lire « has child »)

2. ?p foaf:name ''Tim Berners-Lee''@en

3. ?x rdf:type foaf:person

4. _:<http://dbpedia.org/ontology/> rdf:type ?class

- Que recherchent ces triplets ?
- Écrire la requête SPARQL qui cherche les triplets répondant au critère 4 et affiche les classes résultats par ordre alphabétique du label des classes

```
prefix db-owl: http://dbpedia.org/ontology/

SELECT DISTINCT ?class
WHERE { _:db-owl rdf:type ?class .}
ORDER BY ?class
```

```
prefix db-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?class
WHERE { _:db-owl rdf:type ?class .}
ORDER BY ?class
```

Classes triées par leur ID (et non label)

class
http://dbpedia.org/class/Book
http://dbpedia.org/class/yago/!!!Albums
http://dbpedia.org/class/yago/!T.O.O.H.!Albums
http://dbpedia.org/class/yago/%22UnnamedHero%22Novels
http://dbpedia.org/class/yago/%22WeirdAl%22YankovicAlbums
http://dbpedia.org/class/yago/%22WeirdAl%22YankovicCompilationAlbums
http://dbpedia.org/class/yago/%22WeirdAl%22YankovicSongs
http://dbpedia.org/class/yago/%22WeirdAl%22YankovicVideoAlbums
http://dbpedia.org/class/yago/%3F%3F%3FSongs
http://dbpedia.org/class/yago/%C2%A1All-TimeQuarterback!Albums
http://dbpedia.org/class/yago/%C2%A1Forward,Russia!Albums
http://dbpedia.org/class/yago/%C3%81guiaDeMarab%C3%A1Players
http://dbpedia.org/class/yago/%C3%81guilasCibae%C3%B1asPlayers
http://dbpedia.org/class/yago/%C3%81guilasDeMexicaliPlayers
http://dbpedia.org/class/yago/%C3%81lexUbagoAlbums
http://dbpedia.org/class/yago/%C3%81lvaroTorresSongs
http://dbpedia.org/class/yago/%C3%81ngelCustodioLoyolaAlbums
http://dbpedia.org/class/yago/%C3%81rabeUnidoPlayers
http://dbpedia.org/class/yago/%C3%81satr%C3%BATexts
http://dbpedia.org/class/yago/%C3%81smeginAlbums
http://dbpedia.org/class/yago/%C3%84ngelholmsFFPlayers

dbpedia.org/class/yago/!!!Albums

Exercice 1

1. ?x dbpedia-owl:child ?y (lire « has child »)

2. ?p foaf:name ‘’Tim Berners-Lee’’@en

3. ?x rdf:type foaf:person

4. _:<http://dbpedia.org/ontology/> rdf:type ?class

- Que recherchent ces triplets ?
- Écrire la requête SPARQL qui cherche les triplets répondant au critère 4 et affiche les classes résultats par ordre alphabétique du label des classes

```
prefix db-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT  ?label ?class
WHERE { _:db-owl rdf:type ?class .
                ?class rdfs:label ?label .}
ORDER BY ?label
```

```
prefix db-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT  ?label ?class
WHERE { _:db-owl rdf:type ?class .
            ?class rdfs:label ?label .}
ORDER BY ?label
```

label	class
"AnnotationProperty"	http://www.w3.org/2002/07/owl#AnnotationProperty
"Class"	http://www.w3.org/2002/07/owl#Class
"Ontology"	http://www.w3.org/2002/07/owl#Ontology
"OntologyProperty"	http://www.w3.org/2002/07/owl#OntologyProperty
"Thing"	http://www.w3.org/2002/07/owl#Thing

```
prefix db-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT  ?label ?class
WHERE { _:db-owl rdf:type ?class .
        OPTIONAL  { ?class rdfs:label ?label .}
}
ORDER BY ?label
LIMIT 1000
```

label	class
"AnnotationProperty"	http://www.w3.org/2002/07/owl#AnnotationProperty
"Class"	http://www.w3.org/2002/07/owl#Class
"Ontology"	http://www.w3.org/2002/07/owl#Ontology
"OntologyProperty"	http://www.w3.org/2002/07/owl#OntologyProperty
"Thing"	http://www.w3.org/2002/07/owl#Thing
	http://www.openlinksw.com/schemas/virtrdf#QuadMapFormat
	http://www.openlinksw.com/schemas/virtrdf#QuadStorage
	http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMap
	http://www.openlinksw.com/schemas/virtrdf#QuadMap
	http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMapFormat
	http://www.openlinksw.com/schemas/virtrdf#QuadMapValue
	http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMapATable
	http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMapColumn
	http://www.openlinksw.com/schemas/virtrdf#QuadMapColumn
	http://www.openlinksw.com/schemas/virtrdf#QuadMapFText
	http://www.openlinksw.com/schemas/virtrdf#QuadMapATable
	http://www.openlinksw.com/schemas/virtrdf#array-of-string
	http://www.w3.org/1999/02/22-rdf-syntax-ns#Property
	http://www.w3.org/2000/01/rdf-schema#Class

ss+%0D%0AWHERE+%
AL+%7B%3Fclass+rdfs
el%0D%0ALIMIT+1000%

Exercice 2

- Retrouver dans dbpedia les 1000 premières instances de foaf:Person dont le nom (foaf:name) contient “Tim”
- regex(?l,"Tim")

```
SELECT DISTINCT ?p ?l
WHERE {?p a foaf:Person .
        ?p foaf:name ?l .
FILTER regex(?l,"Tim") .
}
LIMIT 1000
```

p	l
http://dbpedia.org/resource/Georgy_Dobrovolsky	"Georgiy Timofeyevich Dobrovolsky"@en
http://dbpedia.org/resource/Joachim_Christian_Timm	"Joachim Christian Timm"@en
http://dbpedia.org/resource/Kimmo_Timonen	"Kimmo Timonen"@en
http://dbpedia.org/resource/Tim_Allen	"Tim Allen"@en
http://dbpedia.org/resource/Tim_Berne	"Tim Berne"@en
http://dbpedia.org/resource/Tim_Bevan	"Tim Bevan"@en
http://dbpedia.org/resource/Tim_Bogert	"Tim Bogert"@en
http://dbpedia.org/resource/Tim_Booth	"Tim Booth"@en
http://dbpedia.org/resource/Tim_Brent	"Tim Brent"@en
http://dbpedia.org/resource/Tim_Curry	"Tim Curry"@en
http://dbpedia.org/resource/Tim_DeKay	"Tim DeKay"@en
http://dbpedia.org/resource/Tim_DeKay	"Tim Dekay"@en
http://dbpedia.org/resource/Tim_Haines	"Tim Haines"@en

