Bunch-of

- A volte è comodo indicare che un oggetto è **composto da parti <u>senza specificare le relazioni</u> fra queste ultime**
- Questo tipo di oggetti è catturato dalla nozione di **Bunch** (mucchio)
- **Esempi**:
  - **Bunch-of({Mela1, Mela2, Mela3})**: mucchio delle tre mele Mela1, Mela2 e Mela3
  - **Bunch-of({Andrea, Anna, Giada, Giovanni})**: gruppo costituito dalle persone indicate
- **Definizione di Bunch-of: ∀x In(x, s) ⇒ Part-of(x, Bunch-of(s))**
- Attenzione: s non è una categoria, è un insieme di elementi, in particolare Bunch-of(s) = s

(leggere)

Misure (leggere)

- A volte è utile **indicare quantità e metterle in relazione**.
- Come si possono inserire delle misure in una KB?
- Se le quantità sono *numeriche* e *puntuali* si introducono dei *predicati di misura:*
  - Member(d, Giorno) ⇒ **Durata(d) = Ore(24)**
- In altri casi più che esprimere dei precisi valori interessa rappresentare degli ordinamenti:
  - Difficoltà(SISINT) **>** Difficoltà(BASIDATI)
  - Temperatura(Inverno) **<** Temperatura(Estate)

T-box e A-box

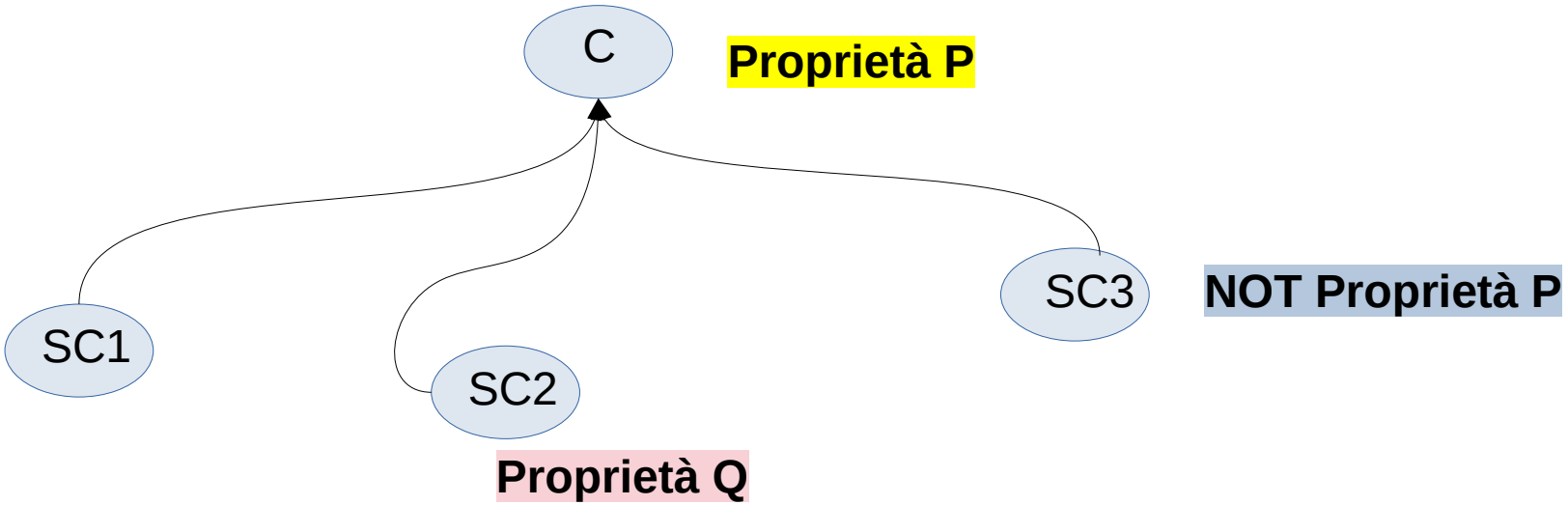
- Un'ontologia si struttura in:
  - **T-box**:
    è generale, contiene una concettualizzazione intensionale, fatta di definizioni, specializzazioni, proprietà
  - **A-box**:
    riguarda istanze specifiche, è estensionale, contiene fatti che devono essere (terminologicamente) coerenti con il contenuto della T-box

- **Esempio**:
  - T-box: Madre è sottoclasse di Donna, Madre(X) ⇒ Donna(X)
  - A-box: Anna è una Madre: Madre(Anna)

Problematiche

- Rappresentare in modo naturale la **norma** e le **eccezioni**
  - Gli uccelli solitamente volano ma alcuni fanno eccezione (struzzi, pinguini, kiwi, emù, …)
  - I cani solitamente hanno il pelo ma alcuni fanno eccezione (cane nudo peruviano, pila argentino, ...)
- Le eccezioni possono **cancellare proprietà ereditate**

Ereditarietà/Cancellazione di proprietà



C, SC, SC1, SC2: concetti di una tassonomia
relazioni **IS-A**

- Classe C ha la proprietà P
- Di default le sottoclassi ereditano P: SC1 eredita P
- Le sottoclassi possono essere ulteriormente specializzate:
  SC2 eredita P e ha anche la proprietà Q, che non vale per le classi sorelle
- Le sottoclassi possono cancellare proprità:
  SC3 è un'eccezione al default perché P non vale

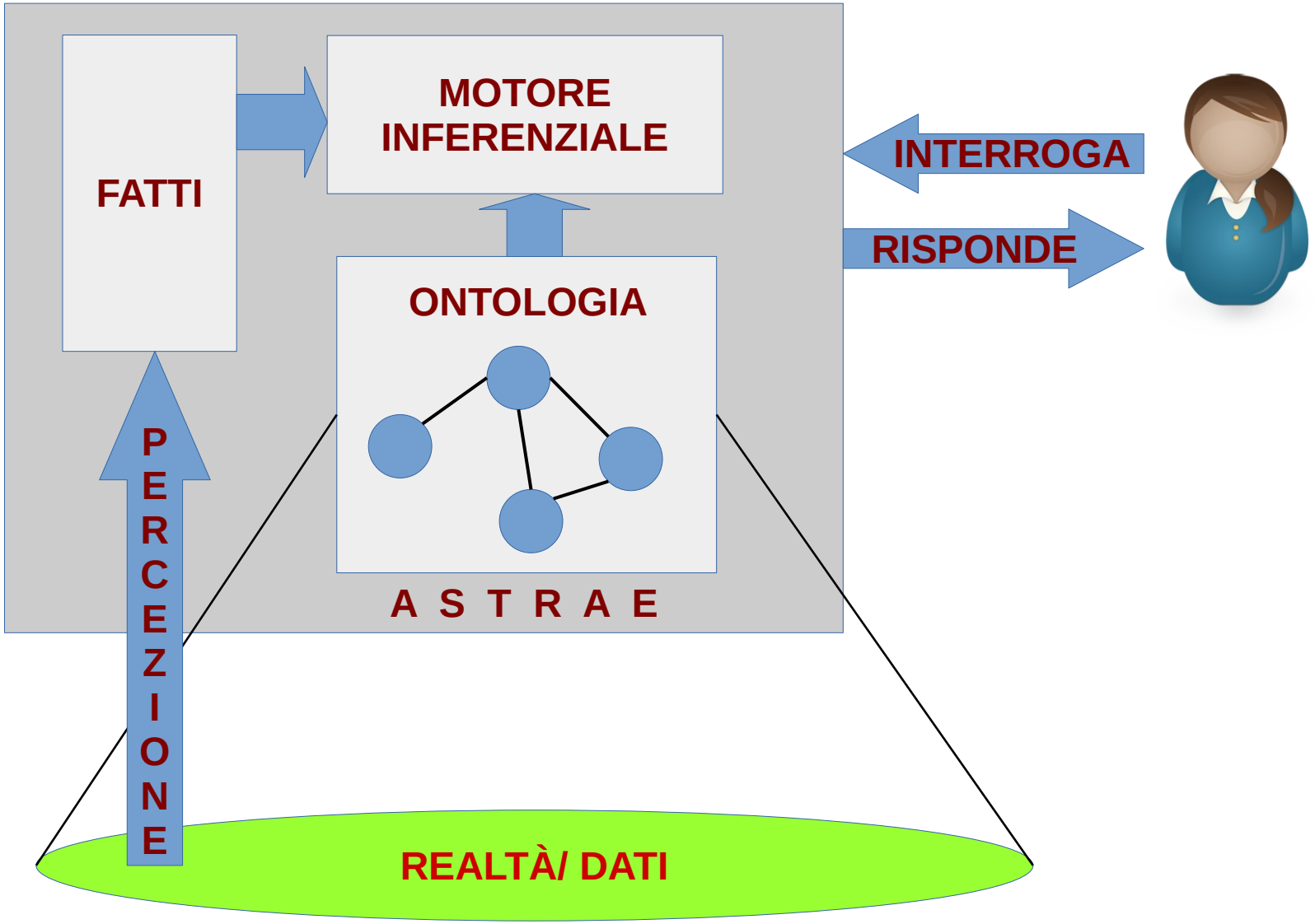
Problematiche

- Molte parole hanno tanti significati diversi
- Esempio, “cane”:
  - Animale
  - Persona vile
  - Costellazione
  - Dente d’arresto di un meccanismo
- Una stessa parola può identificare categorie che appartengono ad **alberi tassonomici diversi**
  - **esempio**: cane appartiene alla tassonomia degli animali e a quella delle costellazioni. Non vogliamo che il meccanismo di inferenza concluda che Canis Major ha il pelo né che Fido ha Sirio come stella alfa
  - come identificare concetti in modo univoco?

