Ingegneria della conoscenza

"Giunti alla logica del prim'ordine come strumento di rappresentazione, cerchiamo di comprendere i fondamenti della definizione di una KB"

Concettualizzazione

- **Quali sono gli aspetti fondamentali della costruzione e del mantenimento delle KB?**
- Il mondo reale non è fatto di formule, è fatto di oggetti
- Gli uomini concettualizzano tali oggetti e le relazione che questi intrattengono gli uni con gli altri
- Esempio: categorizzazione degli oggetti

Concettualizzazione

- In logica FOL abbiamo degli strumenti di base:
  - **Predicati**: *proprietà*
  - **Funzioni**: *riferimenti a elementi del dominio*
  - Nel rappresentare la conoscenza possiamo scegliere se un aspetto vada catturato da un predicato o da una funzione
- Visto però che il *ragionamento avviene per lo più sul piano delle concettualizzazioni,* come usare predicati e funzioni per favorire la rappresentazione di queste ultime?

Categorie

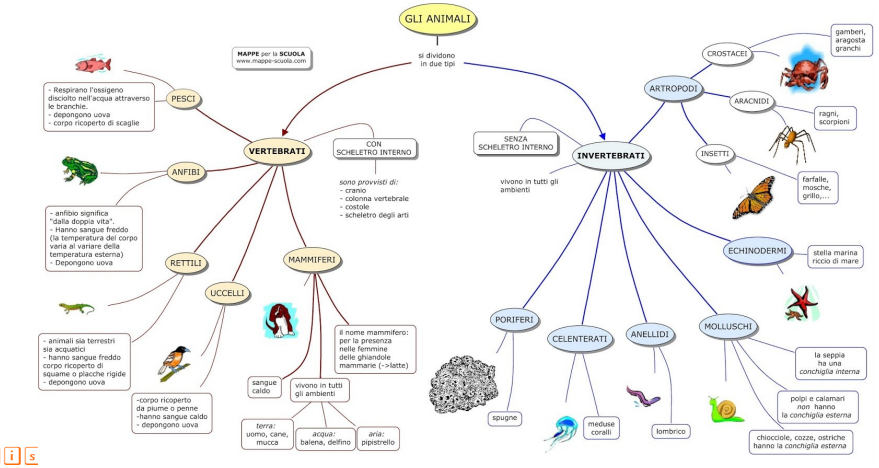
- Gli esseri umani interpretano la realtà per *categorie*
- Una parte consistente dell' apprendimento consiste nel *definire e ridefinire categorie*
- Esempi: dominio sportivo, come rappresentare che P è un pallone?
  1) Possiamo usare un **predicato Pallone(P)**:
     cattura la proprietà di P di essere un pallone
  2) **Alternativa**: reificare la categoria dei palloni e introdurre un predicato binario nuovo che restituisce vero se l' oggetto indicato appartiene alla categoria indicata
- La seconda soluzione consente di standardizzare la rappresentazione di categorie, di introdurre relazioni fra categorie e di implementare meccanismi di eredità di proprietà fra categorie

Tassonomia

- Sia **PalloneCalcio** un oggetto che rappresenta la categoria dei palloni da calcio
- **Member(P, PalloneCalcio)** è un predicato che restituisce vero se P è un elemento della categoria PalloneCalcio (in questo caso P è detto **istanza** di PalloneCalcio)
- PalloniCalcio è una sottocategoria di Palloni, si può esprimere tramite un predicato **Is-a(PalloneCalcio, Pallone)**
- I predicati **Member** ed **Is-a** consentono di organizzare la conoscenza sugli oggetti del dominio in **forma tassonomica**

Tassonomia

- Organizzazione gerarchica di categorie o concetti
- Esempio classico: tassonomia del mondo animale