Exercice 2bis

- Retrouver les instances dbpedia dont le nom contient “Tim” classées par ordre alphabétique

```
SELECT DISTINCT ?p ?l WHERE {?p a foaf:Person .
?p foaf:name ?l .
FILTER regex(?l,"Tim") .
}
ORDER BY ?l
LIMIT 1000
```

p	l
http://dbpedia.org/resource/A.J._Timothy_Jull	"A. J. Timothy Jull"@en
http://dbpedia.org/resource/Alec_Boswell_Timms	"A.B. Timms"@en
http://dbpedia.org/resource/Abdillahi_Suldaan_Mohammed_Timacade	"Abdillahi Suldaan Mohammed 'Timacade'"@en
http://dbpedia.org/resource/Abdillahi_Suldaan_Mohammed_Timacade	"Abdillahi Suldaan Mohammed Timacade"@en
http://dbpedia.org/resource/Abu_Sa'id_Mirza	"Abū Saʿīd Mirza b. Muḥammad b. Mīrānshāh b. Timūr"@en
http://dbpedia.org/resource/Adam_Timmerman	"Adam Timmerman"@en
http://dbpedia.org/resource/Addison_Timlin	"Addison Timlin"@en
http://dbpedia.org/resource/Adrian_Mannix	"Adrian Timothy Mannix"@en
http://dbpedia.org/resource/Al_%22Carnival_Time%22_Johnson	"Al "Carnival Time" Johnson"@en
http://dbpedia.org/resource/Al_%22Carnival_Time%22_Johnson	"Al Carnival Time Johnson"@en
http://dbpedia.org/resource/Al_Timothy	"Al Timothy"@en
http://dbpedia.org/resource/Albert_Timmer	"Albert Timmer"@en
http://dbpedia.org/resource/Al_Timothy	"Albon "Al" Timothy"@en
http://dbpedia.org/resource/Alec_Boswell_Timms	"Alec Boswell Timms"@en
http://dbpedia.org/resource/Aleksandr_Timofeyev	"Aleksandr Dmitriyevich Timofeyev"@en
http://dbpedia.org/resource/Aleksandr_Timofeyev	"Aleksandr Timofeyev"@en
http://dbpedia.org/resource/Aleksandr_Prokopenko	"Aleksandr Timofeyevich Prokopenko"@en
http://dbpedia.org/resource/Aleksandr_Timoshinin	"Aleksandr Timoshinin"@en
[illegible]bers	"Alex Timbers"@en

dbpedia.org/resource/Aleksandr_Prokopenko

Exercice 3: Afficher les parents de Tim Berners-Lee et leur nom

?parent dbpedia-owl:child ?person

http://dbpedia.org/sparql/

```
select distinct ?tb ?p ?namep
where {
?p      a foaf:Person ;
        foaf:name ?namep ;
        dbpedia-owl:child ?tb .
?tb     a foaf:Person ;
        foaf:name "Tim Berners-
Lee"@en .
}
```

tb	p	namep
http://dbpedia.org/resource/Tim_Berners-Lee	http://dbpedia.org/resource/Mary_Lee_Woods	"Mary Lee Woods"@en
http://dbpedia.org/resource/Tim_Berners-Lee	http://dbpedia.org/resource/Conway_Berners-Lee	"Conway Berners-Lee"@en

Exercise 4: Interrogation du site http://fr.dbpedia.org/sparql

```
prefix db-owl: <http://dbpedia.org/ontology/>
 select distinct ?ville ?l ?population
where {
      ?ville db-owl:region <http://fr.db
      ?ville rdf:type db-owl:Settlemen
      Optional {?ville rdfs:label ?l }
     ?ville db-owl:populationTotal ?pc
      filter (?population > 10000)
}
```

ville	l	population
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@fr	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@en	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmauç"@ca	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@de	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@es	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@it	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@pt	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmaux"@eu	10116
http://fr.dbpedia.org/resource/Carmaux	"Carmauç"@oc	10116
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"Lavaur (Tarn)"@fr	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"Lavaur, Tarn"@en	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"La Vaur"@ca	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"Lavaur (Tarn)"@de	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"Lavaur (Tarn)"@es	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"Lavaur (Tarn)"@it	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"Lavaur (Tarn)"@eu	10148
http://fr.dbpedia.org/resource/Lavaur_(Tarn)	"La Vaur (Lengadòc)"@oc	10148
http://fr.dbpedia.org/resource/Saint-Jean_(Haute-Garonne)	"Saint-Jean (Haute-Garonne)"@fr	10259
http://fr.dbpedia.org/resource/Saint-Jean_(Haute-Garonne)	"Saint-Jean, Haute-Garonne"@en	10259
http://fr.dbpedia.org/resource/Saint-Jean_(Haute-Garonne)	"Sent Joan le Vièlh"@ca	10259
http://fr.dbpedia.org/resource/Saint-Jean_(Haute-Garonne)	"Saint-Jean (Alto Garona)"@es	10259

- Que fait cette requête ?
- La modifier pour ne présenter que les 100 premiers résultats.

- Cours VI– Ontologies definitions

From vocabularies to ontologies

- Target: system interoperability
  - Ex: flight search web portals

  Various data bases, various vocabularies -> a single user interface

- Step1: unify a vocabulary (human agreement)
  - Flight, airport, airline, departure and arrival points, date and time, seats, booking, passenger, fares …

- Step2: define a FORMAL vocabulary
  - Formal agreement, define TYPES, take advantage of INFERENCES to deduce implicit knowledge

What sort of content should machines share?

  - A vocabulary
  - Facts / data expressed with this vocabulary
- Not enough!!
  - Something is missing to make sure the machines all give the same meaning to the shared vocabulary, i.e., to constrain its interpretation
- Knowledge about the domain described through the vocabulary
- Ontologies expressed in a logical framework

Ontologies: some definitions

- An ontology is a “formal, explicit specification of a shared conceptualisation” [Gruber 1993, Borst 1997, Studer et al. 1998]
- An ontology is a specific artefact:
  - a logical theory and/or a computational object that expresses the intendedmeaning of a vocabulary, by referring to the nature and the structure of the entities it denotes
- cf. Ontology, the philosophical discipline that studies “what there is”, i.e., the nature and structure of reality

- Ontologies model conceptual knowledge crucial for
  - Semantic interoperability of systems (not on the internet)
  - Precise human communication within a scientific or technical domain
  - Intelligent information extraction, question-answering
  - Natural language understanding
  - Making the Semantic Web real

What is an ontology in practice?

- A theoretical or computational artefact that

1. *models* knowledge of a domain through a vocabulary of concepts:
   - classes to categorize existing individuals in this domain (entities, \things“ that populate the domain)
   - relations establishing links between those individuals.

2. *formalizes* generic, necessary knowledge of this domain and constrains the interpretation of the vocabulary through axioms

What is an ontology in practice?
A knowledge model

- Classes: what types of individuals exist (in general / in my domain of interest)? what distinctions are significant between them?
  - e.g., tables, chairs, students, teachers, humans, courses, disciplines, universities, classrooms, computers...