Non solo tassonomie

- Una KB descrittiva di un dominio può assumere forma più generale di quella tassonomica
- L'insieme dei concetti e delle loro relazioni prende in questo caso il nome di **ontologia** **(rete semantica)**
- Le tassonomie (alberi) sono un caso particolare di ontologia (grafi)

Sistemi che usano ontologie



Usi delle ontologie (o reti semantiche)

- La stessa ontologia può essere **interrogata in modi differenti**
- Alcune tipologie di interrogazione sono:

1) **Istanza appartiene a categoria?**
2) **Istanza gode di proprietà?**
3) **Differenza fra categorie?**
4) **Identificazione di istanze**

Usi delle ontologie (o reti semantiche)

- La stessa ontologia può essere **interrogata in modi differenti**
- Esempi:

1) **Istanza appartiene a categoria?**
   Fido è un mammifero?
2) **Istanza gode di proprietà?**
   Fido può volare?
3) **Differenza fra categorie?**
   Quale differenza c'è fra rocce magmatiche e rocce sedimentarie?
4) **Identificazione di istanze**
   Quali alberghi a tre stelle di Rimini offrono supporto tecnico ai ciclisti?

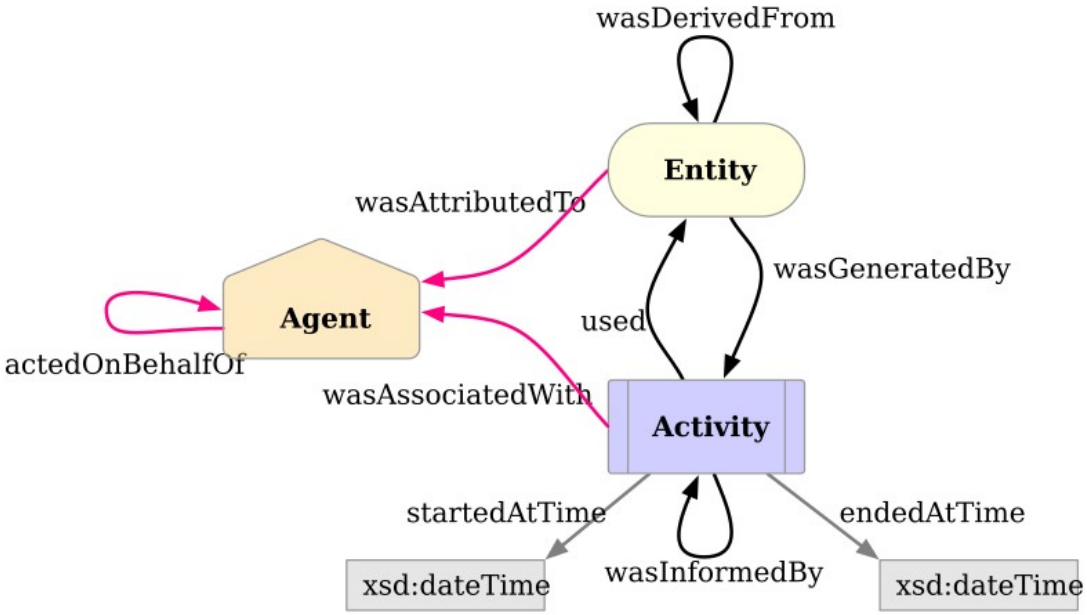
Usi delle ontologie (o reti semantiche)

- Interrogazione ed inferenza sono generalmente inseriti in **sistemi più complessi**, esempio:
  - Categorizzazione di immagini riprese da una telecamera
  - Risposta a quesiti posti in linguaggio naturale
  - Identificazione di malattie sulla base di risultati di analisi di laboratorio

Un esempio: PROV (provenance)

FACOLTATIVO

Questa è l'ontologia di PROV, un linguaggio del web semantico che permette di rappresentare, integrare, tracciare informazioni relative alla la provenienza dei dati, che applicativi di analisi possono usare come input di computazione. Sono catturati concetti essenziali che si integreranno con concetti e relazioni dello specifico dominio applcativo



Agente: persona, organizzazione, software
Attività: uso o creazione di dati
Entità: dato/documento

https://www.w3.org/TR/prov-dm/

(leggere)

Uno scenario di uso di PROV: statistiche di crimini

Lo scenario riguarda lo sviluppo di un file di statistiche di crimini (che chiameremo e0) che appartiene a un file system condiviso dei giornalisti Alice, Bob, Charles, David e Edith. Tutti possono condividere e modificare detto file. Supponiamo che il file si evolva a seguito degli eventi elencati in ordine di occorrenza:

evt1: Alice **crea (pe0)** un file vuoto in /share/crime.txt. Denotiamo questo come **e1**.

evt2: Bob **aggiunge (pe1)** la seguente linea a /share/crime.txt:
“There was a lot of crime in London last month.”
Denotiamo questo **e2**.

FACOLTATIVO

evt3: Charles manda per **mail (pe2)** i contenuti di /share/crime.txt, come attachment, che riferiremo come **e4**. (Faremo riferimento a una copia del file caricata nel mail server.)

evt4: David **edita (pe3)** il file /share/crime.txt come segue:
“There was a lot of crime in London and New-York last month.”
We denote this **e3**.

evt5: Edith invia per **mail (pe4)** i contenuti di /share/crime.txt in attachment, riferito come **e5**.

evt6: fra evt4 ed evt5, qualcuno (unspecified) applica uno **spell checker (pe5)** al file /share/crime.txt. Il file dopo lo spell checking è riferito come **e6**.

(leggere)

https://www.w3.org/TR/2011/WD-prov-dm-20111018/#a-file-scenario

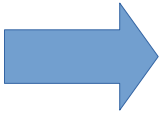
Codifica tramite PROV

FACOLTATIVO

```
processExecution(pe0,create-file,t,,[])
processExecution(pe1,add-crime-in-london,t+1,,[])
processExecution(pe2,email,t+2,,[])
processExecution(pe3,edit-London-New-York,t+3,,[])
processExecution(pe4,email,t+4,,[])
processExecution(pe5,spellcheck,,,[])

wasGeneratedBy(e0, pe0, qualifier())
wasGeneratedBy(e1, pe0, qualifier(fct="create"))
wasGeneratedBy(e2, pe1, qualifier(fct="save"))
wasGeneratedBy(e3, pe3, qualifier(fct="save"))
wasGeneratedBy(e4, pe2, qualifier(port="smtp", section="attachment"))
wasGeneratedBy(e5, pe4, qualifier(port="smtp", section="attachment"))
wasGeneratedBy(e6, pe5, qualifier(file="stdout"))

used(pe1,e1,qualifier(fct="load"))
used(pe3,e2,qualifier(fct="load"))
used(pe2,e2,qualifier(fct="attach"))
used(pe4,e3,qualifier(fct="attach"))
used(pe5,e3,qualifier(file="stdin"))
```



(leggere)

Codifica tramite PROV

FACOLTATIVO

```
wasDerivedFrom(e2,e1)
wasDerivedFrom(e3,e2)
wasDerivedFrom(e4,e2,pe2,qualifier(port=smtp, section="attachment"),qualifier(fct="attach"))
wasDerivedFrom(e5,e3,pe4,qualifier(port=smtp, section="attachment"),qualifier(fct="attach"))

wasComplementOf(e1,e0)
wasComplementOf(e2,e0)
wasComplementOf(e3,e0)
wasComplementOf(e6,e3)

entity(a1, [ type="Person", name="Alice" ])
agent(a1)
entity(a2, [ type="Person", name="Bob" ])
agent(a2)
entity(a3, [ type="Person", name="Charles" ])
agent(a3)
entity(a4, [ type="Person", name="David" ])
agent(a4)
entity(a5, [ type="Person", name="Edith" ])
agent(a5)
```

```
wasControlledBy(pe0,a1, qualifier(role=
"creator"))
wasControlledBy(pe1,a2, qualifier(role=
"author"))
wasControlledBy(pe2,a3, qualifier(role=
"communicator"))
wasControlledBy(pe3,a4, qualifier(role=
"author"))
wasControlledBy(pe4,a5, qualifier(role=
"communicator"))
```

(leggere)