Tassonomie

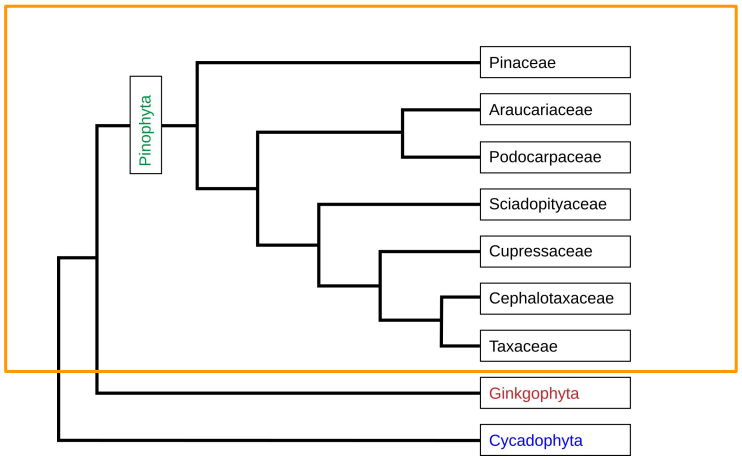
- Le relazioni come quella fra PalloneCalcio e Pallone sono dette **relazioni di sottoclasse**: tutte le istanze di una sottoclasse (PalloneCalcio) sono anche istanze della sovraclasse (Pallone)
- Una **tassonomia** è l' organizzazione delle categorie risultante da un insieme di regole di sottoclasse

Categorie, proprietà, ereditarietà

- È possibile, ed utile, caratterizzare le categorie di una tassonomia tramite la definizione di **proprietà**, esempio:
  Member(X, Pallone) ⇒ Sferico(X)
- Tramite le relazioni di sottoclasse le istanze di una classe **ereditano le proprietà delle sovraclassi**.

  Non servirà quindi scrivere:
  Member(X, PalloneCalcio) ⇒ Sferico(X)
  perché già vale a causa dell' ereditarietà

Esempio: tassonomia delle conifere



Pinophyta ≡ Conifera

Decomposizioni e disgiunzioni

- Supponiamo di avere definito le sottocategoriedi Pallone: (1) PalloneCalcio e (2) PalloneBasket
- Intuitivamente sappiamo che sono ***disgiunte*** ma per renderne consapevole il motore inferenziale occorre catturare esplicitamente questa relazione fra le due sottoclassi
- **Due categorie**:
  - **Sono disgiunte**: quando non hanno istanze comuni
  - **Costituiscono una decomposizione esaustiva**: quando tutte le istanze della sovracategoria appartengono necessariamente ad almeno una delle categorie considerate (che potranno avere anche istanze comuni)
  - **Costituiscono una partizione**: quando sono disgiunte e costituiscono una decomposizione esaustiva
- **Esempio**:
  Pinaceae, Araucariaceae, …, Taxaceae costituiscono una partizione di pinaceae

In formule

- Consideriamo una categoria C ed un insieme di categorie S = {X1, …, Xn}.
- Vengono definite le seguenti proprietà di S:
  - **S è un insieme di disgiunto di categorie**
    Disjoint(S) ⇔
    $\forall Xi,Xj \in S, Xi \neq Xj \Rightarrow Intersection(Xi, Xj) = \{\}$
  - **S è una decomposizione esaustiva di C**
    ExhaustiveDec(S, C) ⇔
    $\forall I\ (Member(I, C) \Leftrightarrow \exists Xi\ Is\text{-}a(Xi, C) \wedge Member(I, Xi))$
  - **S è una partizione di C**
    Partition(S, C) ⇔ Disjoint(S) ∧ ExhaustiveDec(S, C)

Esempio

- Sia C=Pinophyta ed S = {pinaceae, araucariaceae…, cycadophita}.
- Su S (rispetto a C) valgono tutte e tre le proprietà:
  - **Disjoint(S)**: per costruzione in una tassonomia i sottoconcetti sono disgiunti
  - **ExhaustiveDec(S, C)**: per costruzione non esistono istanze della classe esterne alle sottoclassi
  - **Partition(S, C)**: per costruzione l' insieme completo delle sottoclassi è quindi una partizione