- Classes correspond to *unary predicates* in FOL (First Order Logic)
  - Human(Lea) encodes the fact that the individual Lea is an instance of the class Human

:Lea rdf:type :Human in RDF

What is an ontology in practice? A knowledge model

- Relations: in which ways may those individuals be related?
  - e.g., humans may sit on chairs, students may be enrolled in universities,
  - a course may be taught by a teacher...
- Relations correspond to *n-ary predicates* (n > 1) in FOL
  - **enrolledAt**(Lea,UPS) encodes the fact that the individuals Lea and UPS are related by the relation **enrolledAt**

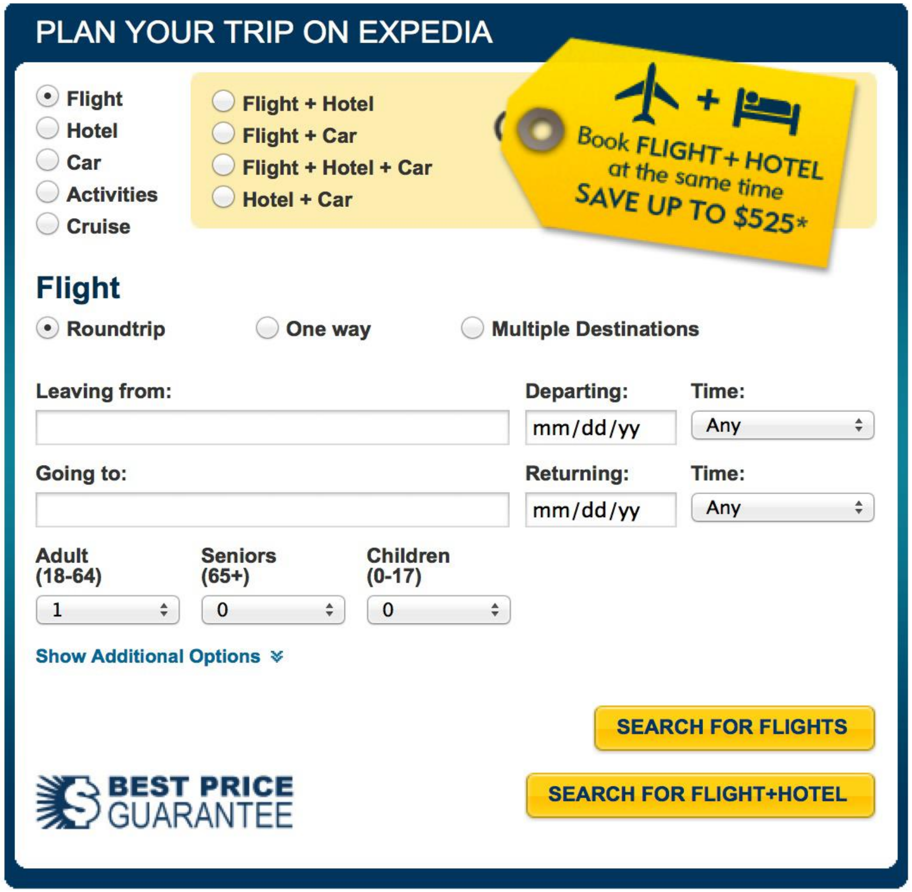
What is an ontology in practice? A formal theory

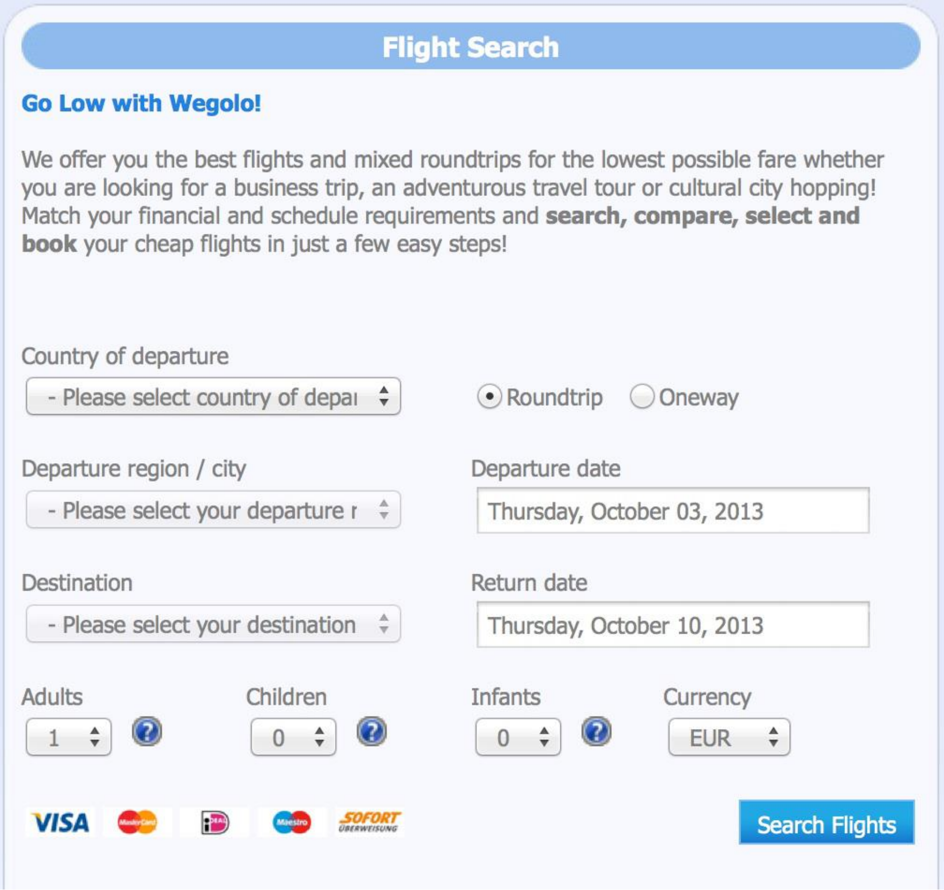
- Taxonomic links between classes (IS-A, subsumption)
  e.g., a student is a human:
  Student IS-A **Human**
  $\forall x\ (\mathbf{Student}(x) \rightarrow \mathbf{Human}(x))$

- Characterization of the relation arguments, esp. the domain and range of binary relations
  e.g., only students and universities can be related through enrollment:
  $\forall\ xy\ (\mathbf{enrolledAt}(x; y) \rightarrow \mathbf{Student}(x) \wedge \mathbf{University}(y))$
- More complex constraints on classes and relations

- Specific contingent facts are NOT part of the ontology
  e.g., $\mathbf{Human}(\mathcal{Lea})$, $\mathbf{enrolledAt}(\mathcal{Lea}, \mathcal{UPS})$
  They form a *knowledge base (populated ontology)*

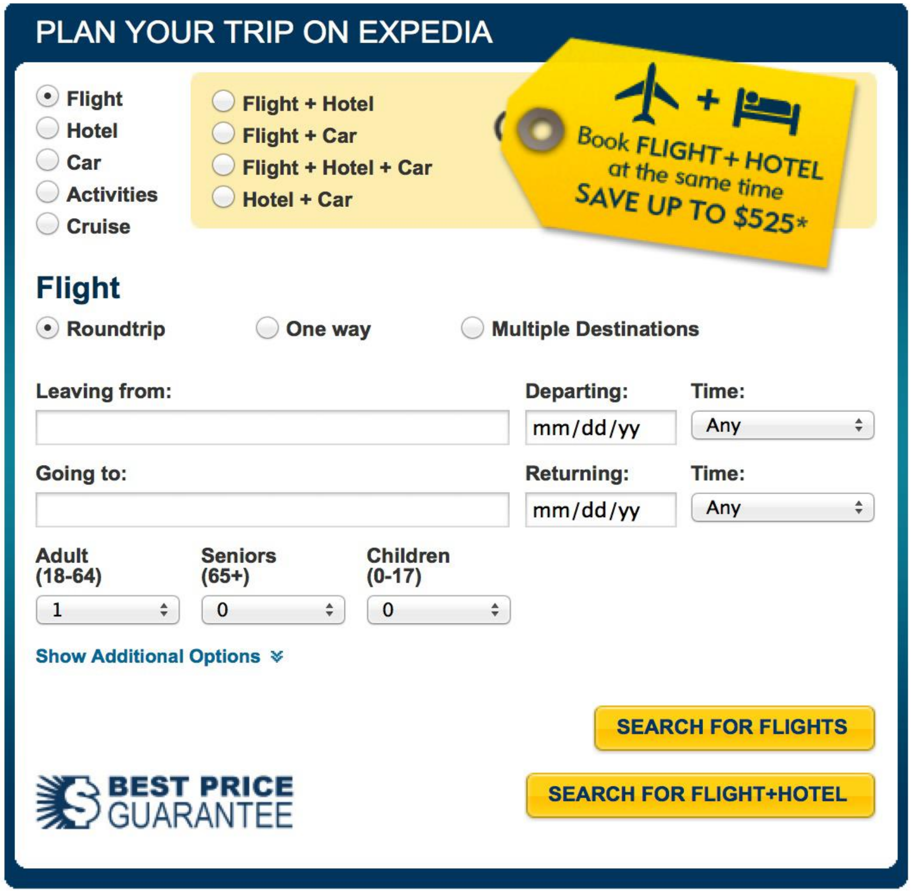
A basic ontology: flight search engines





- Which classes can we identify?
- Which classes will we need?