Graficamente

FACOLTATIVO

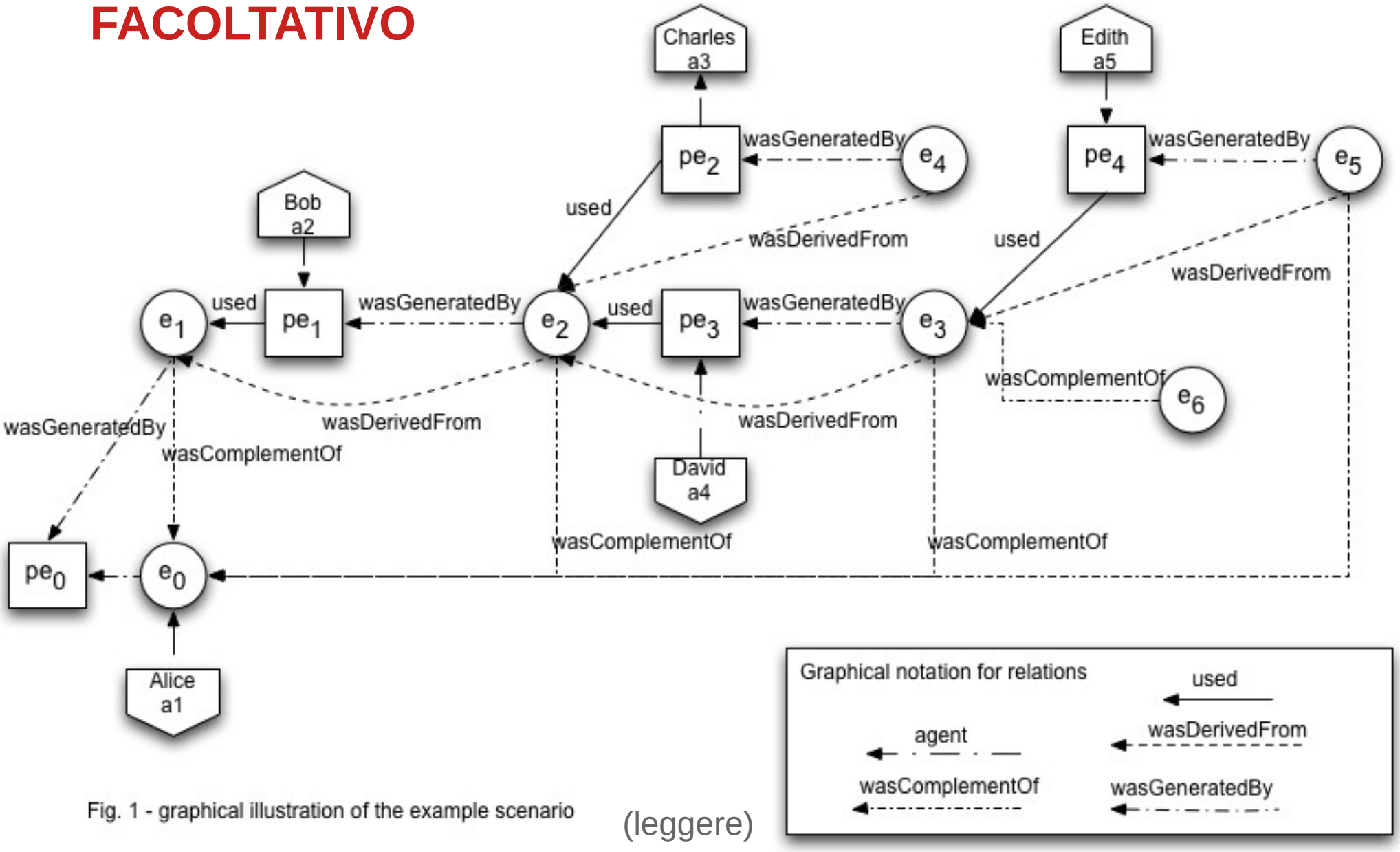
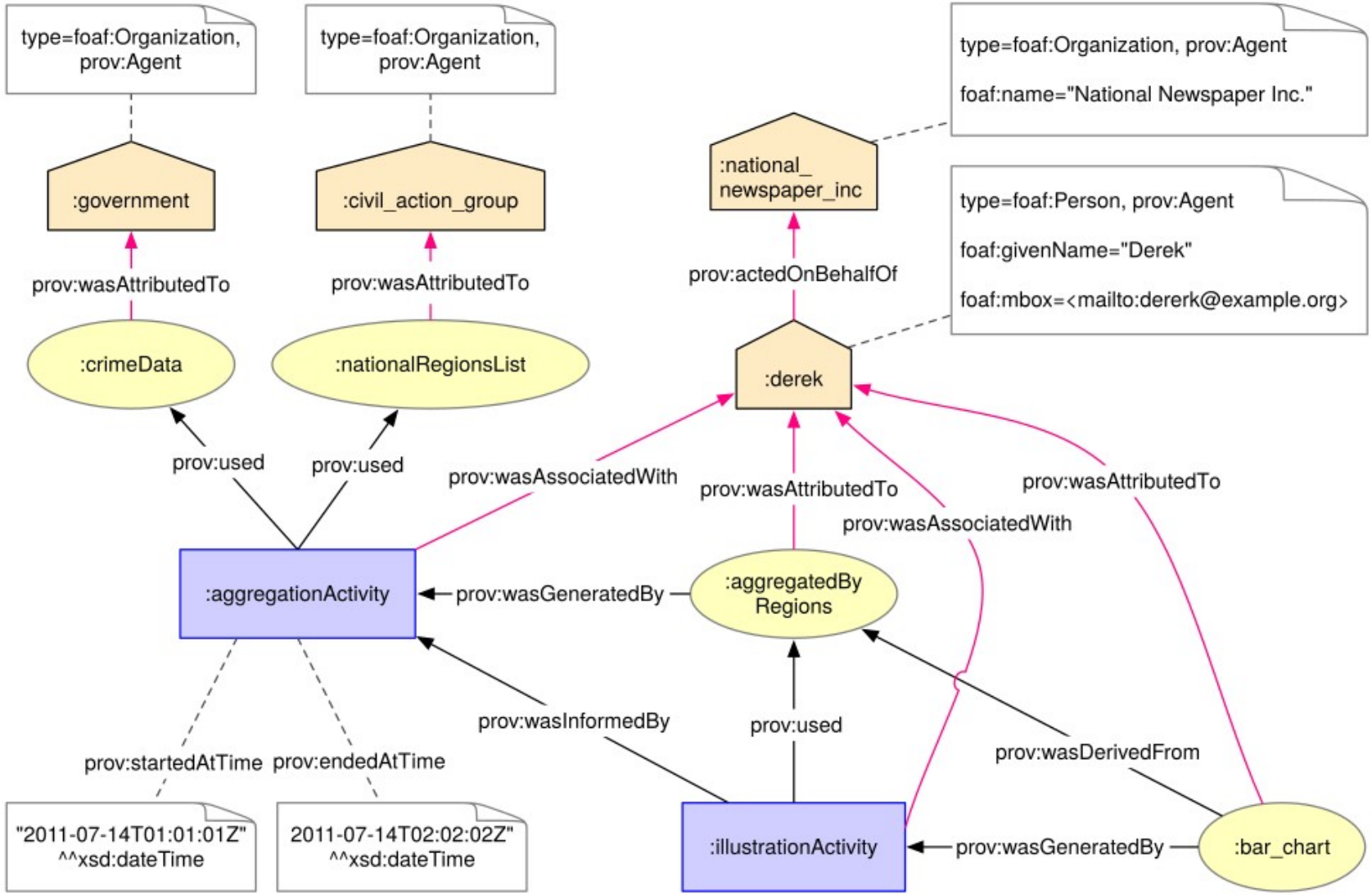


Fig. 1 - graphical illustration of the example scenario

(leggere)

Esempio versione più recente

FACOLTATIVO



(leggere)

Tipi di interrogazioni

FACOLTATIVO

Su una rappresentazione come la precedente è possibile automatizzare interrogazioni tipo:

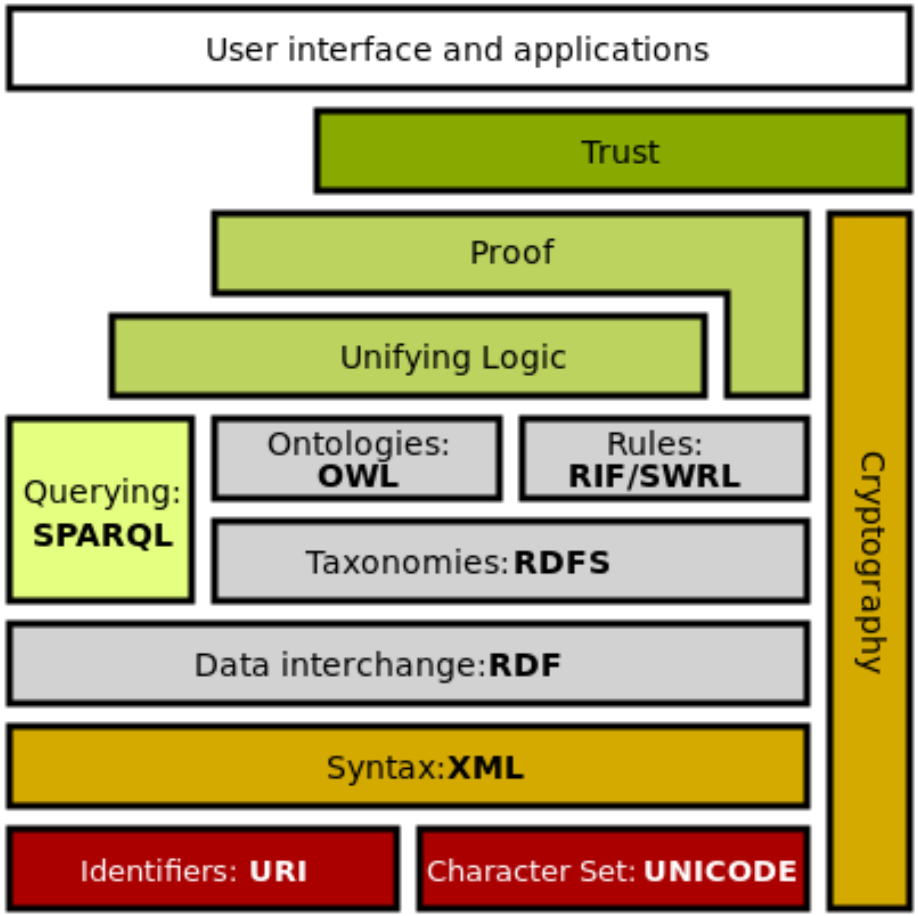
1) Su cosa si basa questa risorsa (esempio: su quali fonti)?
2) Com'è stata costruita?
3) Chi l'ha fatta?
4) Quando è stata fatta?
5) Quali nuove risorse si basano su di questa?
6) Per quali scopi è stata utilizzata questa risorsa?
7) Chi l'ha usata?
8) Quali altre risorse sono derivate dalle stesse fonti di questa?

Esempi di utilizzo concreto:

- verificare il rispetto delle norme sulla privacy
- verificare la liceità d'uso, per esempio, legata ai diritti associati al materiale audio/video

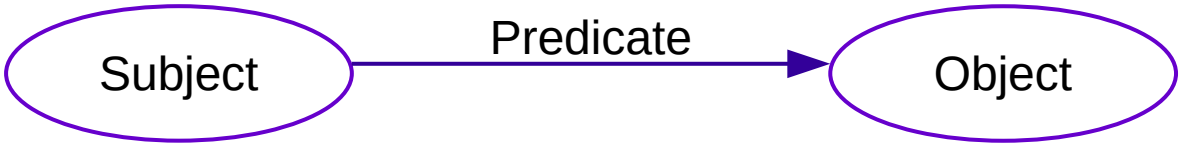
Semantic Web

- Da World-Wide Web a **Semantic Web**: estensione del WWW in cui il materiale pubblicato è arricchito da **metadati** che abilitano l'interpretazione, l'inferenza, l'interrogazione, l'elaborazione automatica
- Termine ideato da Tim Berners-Lee (premio Turing 2016)
- Linguaggi standardizzati dal **consorzio W3C**



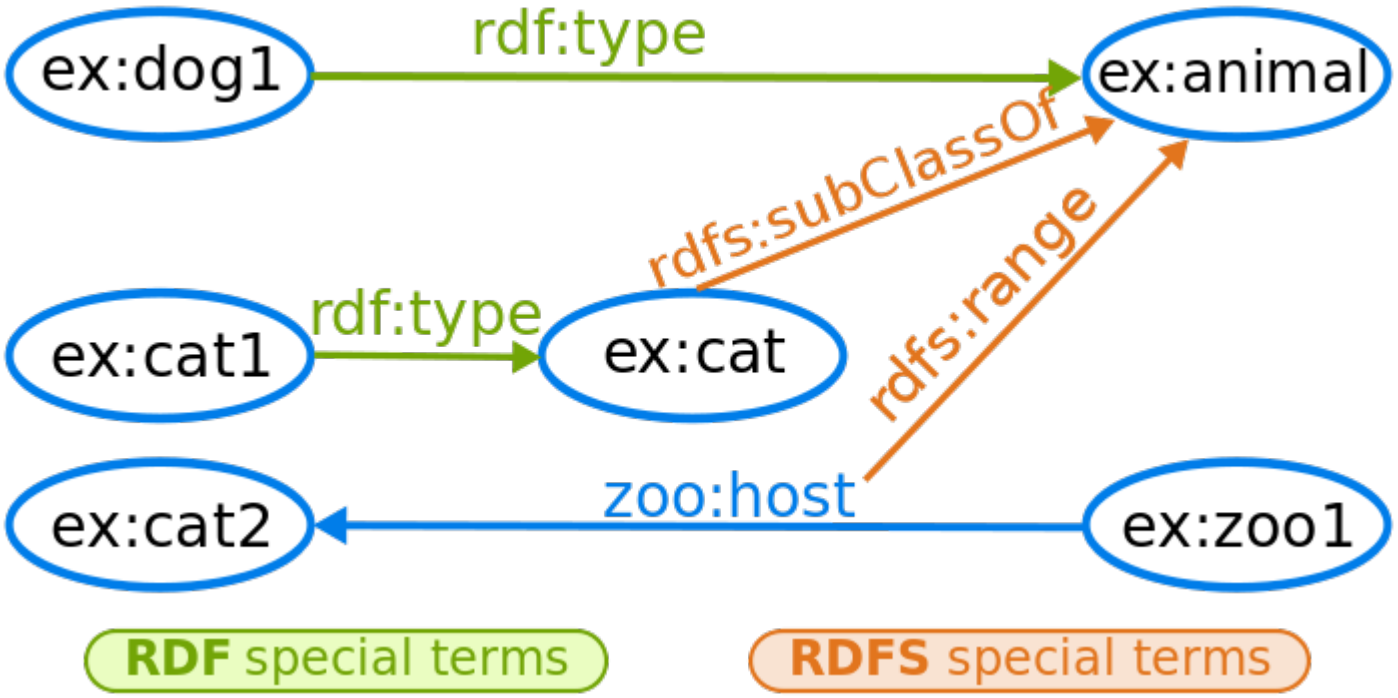
Data interchange: RDF (resource description framework)

- È un modello (e linguaggio) di rappresentazione
- È la base di linguaggi come **OWL** e SKOS, che permettono di scrivere ontologie, **FOAF** (friend of a friend) per applicazioni sociali, ecc.
- Conoscenza espressa da **statement**, cioè triple **soggetto – predicato – oggetto**: Il predicato mette in relazione soggetto e oggetto



- Soggetto, predicato e oggetto sono **IRI** (internationalized resource identifier, per esempio degli URL)
- **RDFS (RDF Schemas)** permette di realizzare tassonomie appoggiandosi a RDF
- Un insieme di triple costituisce un **grafo RDF**