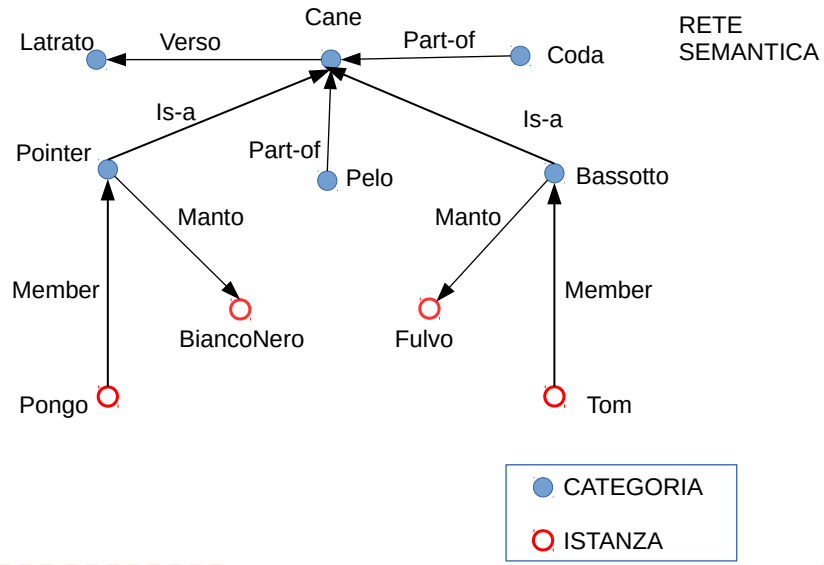
Relazioni strutturali: Part-of

- Conoscenza strutturale: dice come sono composte le cose
- Un altro aspetto naturale nella rappresentazione della conoscenza è indicare che alcuni oggetti sono parte di altri:
  - Le gambe sono parte del tavolo: **Part-of(Leg, Table)**
  - La testa è parte del corpo: **Part-of(Head, Body)**
  - La corolla è parte del fiore: **Part-of(Corolla, Flower)**
- La relazione Part-of gode della proprietà transitiva:
  - **Part-of(X, Y) ∧ Part-of(Y, Z) ⇒ Part-of(X, Z)**

Relazioni strutturali

- Le relazioni strutturali includono anche **predicati dipendenti dal dominio** (esempio: On-top, Implements, ...)
- È possibile definire formule che catturano la **struttura di tipologie di oggetti**
- **Esempio**:
  Flower(x) ⇒ ∃Corolla, Stelo Part-of(Corolla, x) ∧ Part-of(Stelo, x)
  ∧ On-top(Corolla, Stelo)

Esempio



Bunch-of

- A volte è comodo indicare che un oggetto è **composto da parti senza specificare le relazioni 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

Misure

- 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**

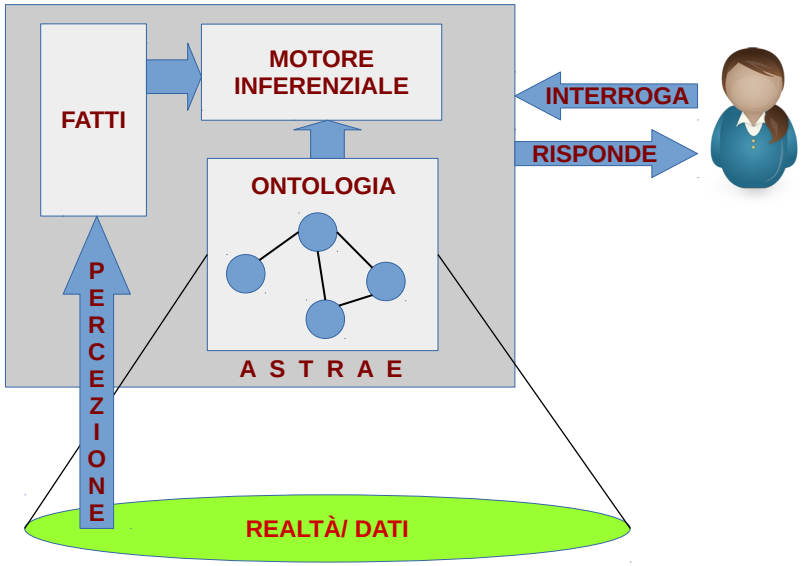
Problematiche

- 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

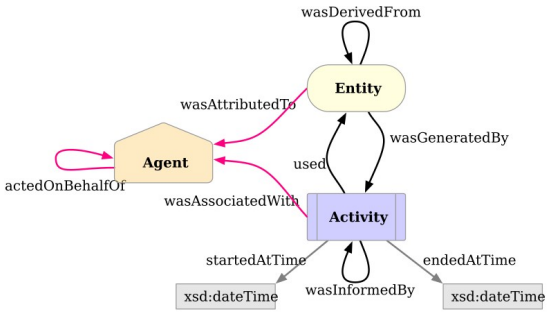


Usi delle ontologie (o reti semantiche)