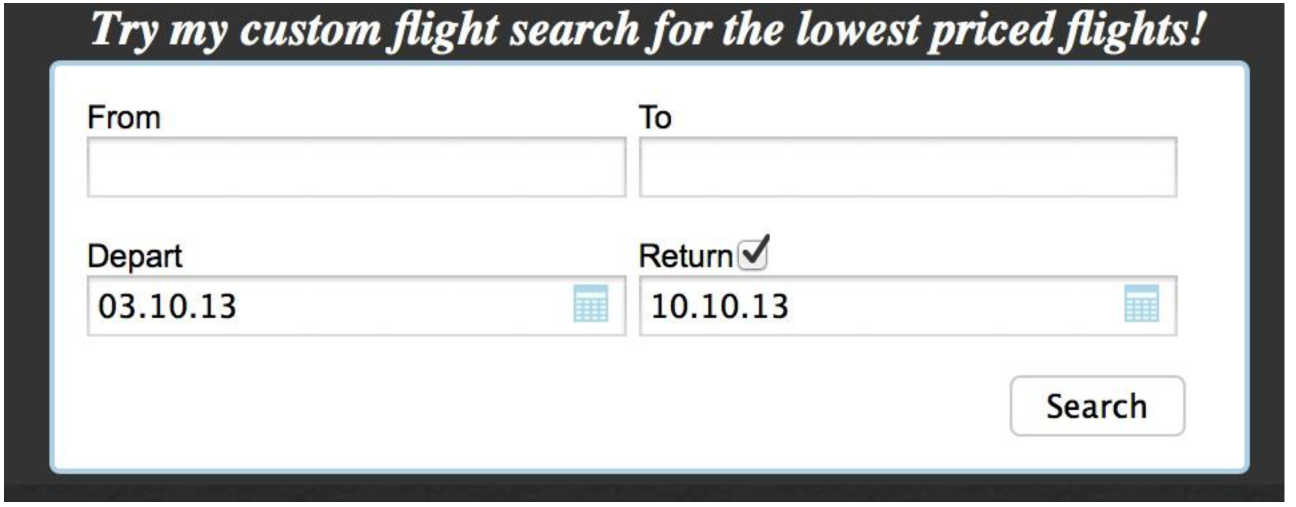
A basic ontology: competency questions



- Is there a direct flight from Toulouse to Heraklion leaving on May,23rd 2014?
- How much is a roundtrip flight from Toulouse to Heraklion around May 23rd, 2014?
- Are there 3 roundtrip seats available on flights to Heraklion from Toulouse on May23rd with a back flight on May 30th, 2014?

Flight, RoundTripF, OneWayF, MultipleDestF, Airport, Town, Date, Time, Human, Adult, Senior, Child, Infant, Currency
but also : Booking, Seat, Fare, Airline, FlightID, directF, stop…

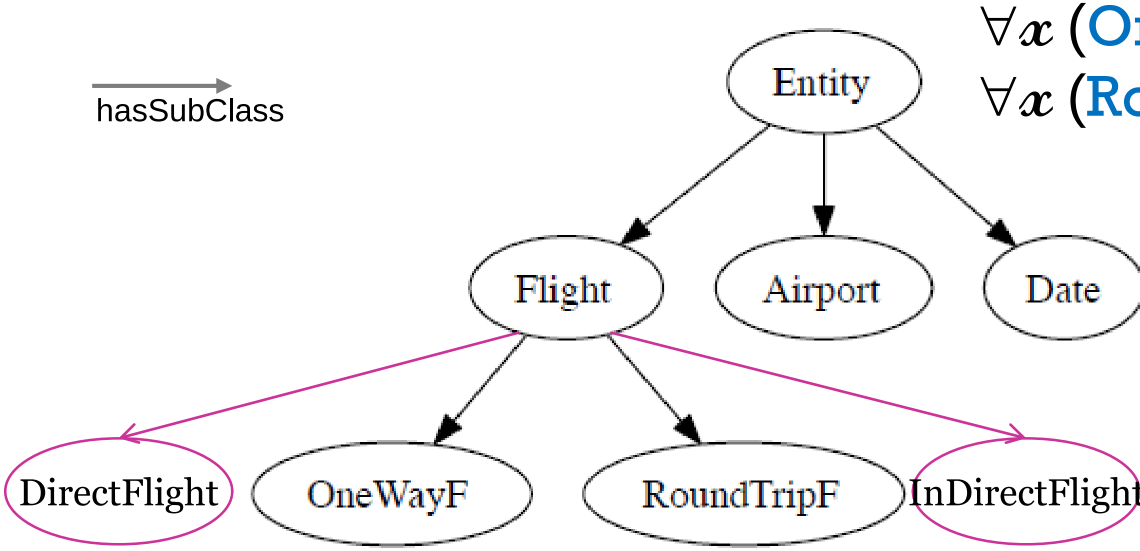
A basic ontology little by little



- Classes? Relations? Domains and ranges?
- Classes: Flight, RoundTripF, OneWayF, Airport, Date
- Relations, all binary:
  - departsFrom(Flight, Airport)
  - goesTo(Flight, Airport)
  - departsOn(Flight, Date)
  - returnsOn(Flight, Date)

A basic ontology: taxonomy of classes

- Organize classes into a subsomption hierarchy
- Classes1: Flight, RoundTripF, OneWayF, Airport, Date
- Classes2: Flight, RoundTripF, OneWayF, DirectFlight, IndirectFlight, Airport, Date