Esempio



Notazione: **namespace:resource**

Esempio

- Vogliamo esprimere che l'autrice di una certa pagina web è una certa Enrica Genovese. Definiamo lo statement:
  - **Soggetto**: http://xx.yy.zz/esempio.html
  - **Predicato**: author
  - **Oggetto**: Enrica Genovese
- RDF è realizzato in **XML**, la rappresentazione finale sarà:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:a="http://nome_di_un_dominio/schema_autore/">
  <rdf:Description about="http://xx.yy.zz/esempio.html">
    <a:author>
      Enrica Genovese
    </a:author>
  </rdf:Description>
</rdf:RDF>
```

NB: RDF non è pensato per essere direttamente scritto o interpretato da un essere umano

Passi per costruire un'ontologia

- Supponiamo che ci venga chiesto di esporre i dati contenuti in un database utilizzando un formalismo sul quale sia possibile fare delle inferenze:
  1) Costruiamo un'ontologia
  2) Creiamo uno strumento che permetta di descrivere i dati contenuti nel DB nel linguaggio ontologico costruito
- DB di partenza, es. mantiene le informazioni relative a un gruppo di canili

Costruire un'ontologia

- **Identificazione dei concetti:**
  - Elencare tutti i concetti riferiti nel DB di partenza
  - I concetti sono solitamente catturati da sostantivi
  - Definire per ciascuno un'etichetta e una breve descrizione
  - In seconda passata identificazione delle sottoclassi

Costruire un'ontologia

- **Identificazione dei concetti:**
- **Esempio**:
  - Cane: specie animale
  - Bassotto: sottospecie di cane
  - Gatto: specie animale
  - Meticcio: incrocio di sottospecie
  - Cucciolo: animale di età inferiore o uguale a 6 mesi
  - Orfano: animale privo di genitori
  - TagliaMedia: cane di dimensioni non superiori a …
  - Struttura: nome che identifica un canile
  - …

Costruire un'ontologia

- **Identificazione delle proprietà:**
  - Elencare tutte le relazioni catturate nel DB di partenza
  - Le relazioni tipicamente sono esprimibili come verbi
  - Definire per ciascuna un'etichetta e una breve descrizione

Costruire un'ontologia

- **Identificazione delle proprietà:**
- **Esempio**:
  - ospitatoDa: indica quale canile ospita un certo animale
  - coloreManto: indica il colore dell'animale
  - HaDisabilità: se necessario indica la disabilità di un animale
  - haTatuaggio: indica l'identificativo tatuato sull'animale, non obbligatorio
  - haMicrochip: ...
  - ...

Costruire un'ontologia

- Esistono ontologie già definite che possono essere (parzialmente) riutilizzate?
- Non proprio, nel caso dell'esempio, ma esistono ontologie utili come la "Wildlife Ontology" della BBC (http://www.bbc.co.uk/ontologies/wo), che contiene concetti come Species, Feeding Habit e Behavioural Pattern
- Fare riferimento ai concetti già definiti ogni volta che è possibile

Costruire un'ontologia

- Scrivere tutto quanto definito nei passi precedenti in un linguaggio per ontologie (RDF o OWL, per esempio). Avremo così la nostra **T-box**
- Annotare i dati, riempiendo la **A-Box**
- Validare il risultato e raffinare la **KB**
- **Costruire un sistema di interrogazione**: le applicazioni che sfrutteranno l'ontologia e i dati annotati appoggiandosi a dei motori inferenziali
- **Esempio**: una **guida alla scelta di un cane da adottare,** che sostituisca consigli statici tipo:
  "Che cane adottare? La taglia giusta. Cani diversi per taglia hanno anche esigenze diverse, ed esigono quindi proprietari con caratteristiche e possibilità diverse. Il cane di piccola taglia non necessita di grandi spazi abitativi e può vivere e lasciar vivere tutti anche in un piccolo appartamento. Certo anche il cane piccolo avrà voglia di scatenarsi in qualche corsa, ma non è un'esigenza così forte come per un cane di grande mole. In più è facilmente trasportabile (mezzi pubblici, negozi,ecc). …" (da: https://www.adottauncane.net/consigli.html)

Strumenti

- **Protégé**: **editor, ambiente di sviluppo e mantenimento di KB** sviluppato presso lo Stanford Center for Biomedical Informatics Research (Stanford Univ. School of Medicine). Versione desktop e web.
- **CEL, FaCT++, HermiT, Pellet, Racer Pro**: **reasoner** (esempio, Racer Pro: http://franz.com/agraph/racer/racer_features.lhtml)
- **Alignment API**: API per effettuare compiti di **ontology alignment**