- La stessa rete semantica può essere **interrogata in modi differenti**
- Alcuni tipologie di quesiti generici sono:
  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?
- 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)

Questa è l'ontologia di PROV, un linguaggio del web semantico che permette di rappresentare, integrare, tracciare informazioni relative alla 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/

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**.

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**.

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

Codifica tramite PROV

```
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"))
```


Codifica tramite PROV

```
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"))
```


Graficamente

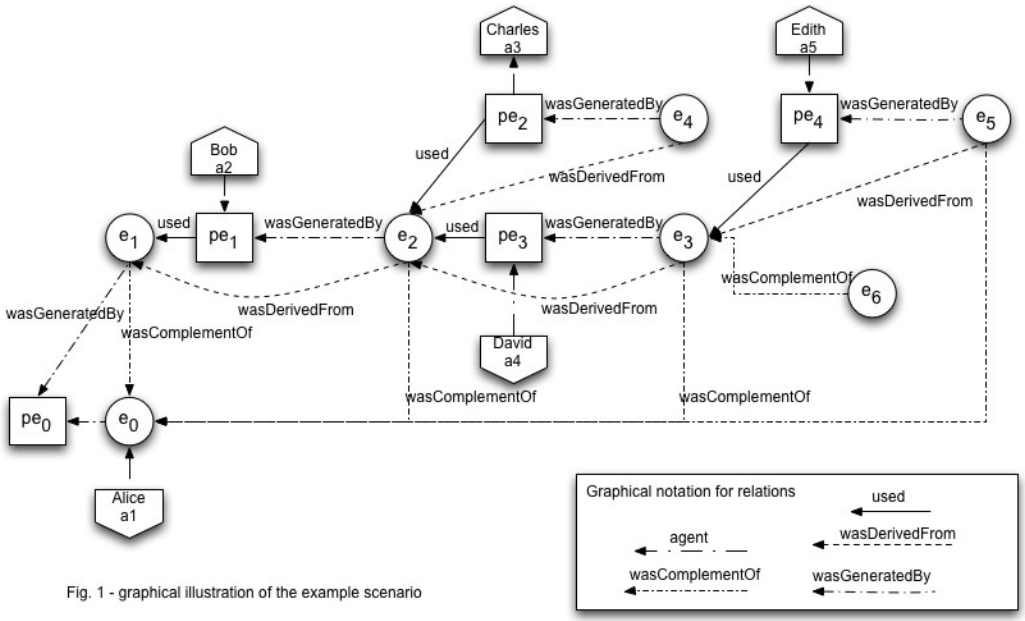
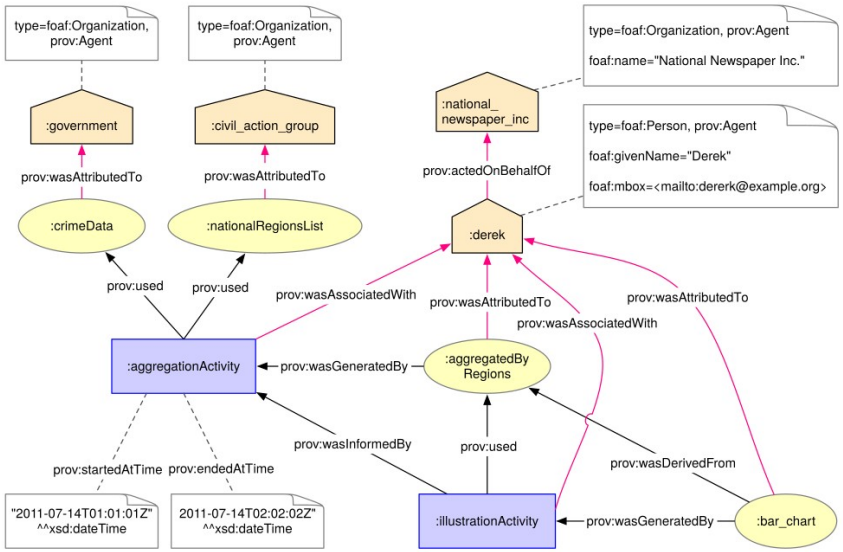