Subsomption axioms

$\forall x\ (\text{OneWayF}(x) \rightarrow \text{Flight}(x))$

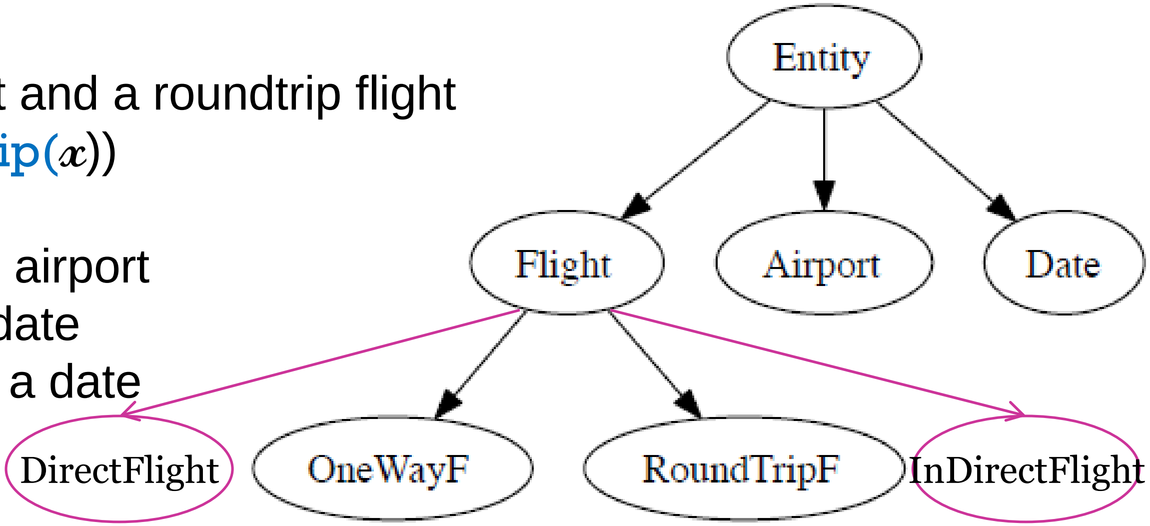
$\forall x\ (\text{RoundTripF}(x) \rightarrow \text{Flight}(x))$

A basic ontology: taxonomy of classes

- Often (but not always!) we can complement subsumption axioms with disjunction axioms between siblings. Here we have:

Nothing is both a oneway flight and a roundtrip flight
$\forall x\ (\mathrm{OneWayF}(x) \rightarrow \neg \mathrm{RoundTrip}(x))$

Nothing is both a flight and an airport
Nothing is both a flight and a date
Nothing is both an airport and a date

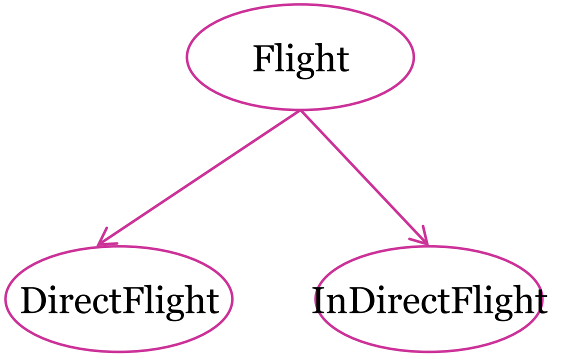


BUT a oneway flight can be either a direct flight or an indirect flight

$\exists x\ (\mathrm{OneWayF}(x) \wedge \mathrm{DirectFlight}(x))$
$\exists x\ (\mathrm{RoundTripF}(x) \wedge \mathrm{DirectFlight}(x))$ …

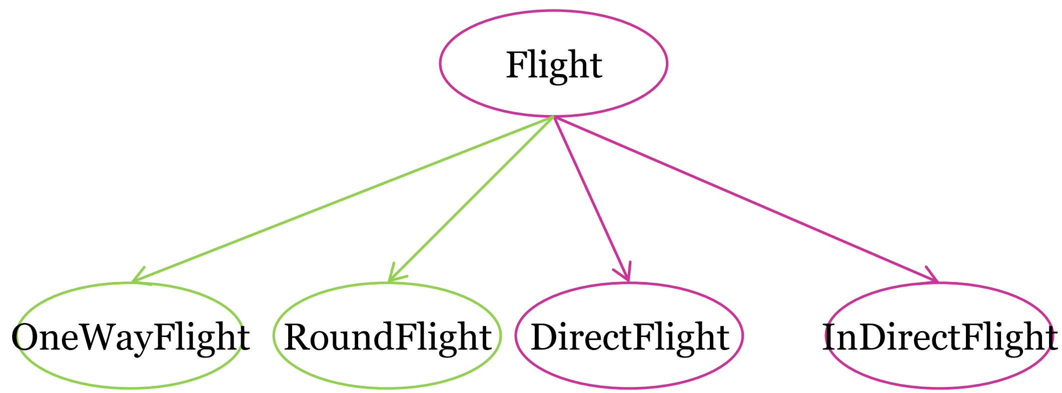
Ontology design: differenciation criteria

- Aristotle definition:
  - Concept = Definendum + differenciae
  - A cat is a domesticated feline of small size
- Archonte method (Bachimont, 2003) : any new concept will be inserted in an ontology if we can find at least one of each features
  - Common feature with father concept
  - Common feature with siblings
  - Different feature from father
  - Different feature from siblings



Ontology design: differenciation criteria

- Pb = mixing various criteria, building an heterogeneous partition
- An instance can belong to two sibling classes
- … but it is not a “clear” representation



A basic ontology: domains and ranges of relations

- departsFrom and goesTo relate instances of Flight to instances of Airport
  $\forall xy\ (\text{departsFrom}(x, y) \rightarrow \text{Flight}(x) \wedge \text{Airport}(y))$
  $\forall xy\ (\text{goesTo}(x, y) \rightarrow \text{Flight}(x) \wedge \text{Airport}(y))$

- departsOn relates instances of Flight to instances of Date
  $\forall xy\ (\text{departsOn}(x, y) \rightarrow \text{Flight}(x) \wedge \text{Date}(y))$

- returnsOn relates instances of RoundTripF to instances of Date
  $\forall xy\ (\text{returnsOn}(x, y) \rightarrow \text{RoundTripF}(x) \wedge \text{Date}(y))$

- RDFs representation of these relations?
- What can be inferred from these relations and the taxonomy of classes?

A basic ontology: domains and ranges of relations

- departsFrom and goesTo relate instances of Flight to instances of Airport

  $\forall xy\ (\text{departsFrom}(x, y) \rightarrow \text{Flight}(x) \wedge \text{Airport}(y))$

  $\forall xy\ (\text{goesTo}(x, y) \rightarrow \text{Flight}(x) \wedge \text{Airport}(y))$

```
<rdfs:Class rdf:about=“#Flight">
    <rdfs:subClassOf rdf:resource=“#Thing "/>
</rdfs:Class>
<rdfs:Class rdf:about=“#OneWayFlight">
    <rdfs:subClassOf rdf:resource=“#Flight "/>
</rdfs:Class>
<rdf:Property rdf:about=“# departsFrom">
    <rdfs:domain rdf:resource=“#Flight"/>
    <rdfs:range rdf:resource=“#Airport"/>
</rdf:Property>
```

```
<rdf:Property rdfabiout=“#goesTo">
    <rdfs:domain rdf:resource=“#Flight"/>
    <rdfs:range rdf:resource=“#Airport"/>
</rdf:Property>
```

Existence and unicity

- We want to make sure that any flight departs from and goes to some airport, at a given date, etc. We here need to add existence constraints to the domain and range constraints.

$\forall x\ (\mathrm{Flight}(x) \rightarrow \exists\ yzt(\mathrm{departsFrom}(x; y) \wedge \mathrm{goesTo}(x; z) \wedge \mathrm{departsOn}(x; t)))$

$\forall x\ (\mathrm{RoundTripF}(x) \rightarrow \exists\ t\ \mathrm{returnsOn}(x; t))$

  - What about the reverse? Does an airport imply the existence of a flight?
  - Is it necessary to specify the types of $y\ z\ t$?

- In some cases, we have in addition unicity constraints (the binary relation is functional). Here, all the relations are functional, i.e., any flight departs from and goes to a unique airport, on a unique date, etc.