- **OWLTools**: **java API**
- **Sesame**: basato su Java permette di parsificare, conservare, interrogare KB in RDF e di fare inferenze
- **Tripliser**: costruisce grafi RDF, utile quando si gestiscono grandi volumi di dati
- https://www.w3.org/RDF/Validator/: valida RDF e produce grafi RDF

Strumenti

- **Apache Jena**: framework Java per lo sviluppo di applicazioni orientate al web semantico. Permette di leggere e scrivere RDF
- **AllegroGraph**: triplestore (DB a triple) per RDF
- **Turtle, N3**: linguaggi di rappresentazione alternativi di RDF
- **SPARQL**: linguaggio di interrogazione per rappresentazioni RDF
- …

Applicazioni

- **Linked (Open) Data**
- **DBpedia**: wikipedia + annotazioni semantiche
- **CreativeCommons**: fornisce descrizioni delle licenze in RDF per consentire l'applicazione di tecniche di ragionamento automatico
- **FOAF**: friend of a friend ontology
- **Press Association**: agenzia stampa che pubblica notizie annotate semanticamente
- *Ecc.*

Esempio: CC

creative commons

Describing Copyright in RDF

Creative Commons Rights Expression Language

The Creative Commons Rights Expression Language (CC REL) lets you describe copyright licenses in RDF. For more information on describing licenses in RDF and attach descriptions to digital works, see CC REL in the Creative Commons wiki.

Classes

Work

a potentially copyrightable work

License

a set of requests/permissions to users of a Work, e.g. a copyright license, the public domain, information for distributors

Jurisdiction

the legal jurisdiction of a license

Permission

an action that may or may not be allowed or desired

Requirement

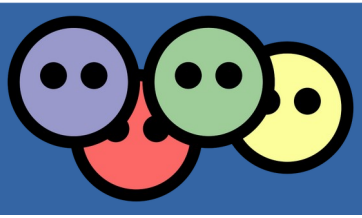
an action that may or may not be requested of you

Prohibition

something you may be asked not to do

Permissions

Esempio: FOAF



- Progetto finalizzato a linkare persone che usano il web Definisce un ricco vocabolario, il cui nucleo contiene le categorie:

Agent, Person
Name, title
Img, depiction (depicts)
FamilyName, givenName
Knows, based_near
Age, made (maker)
primaryTopic (primaryTopicOf)
Project, Organization
Group, member
Document
Image

Class: foaf:Agent

Agent - An agent (eg. person, group, software or physical artifact).

Status: stable

Properties include: gender yahooChatID account birthday icqChatID aimChatID jabberID made mbox interest tipjar skypeID topic_interest age mbox_sha1sum status msnChatID openid holdsAccount weblog

Used with: maker member

Has Subclass Group Person Organization

The Agent class is the class of agents; things that do stuff. A well known sub-class is Person, representing people. Other kinds of agents include Organization and Group.

The Agent class is useful in a few places in FOAF where Person would have been overly specific. For example, the IM chat ID properties such as jabberID are typically associated with people, but sometimes belong to software bots.

Fonte: http://xmlns.com/foaf/spec/#sec-glance

OWL 2: Web Ontology Language

- **OWL2:** linguaggio **dichiarativo** del semantic web ideato per **definire ontologie** tramite specifica di classi (categorie), proprietà, individui e valori
- Le ontologie OWL possono essere **pubblicate sul web** e **riferite da altre ontologie**, per costruire KB più complesse e raffinate

Modellare conoscenza in OWL

- Il linguaggio prevede tre elementi:
  - **Entità**: elementi usati per riferirsi a oggetti del mondo reale. Sono elementi atomici che possono essere usati negli assiomi
  - **Assiomi**: affermazioni (statement) di base espressi da un'ontologia OWL
  - **Espressioni**: combinazioni di entità che costituiscono descrizioni complesse sulla base di altre

Modellare conoscenza in OWL

- Il linguaggio prevede tre elementi:
  - **Entità:**
    oggetti, categorie e relazioni che costituiscono gli statement
    Maria, donna, Luca, Anna, essere-sposati
  - **Assiomi**:
    Maria è una donna
    Luca e Anna sono sposati
  - **Espressioni**:
    definite tramite costruttori, supponiamo di avere la categoria medico e la categoria donna, è possibile combinarle definendo la categoria delle donne medico

File OWL

FACOLTATIVO

- **Preambolo**
  Prefix(:=<http://example.com/owl/families/>)
  Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
  Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
  Prefix(**owl:=<http://www.w3.org/2002/07/owl#>**)
  Ontology(<http://example.com/owl/families>
  …)
- I prefissi permettono di fare sinteticamente riferimento a elementi definiti in altre ontologie (o altri file)
- Il prefisso più l'etichetta sono composti nell'identificatore dell'elemento di interesse, esempio **owl:Thing** diventa **http://www.w3c.org/2002/07/owl#Thing**
- **Ontology specifica l'URI del file contenente l'ontologia definita.** Sarà generalmente salvato nel file system ed eventualmente accessibile via web

OWL: sintassi

FACOLTATIVO

- È possibile scrivere ontologie OWL utilizzando sintassi differenti:

1) **Functional-Style (quella che useremo):**
   ClassAssertion(:Persona :Maria)
   noi useremo questa notazione per introdurre alcuni elementi sintattici importanti del linguaggio

2) **RDF/XML (prima a essere utilizzata):**
   <Persona rdf:about="Maria"/>

3) Turtle:
   :Maria rdf:type :Person .

4) Manchester:
   Individual: Maria

Elementi della KB

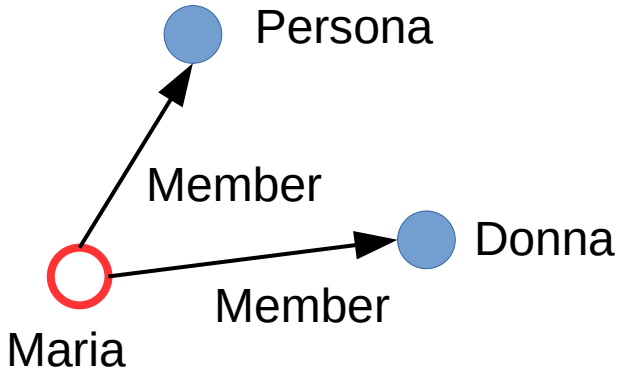
FACOLTATIVO

- **Nella terminologia di OWL:**
  - Oggetti (in FOL costanti): **individui**
  - Categorie (in FOL predicati unari): **classi**
  - Relazioni (in FOL predicati binari): **proprietà**
- **Esempi:**
  - **Dichiarazione di individuo (assioma):**
    Declaration(NamedIndividual(:Maria))
  - **Dichiarazione di classe (assioma):**
    Declaration(Class(:Persona))
  - **Dichiarazione di proprietà (assioma):**
    Declaration(ObjectProperty(:donna))

Classi e istanze

FACOLTATIVO

- **ClassAssertion** lega un'istanza a una classe, il primo argomento identifica la classe, il secondo l'individuo del quale si dichiara una proprietà
- **Esempi**:
  ClassAssertion(:Persona :Maria)
  ClassAssertion(:Donna :Maria)
  Maria è un individuo, membro (istanza) delle classi Persona e Donna

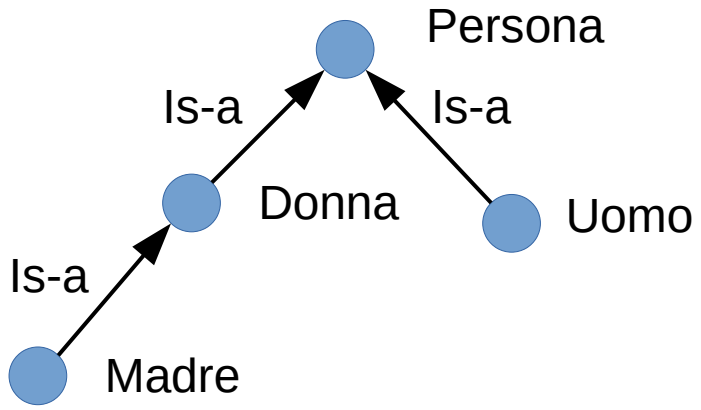


Relazione sottoclasse, equivalent, disjoint

FACOLTATIVO

- **SubClassOf** permette di definire tassonomie tramite la specificazione di relazioni di sottoclasse. Il primo parametro indica la sottoclasse, il secondo la sopraclasse. Tutte le istanze della sottoclasse apparterranno anche alla sopraclasse

- **Esempi**:
  SubClassOf(:Donna :Persona)
  SubClassOf(:Madre :Donna)
  Madre è sottoclasse di Donna, che è sottoclasse di Persona (axioms)



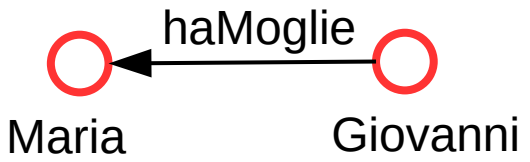
- **EquivalentClasses(:Persona :Umano)** indica l'equivalenza di due classi. È come dire che Persona è sottoclasse di Umano e al contempo Umano lo è di Persona (axiom)

- **DisjointClasses(:Donna :Uomo)** indica che le due classi non hanno istanze in comune (axiom)

Proprietà

FACOLTATIVO

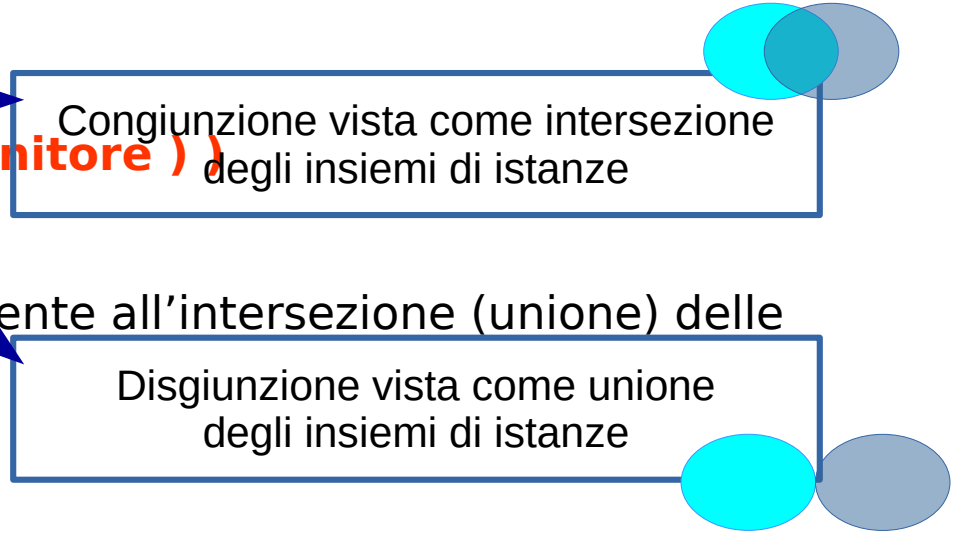
- **ObjectPropertyAssertion** permette di legare due individui tramite una proprietà
- **Esempio**
  ObjectPropertyAssertion(:haMoglie :Giovanni :Maria)
  l'individuo Giovanni ha per moglie l'individuo Maria (assioma)
- È possibile, se necessario, specificare che una proprietà è sottoproprietà di un'altra, esempio avere moglie è sottoproprietà di avere un coniuge (axiom):
  SubObjectPropertyOf(:haMoglie :haConiuge)
- È opzionalmente possibile caratterizzare dominio e codominio di una proprietà
- **Esempio**
  la proprietà haMoglie caratterizza le istanze di Uomo legandole a istanze di Donna
  ObjectPropertyDomain(:haMoglie :Uomo)
  ObjectPropertyRange(:haMoglie :Donna)



Classi complesse

FACOLTATIVO

- È possibile specificare classi, e relazionarle, utilizzando opportuni costrutturi, esempi:
- **EquivalentClasses(:Madre**
  ObjectIntersectionOf(:Donna :Genitore))
  EquivalentClasses(:Genitore
  ObjectUnionOf(:Madre :Padre))
  la classe Madre (Genitore risp.) è equivalente all'intersezione (unione) delle classi Donna e Genitore (Madre e Padre)

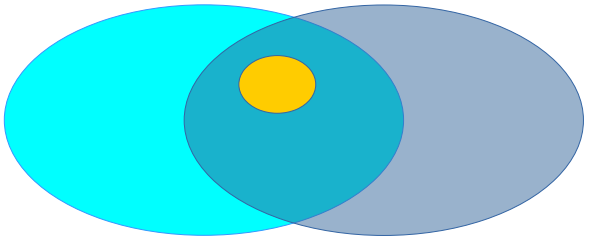


Intersection, Union … sono esempi di espressioni

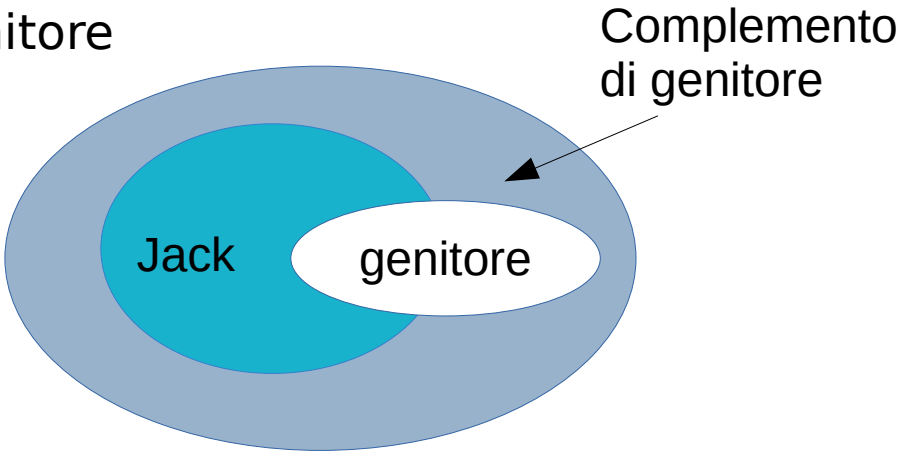
Classi complesse

FACOLTATIVO

- **SubClassOf(:Nonno**
  ObjectIntersectionOf(:Uomo :Genitore))
  Nonno è sottoclasse dell'intersezione fra Uomo e Genitore



- **ClassAssertion(**
  ObjectIntersectionOf(:Persona
  ObjectComplementOf(:Genitore))
  :Jack)