Fig. 1 - graphical illustration of the example scenario

Esempio versione più recente



Tipi di interrogazioni

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

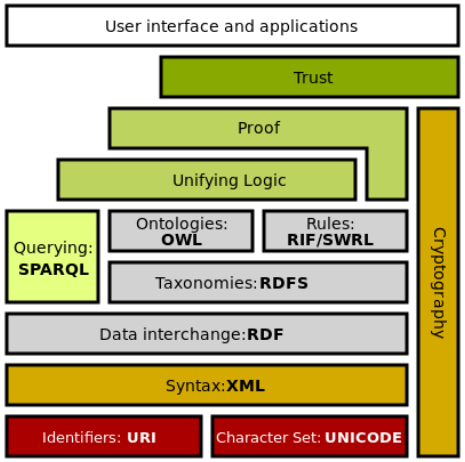
1) Su cosa si basa questa risorsa (esempio: su quali fonti)?
2) Com'è stata costruita?
3) Chi l'ha fatta?
4) Quando è stata fatta?
5) Quali nuove risorse si basano su di questa?
6) Per quali scopi è stata utilizzata questa risorsa?
7) Chi l'ha usata?
8) Quali altre risorse sono derivate dalle stesse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

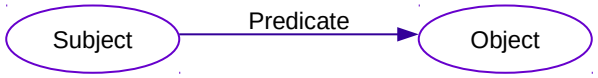
Semantic Web

- 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**



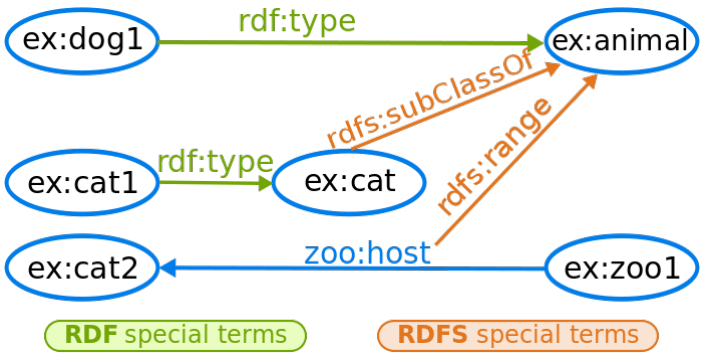
Data interchange: RDF (resource description framework)

- 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

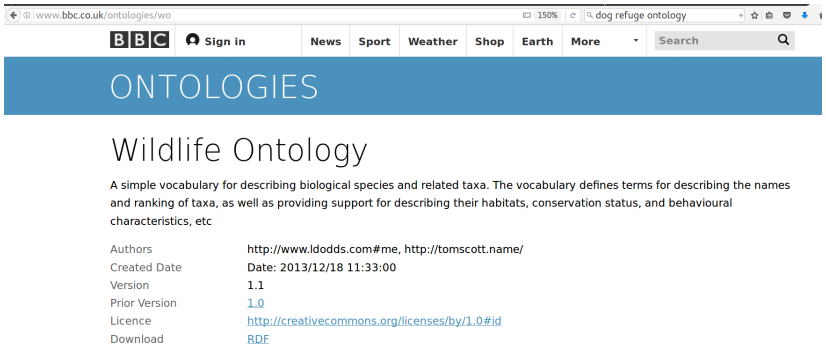
- **Identificazione delle proprietà:**
  - Elencare tutte le relazioni catturate nel DB di partenza
  - Le relazioni tipicamente sono esprimibili come verbi
  - Definire per ciascuna un' etichetta e una breve descrizione

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