$\forall xyz\ ((\mathrm{departsFrom}(x; y) \wedge \mathrm{departsFrom}\ (x; z)) \rightarrow y = z)$

$\forall xyz((\mathrm{goesTo}(x; y) \wedge \mathrm{goesTo}(x; z)) \rightarrow y = z)$

$\forall xyz((\mathrm{departsOn}(x; y) \wedge \mathrm{departsOn}(x; z)) \rightarrow y = z)$

$\forall xyz((\mathrm{returnsOn}(x; y) \wedge \mathrm{returnsOn}(x; z)) \rightarrow y = z)$

What exactly is a roundtrip flight?

- A roundtrip flight is composed of two oneway flights with matching airports
  - New ternary relation: isComposedOf

- Contraint on classes of arguments

$\forall xy\, z\, (\text{isComposedOf}\,(x; y; z) \rightarrow \text{RoundTripF}\,(x) \wedge \text{OneWayF}(y) \wedge \text{OneWayF}(z))$

- Constraint on airports

$\forall\, xyzab(\text{isComposedOf}\,(x; y; z) \wedge \text{departsFrom}(x; a) \wedge \text{goesTo}(x; b) \rightarrow$
$\text{departsFrom}\,(y; a) \wedge \text{goesTo}\,(y; b) \wedge \text{departsFrom}\,(z; b) \wedge \text{goesTo}\,(z; a))$

- Constraint on dates

$\forall xy\, zab\, (\text{isComposedOf}\,(x; y; z) \wedge \text{departsOn}\,(x; a) \wedge \text{returnsOn}\,(x; b) \rightarrow$
$\text{departsOn}\,(y; a) \wedge \text{departsOn}\,(z; b))$

- What constraints are still missing here?
- What if we wanted to use a more general relation?

W3C Standard languages for ontologies

- From RDFs to OWL

Web Ontology Language OWL

- Why is OWL needed?
  - More primitives
  - more complex ontologies
  - Richer concept and property representations
  - More inferences
- Name Space
  - http://www.w3.org/2002/07/owl#
  - primitives OWL are defined here
  - Same principle as for RDFS
  - Prefix owl:

OWL Web Ontology Language

- Historic Ontology languages
  - DAML : standard DARPA (Defense Advanced Research Project Agency) - DARPA Agent Markup Language
  - OIL : Ontology Inference Layer (European project)
  - DAML + OIL
- OWL : Web Ontology Language
  - W3C standard
  - AI inspired Knowledge Representation Language
  - Inference mechanism
  - Formal validation of properties : cardinality, transitivity, …

OWL: inspiration

- Description logics → reasoning
  - Concepts: set of entities
  - Roles: sets of relations between entities (sets of triples)
  - Terminology box or T(Box) → classes →ontology
  - Assertion box A(Box) → instances → knowledge base
- Frames → compact representation of classes
- XML → web navigation

OWL: heritage

http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

mars 1999 **RDFS**

octobre 2000 **DAML**

OIL novembre 2000

mars 2001 **DAML + OIL**

février 2004 **OWL**
OWL lite, OWL DL, OWL Full

octobre 2009 **OWL 2**
OWL 2 – EL, OWL 2 – QL, OWL 2 - RL

Description Logics and ontologies

- Cf roundTrip : no ternary relation in OWL !

$\forall f\ f1\ f2$ (isComposedOf $(f; f1; f2) \rightarrow$ RoundTripF (f) ^ OneWayF$(f1)$ ^ OneWayF$(f2)$)

- Definition using binary relations

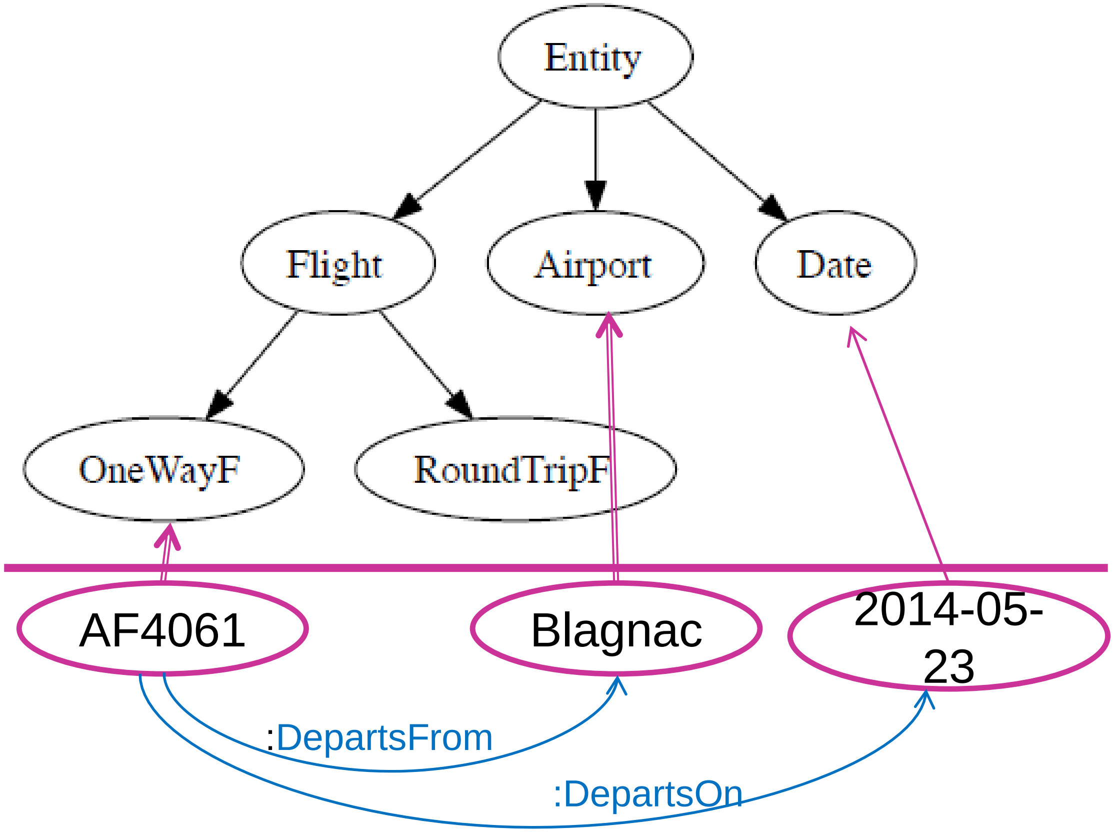
$\forall f$ (RoundTripF $(f) \rightarrow \exists$ *date1*, *date2*, *f1*, *f2*, *city1*, *city2* OneWayF$(f1)$ ^ OneWayF$(f2)$ ^ departsFrom $(f1; city1)$ ^ goesTo $(f1; city2)$ ^ departsFrom $(f2; city2)$ ^ goesTo $(f2; city1)$ ^ departsOn $(f1; date1)$ ^ departsOn $(f2; date2)$ ^ before$(f1; f2)$)