  L'individuo Jack è una Persona non Genitore



Quantificazione esistenziale e universale

FACOLTATIVO

QUANTIFICAZIONE ESISTENZIALE

EquivalentClasses(
:Genitore
ObjectSomeValuesFrom(:haFiglio :Persona))

La classe Genitore è l'insieme di quegli individui che hanno almeno un'istanza di Persona che è loro figlio

QUANTIFICAZIONE UNIVERSALE

EquivalentClasses(
:PersonaFelice
ObjectAllValuesFrom(:haFiglio :PersonaFelice))

una persona è felice quando tutti i suoi figli sono felici. Include come felici anche tutte le persone che non hanno figli

Datatype

FACOLTATIVO

- I data type permettono di specificare insiemi di valori tramite i quali è possibile descrivere istanze e proprietà, per esempio:

DataPropertyDomain(:haEta :Persona)
DataPropertyRange(:haEta xsd:nonNegativeInteger)

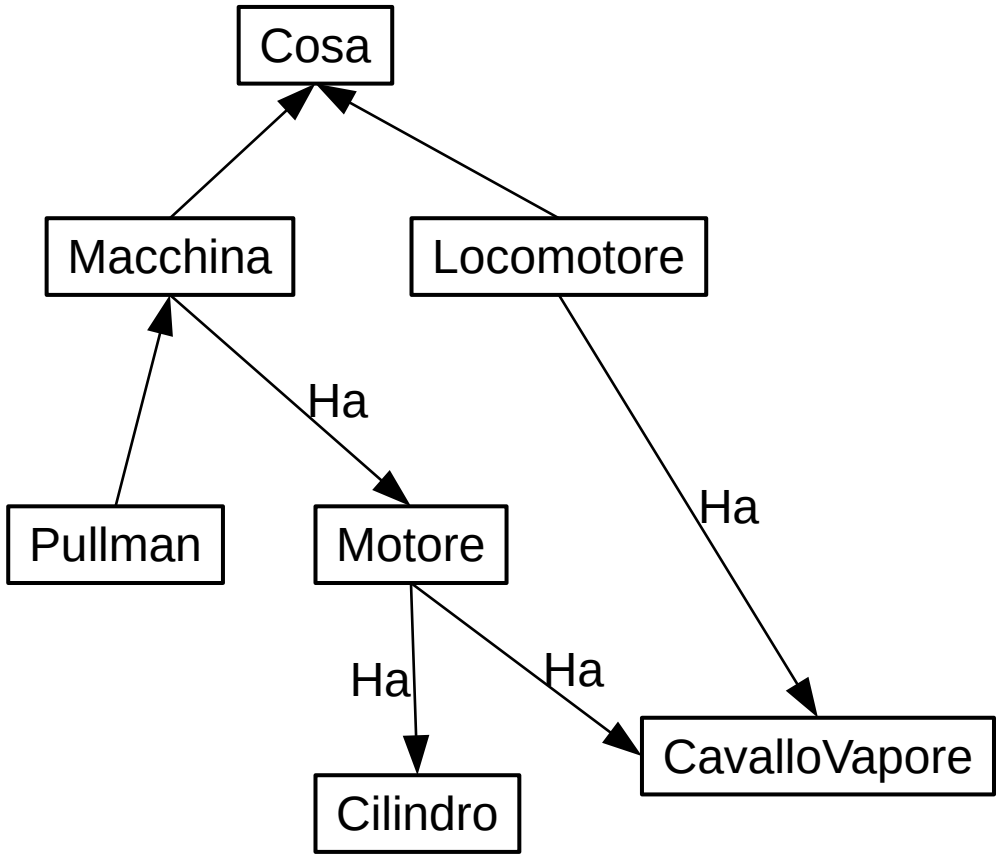
Allineamento ontologico

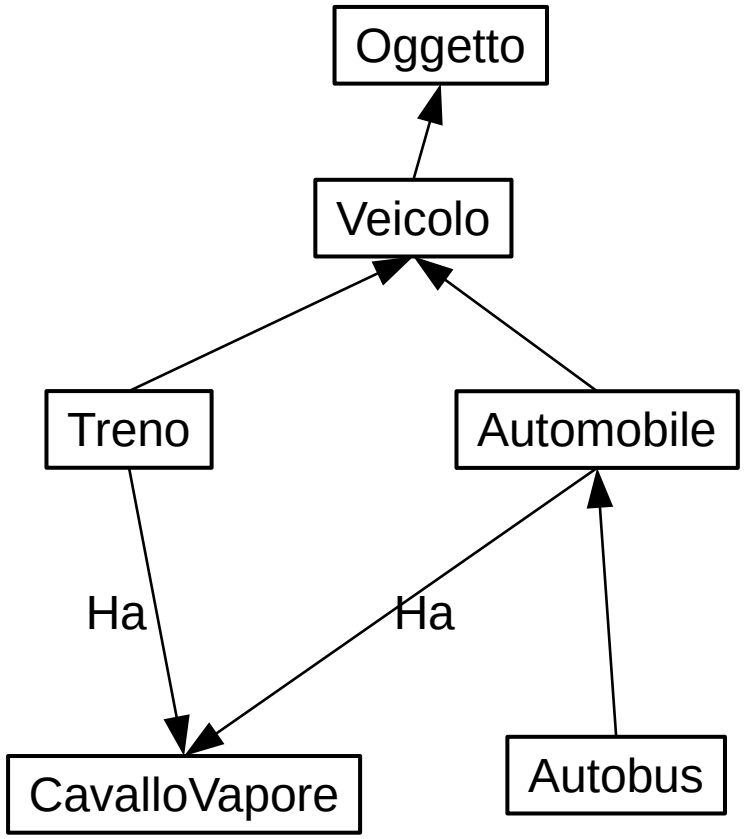
- Un problema frequente è combinare concettualizzazioni sviluppate separatamente e indipendentemente
- **Matching di ontologie**: date due ontologie O1 e O2 costruire un allineamento individuando le relazioni fra concetti corrispondenti
- La corrispondenza in generale sarà imperfetta

Esempio

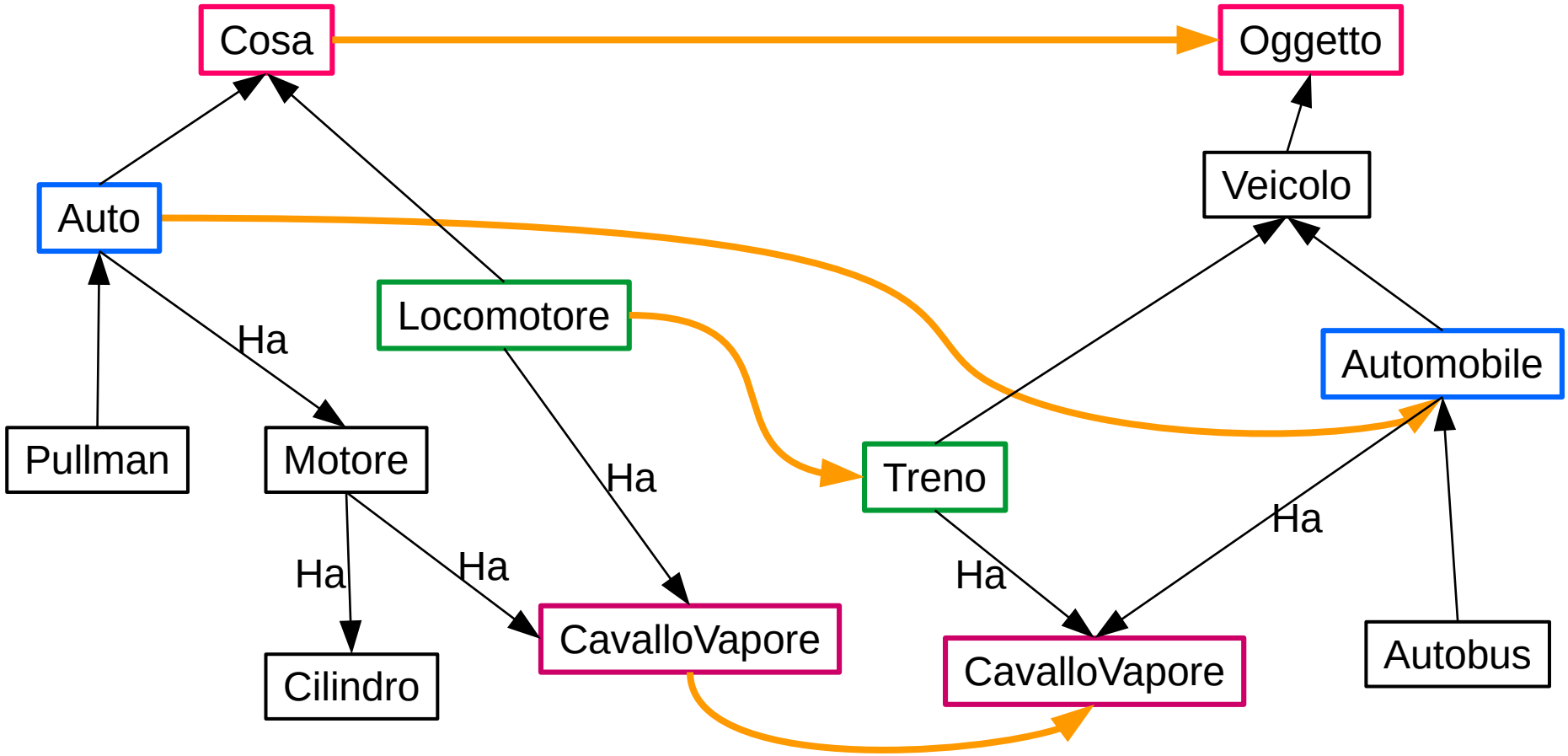
- Due agenti che interagiscono per scambio di messaggi fanno riferimento a **ontologie condivise**: (1) per riferisi al dominio del discorso; (2) per fare riferimento alle stesse definizioni di atti comunicativi e protocolli di interazione
- Fra le proposte **FIPA Ontology** (FIPA: Foundation for Physical Agents) postulano la presenza di un agente dedicato a gestire ontologie (ontology agent) e che fornisce fra i suoi servizi:
  - Discovery di ontologie pubbliche
  - Traduzione di espressioni in ontologie differenti
  - Rispondere a query relative alle differenze fra termini o ontologie

Allineamento ontologico, esempio





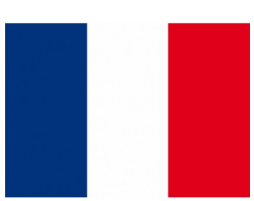
Allineamento ontologico, esempio



Relazioni fra ontologie

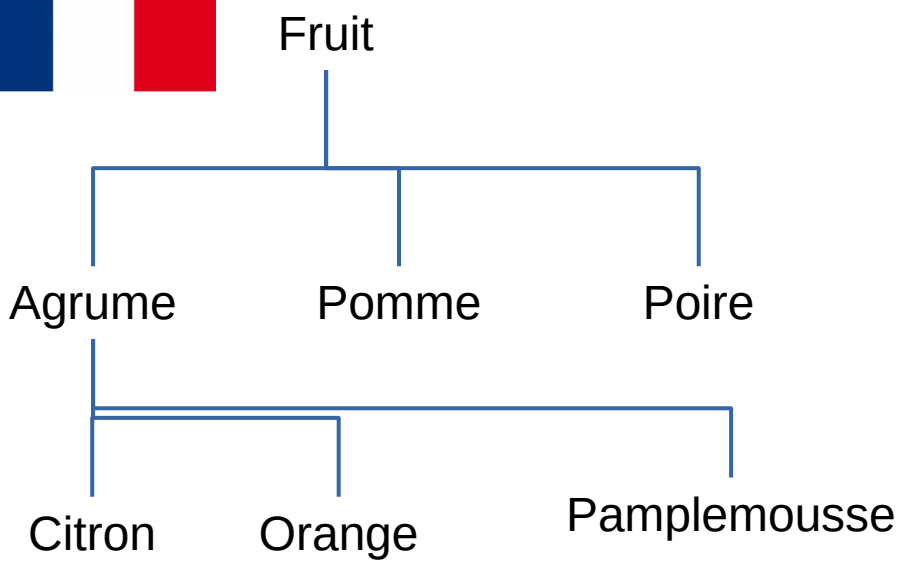
- **Identical**: O1 e O2 sono la stessa ontologia (esempio: un'ontologia e la sua copia in un disco mirror)
- **Equivalent**: condividono vocabolario e assiomatizzazione ma sono espresse in linguaggi differenti (esempio: SKOS e RDF)
- **Extension**: O1 estende O2 quando tutti i simboli definiti in O2 sono preservati in O1 insieme alle loro proprietà e relazioni ma non vale il viceversa
- **Weakly-Translatable**: siano Osource e Odest due ontologie, è possibile tradurre espressioni Osource in espressioni Odest con perdita di informazione
- **Strongly-Translatable**: slide
- **Approx-Translatable**: slide

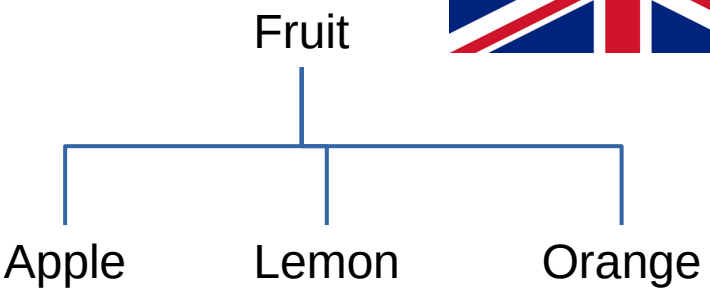
Weakly-Translatable, esempio



Osource

Odest





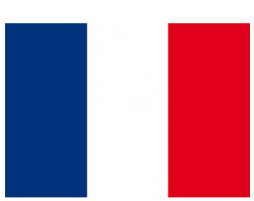
Osource è weakly-translatable in Odest usando il mapping:
Pomme → Apple, Citron → Lemon, Orange → Orange, Fruit → Fruit

Si ha perdita di informazione relativa a Poire, Pamplemousse e alla sottocategorizzazione di Agrume in Citron, Orange e Pamplemousse

Strongly-Translatable

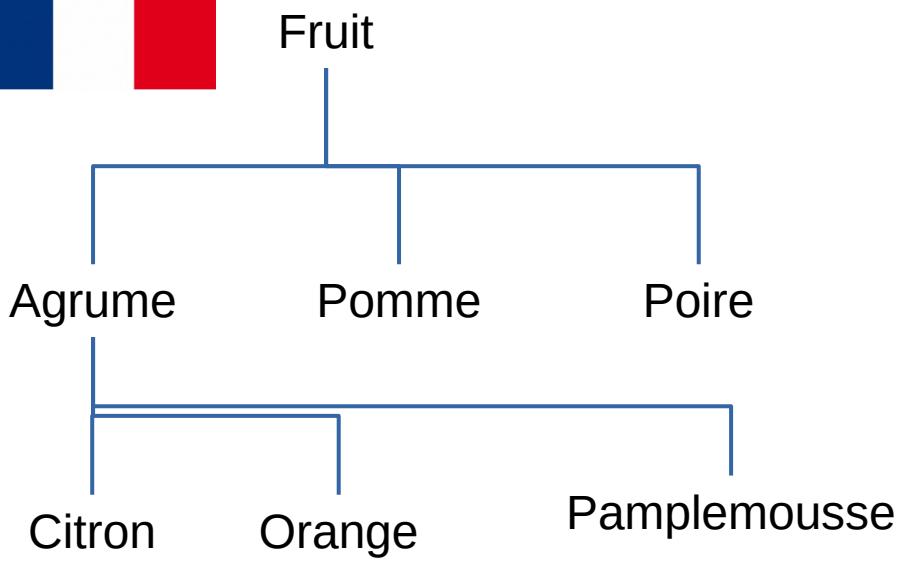
- Osource è strongly-translatable in Odest quando:
  - Il suo vocabolario è totalmente mappabile in quello di Odest
  - l’assiomatizzazione di Osource vale in Odest
  - Non c’è perdita di informazione
  - Non si introducono inconsistenze

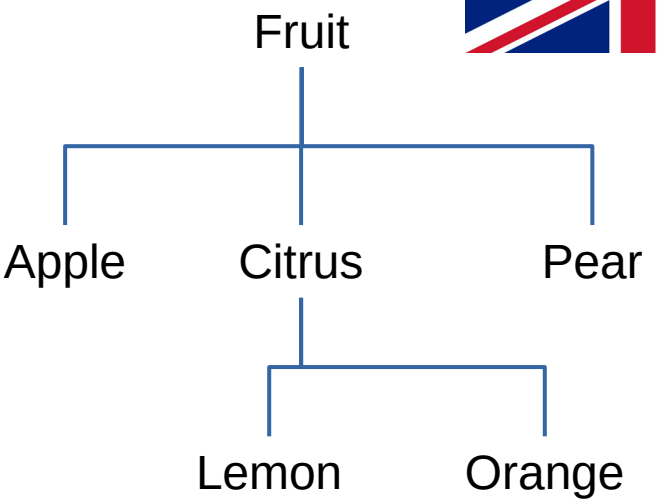
Strongly-Translatable, esempio



Odest

Osource





Osource è strongly-translatable in Odest usando il mapping:
Fruit → Fruit, Apple → Pomme, Pear → Poire, Citrus → Agrume, Lemon → Citron, Orange → Orange

Non si ha perdita di informazione. Si noti però che in Osource non vi è un concetto equivalente a Pamplemousse

Approx-Translatable

- Osource è **Approx-Translatable** in Odest quando è **Weakly-Translatable e possono essere introdotte delle inconsistenze**
- Tipicamente questo accade perché alcuni concetti che dovrebbero corrispondere l'uno con l'altro risultano affini ma non identici
- **Esempio**:
  Il coriandolo, a seconda della tradizione culinaria considerata, viene visto come affine al prezzemolo (là dove se ne usano le foglie) o al pepe (là dove se ne utilizzano i semi). È la stessa pianta ma a seconda dell'ontologia di riferimento potrà avere proprietà (ontologiche) differenti