Description Logics and ontologies

- T(Box)
  - Primitive concepts
  - Definite concepts

  = formulas

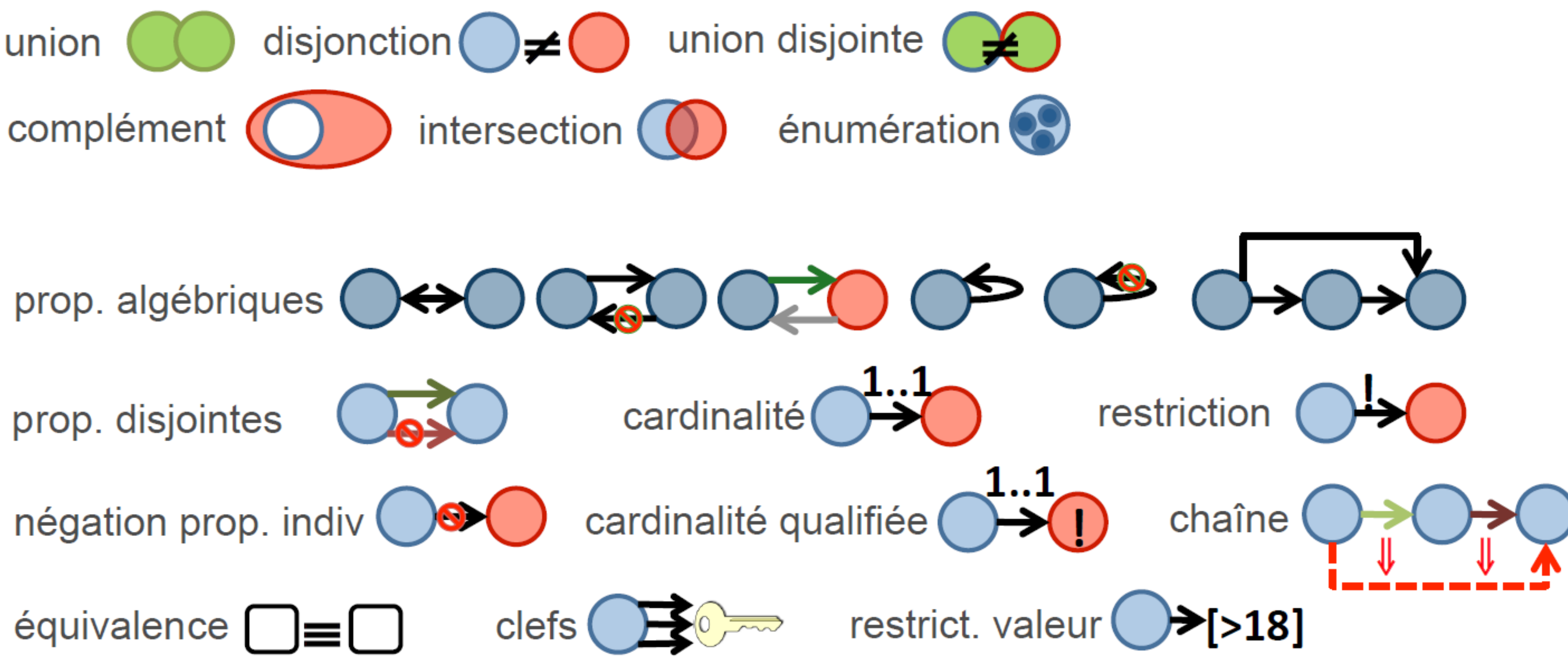
- A(Box)



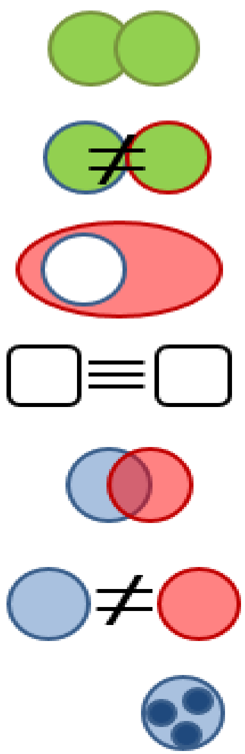
OWL in one…

https://www.fun-mooc.fr/courses/inria/41002S02/session02/about

une vue graphique des constructeurs logiques offerts



Class description primitives



owl:unionOf

owl:disjointUnionOf

owl:complementOf

owl:equivalentClass

owl:intersectionOf

owl:disjointWith

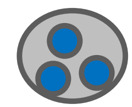
owl:oneOf

Class description (1)

- The class of all OWL classes owl:Class
  owl:Class rdfs:subClassOf rdfs:Class

- Named class
  <owl:Class rdf:ID="Human"/>

- Enumeration of individuals: extended definition



```
<owl:Class rdf:ID="Continent" > <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Africa"/>
        <owl:Thing rdf:about="#America"/>
        <owl:Thing rdf:about="#Asia"/>
        <owl:Thing rdf:about="#Australia"/>
        <owl:Thing rdf:about="#Antarctica"/>
        <owl:Thing rdf:about="#Europe"/>
    </owl:oneOf>
</owl:Class>
```

$$\forall x\ Continent(x) \rightarrow (x = Africa) \wedge (x = America) \wedge (x = Asia) \wedge (x = Australia) \wedge (x = Antartica) \wedge (x = Europe)$$

```
<Continent> a owl:Class ;
    owl:oneOf
    ( <Europe> <Africa> <America> <Asia>
    <Australia> <Antartica>) .
```

1. Turtle writing
2. Formal semantics

Class description (2)

- rdfs:subclassOf

```
<owl:Class rdf:ID="Opera">
      <rdfs:subClassOf rdf:resource="#MusicalWork"/>
</owl:Class>
```

- Disjunction of classes

:Flight owl:disjointWith :Plane


```
<owl:Class rdf:ID="Flight">
   <owl:disjointWith rdf:resource="#Plane"/>
</owl:Class>

<Flight>   a owl:Class ;
           owl:disjointWith <Plane> .
```

Class description (3)

- Class definition as union of classes
  :MyFavoritePet = owl:unionOf (:cat :rabbit)
  $\forall x\ (\mathrm{MyFavoritePet}(x) \leftrightarrow (\mathrm{Cat}(x) \lor \mathrm{Rabbit}(x))\ )$

union 

```
<owl:Class rdf:ID=« MyFavoritePet">
    <owl:equivalentClass>
        <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Cat"/>
            <owl:Class rdf:about="#Rabbit"/>
        </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

```
< MyFavoritePet > a owl:Class ;
    owl:equivalentClass [
        a owl:Class ;
        owl:unionOf ( <Cat> <Rabbit> )
    ] .
```

Class description (4)

- Class definition as intersection of classes
  :Man = owl:intersectionOf (:cat :rabbit)
  $\forall x\ (\mathrm{Man}(x) \leftrightarrow (\mathrm{Person}(x) \wedge \mathrm{Male}(x))\ )$

```
<owl:Class rdf:ID= "Man">
    <owl:equivalentClass>
        <owl:Class>
        <owl:intersectonOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Person"/>
                <owl:Class rdf:about="#Male"/>
        </owl:intersectonOf >
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

```
<Man> a owl:Class ;
    owl:equivalentClass [
        a owl:Class ;
        owl:intersectionOf ( <Person> <Male> ) .
    ] .
```

More precise property descriptions

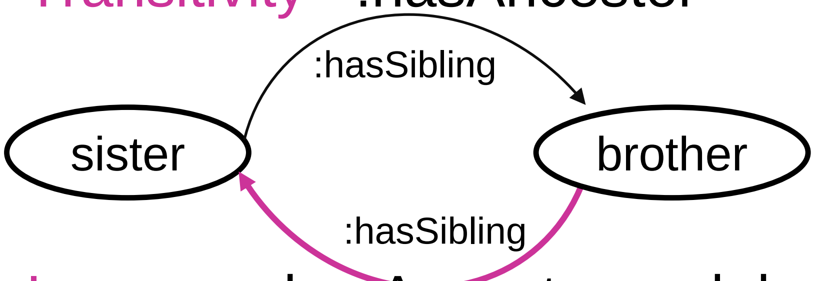
- **3 types of properties**
  - **owl:ObjectProperty** link resources :Flight :departsFrom :Airport
  - **owl:DatatypeProperty** links resources with (typed) litteral values :Flight :departsOn ^^xsd:date
  - **owl:AnnotationProperty** ignored by inference engines, just used as comments or extensions
- Constrains on properties
  - classes vs values
  - Property restriction :IntFlight :departsFrom :IntAirport
  - Cardinality :Flight :departsFrom exactly one :Airport

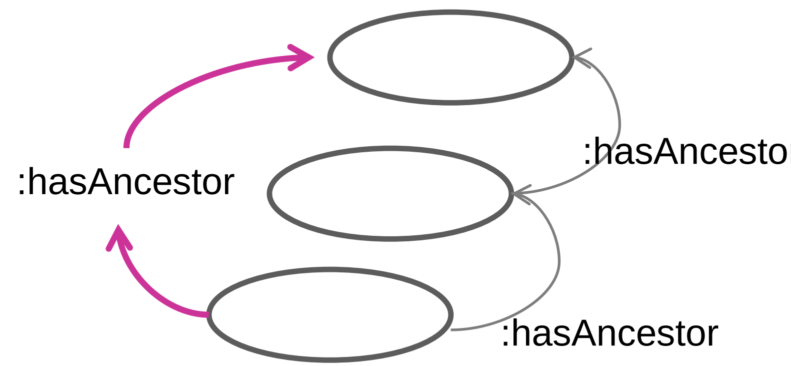
More precise property definitions

- Functional properties = mandatory property
  - Ex: if :departsFrom is functional it means
  - If f :Flight(f) then ∃ a, :Airport(a) and f :departsFrom a
  - $\forall \ell \ (\mathrm{Flight}(\ell) \rightarrow \exists\ a,\ (\mathrm{Airport}(a) \wedge \mathrm{departsFrom}\ (\ell, a))$

prop. algébriques 

- Symmetry :hasSibling
- Transitivity :hasAncestor





- Inverse :hasAncestor and :hasDescendant

Property definition : Property restriction

- Value constrains
  - Restricts all the values of a property

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:allValuesFrom rdf:resource="#Human"/>
</owl:Restriction>
```

  - Restricts at least one value of a property

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:someValuesFrom rdf:resource="#Physician"/>
</owl:Restriction>
```

  - Assigns a value to a property

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#hasParent"/>
    <owl:hasValue rdf:resource="#Marie"/>
</owl:Restriction>
```

Property definition : Property restriction

- Cardinality constraints

  nb of times that a resource can play the same role for a property with distinct values

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:maxCardinality>2</owl:maxCardinality>
</owl:Restriction>
```

owl:minCardinality

owl:cardinality

Class definition

```
<owl:Class rdf:about="#MarieChild">
	<rdfs:subClassOf>
			<owl:Restriction>
			<owl:onProperty rdf:resource="#hasParent"/>
			<owl:hasValue rdf:resource="#Marie"/>
			</owl:Restriction>
	</rdfs:subClassOf>
</owl:Class>
```

Ex: what does this class define?

```
@prefix ex: <http://example.org/>
ex:PersonList rdfs:subClassOf
          [
          a owl:Restriction ;
          owl:onProperty rdf:first ;
          owl:allValuesFrom ex:Person
          ] ,
          [
          a owl:Restriction ;
          owl:onProperty rdf:rest ;
          owl:allValuesFrom ex:PersonList
          ] .
```

Ex: what does this class define?

```
@prefix ex: <http://example.org/>

ex:Human rdfs:subClassOf [
      owl:intersectionOf (
            [
            a owl:Restriction ;
            owl:onProperty ex:hasFather ;
            owl:maxCardinality 1
            ] ,
            [
            a owl:Restriction ;
            owl:onProperty ex:hasMother ;
            owl:maxCardinality 1
            ] .
 )]
```

- Any ex:Human has these 2 properties
  - at most one Father
  - at most one Mother
- Not all things that have One Father and One mother are Humans
- Else, use owl:equivalentClass

Axiomes de propriétés

prop. algébriques 

- Propriétés algébriques de propriétés

  <owl:SymmetricProperty rdf:ID="hasSpouse/">
  <owl:TransitiveProperty rdf:ID="hasAncestor/">
  <owl:ReflexiveProperty rdf:about="#hasRelative"/>

- Relations entre propriétés
  - Relations inverse

    <owl:ObjectProperty rdf:ID="hasChild">
    < owl:inverseOf rdf:resource="#hasParent"/>
    </owl:ObjectProperty>

  - Relations équivalentes (en termes d’extension)

    owl:equivalentProperty

- Contraintes de cardinalité

  <owl:FunctionalProperty rdf:ID="#hasMother"/>
  <owl:InverseFunctionalProperty rdf:ID="#isMotherOf"/>

Annotations

- Sur les classes, propriétés et individus
  - à l'aide de propriétés instances de la classe

    owl:AnnotationProperty
    - owl:versionInfo
    - rdfs:label
    - rdfs:comment
    - rdfs:seeAlso
    - rdfs:isDefinedBy

Individus

- Description of types and properties using RDF
- Comparing individuals
  - owl:sameAs

```
<rdf:Description rdf:about="#William_Jefferson_Clinton">
<owl:sameAs rdf:resource="#BillClinton"/>
</rdf:Description>
```

```
<rdf:Description rdf:ID="FootBallTeam">
<owl:sameAs rdf:resource="ns2:#SoccerTeam"/>
</rdf:Description>
```

  - owl:differentFrom
  - owl:allDifferent

Header of an ontology file

- An ontology is a resource described using the classes owl:OntologyProperty and owl:Ontology

```
<owl:Ontology rdf:about="">
    <owl:versionInfo>v 1.1 2008</owl:versionInfo>
    <rdfs:comment>An example ontology</rdfs:comment>
    <owl:imports rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food"/>
    <owl:priorVersion rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine"/>
    <rdfs:label>Wine Ontology</rdfs:label
</owl:Ontology>
```

OWL profiles

- Chaque *profile* correspond à un sous-ensemble des primitives de OWL.
- Choisir un *profile*, c'est choisir une expressivité pour décrire une ontologie.
- Plus le degré d'expressivité est grand, plus les inférences sont complexes.

OWL1 variants

- OWL LITE : reduced set of primitives
  - Includes RDF and RDFs
  - Simple constrains
  - Computable (NP)
- OWL DL : more use constrains
  - Close to description logics
  - Ensures usability and computability
- OWL FULL : tous les constructeurs et pas de contrainte
  - A class can be an instance of another classe
  - Enables to define meta-models
  - Undecidable

OWL2 profiles

- **EL :** large number of properties and/or classes ; polynomial complexity.
- **QL :** large number of instances, enables conjunctive queries using relational DB, LOGSPACE complexity
- **RL :** scalable reasonning without loosing expressivity; inference rules in polynomial time
- **DL :** the most expressive profile

Training to build ontologies

- Download and use Use Protégé 5.5
  https://protege.stanford.edu/products.php#desktop-protege

- FHKB (Family History Knowledge Base) : a Training ontology

http://mowl-power.cs.man.ac.uk/fhkbtutorial/ontology/fhkb.owl