Introduzione al web semantico

Il modello semantico
e gli strumenti di supporto

La gestione del web nella società dell'informazione

Il web alla base della società dell'informazione

☆ ***Web***

- Spazio di informazioni in cui l'utente umano si orienta grazie a:
  - la sua esperienza di navigazione
  - Le possibili capacità di evocazione di espressioni chiave
- La capacità espressiva di un collegamento automatico dipende dalla applicazione che lo gestisce:
  - In un motore di ricerca si inserisce una certa espressione nella convinzione che quella saprà individuare nel modo più efficace possibile il contenuto cercato.
  - ➢ l'efficacia dell'operazione dipende dagli algoritmi che il motore utilizza
  - Tra le voci di una barra di navigazione, si dovrà scegliere quale espressione si adatti meglio a individuare, come titolo generico, un contenuto.
  - ➢ l'efficacia dipende da chi ha pensato i contenuti del sito

Il web alla base della società dell'informazione

➨ *Ad ogni modo ci si affida ad una espressione unica che ha un rapporto molto generico col contenuto effettivamente ricercato*

↳ ***Risultato***

- Non sempre la Rete ci porta dove vorremmo ed è difficile orientarsi quando si è alla ricerca di qualche cosa e non sappiamo dove reperirlo
- ➢ Scorrere una lunga quantità di elenchi alla ricerca della informazione giusta è ormai la nostra abitudine e rassegnazione.

↳ Possiamo dire che la Rete è rigida

- limite importante per la diffusione di Internet anche tra coloro che faticano ad operare con il computer
- dalla sua creazione è rimasta +/- la stessa a livello tecnologico
- I veri cambiamenti tecnici che accompagneranno la creazione di un nuovo Web sono relativamente recenti e poco diffusi.
  - Anche XML è ancora relativamente conosciuto

L'ingegneria della conoscenza e il web

- *Applicazioni Web*
- *basate sulla centralità dei dati*
  - l'informazione utile risiede essenzialmente nei dati, e i processi si limitano spesso a reperire i dati necessari e a fornirne una presentazione adeguata
- *E sull'interoperabilità nei sistemi aperti*
  - Sistema aperto: sistema scoperto e utilizzato da un'applicazione remota senza bisogno dell'intervento umano.
  - L'interoperabilità in un ambiente aperto richiede che le applicazioni possano accedere a un *repertorio di conoscenze comuni* e siano in grado di sfruttare tali conoscenze in modo autonomo

L'ingegneria della conoscenza e il web

↳ ***Web:***

✓ essenzialmente un colossale archivio di dati interrogato quotidianamente da milioni di utenti

➨ *Il problema sul tappeto è creare un'infrastruttura che ci aiuti a risolvere i problemi più in fretta: il web è pieno di informazioni che spesso non possono essere utilizzate in modo efficiente. (Berners-Lee)*

L'ingegneria della conoscenza e il web

☆ ***Web semantico***

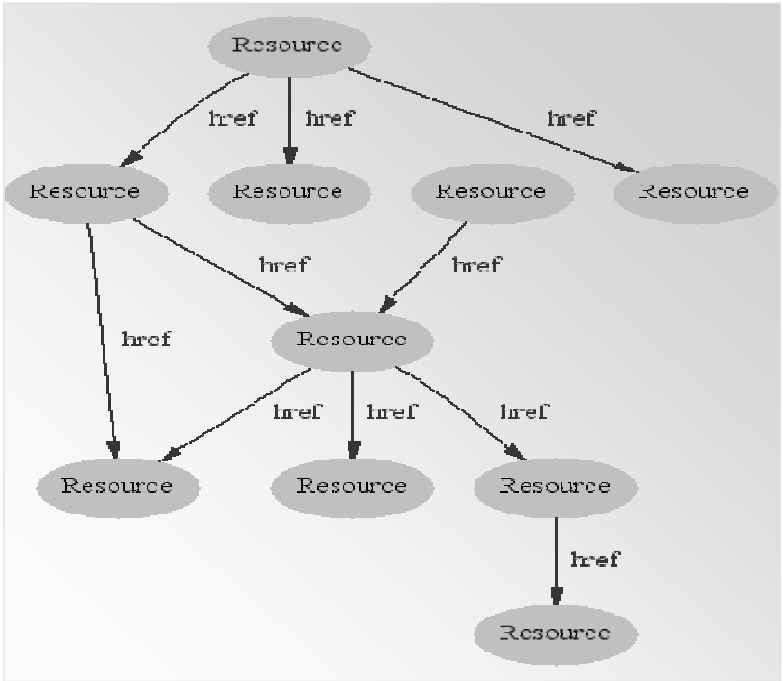
➨ L'idea di fondo del web semantico è quella di far diventare la Rete in grado di capire le nostre richieste.

➨ E per far questo, all'interno dell'architettura della Rete, dalle semplici pagine web fino ai database più complessi, dovranno esserci elementi in grado di consentire a determinati agenti informatici una certa capacità d'azione.

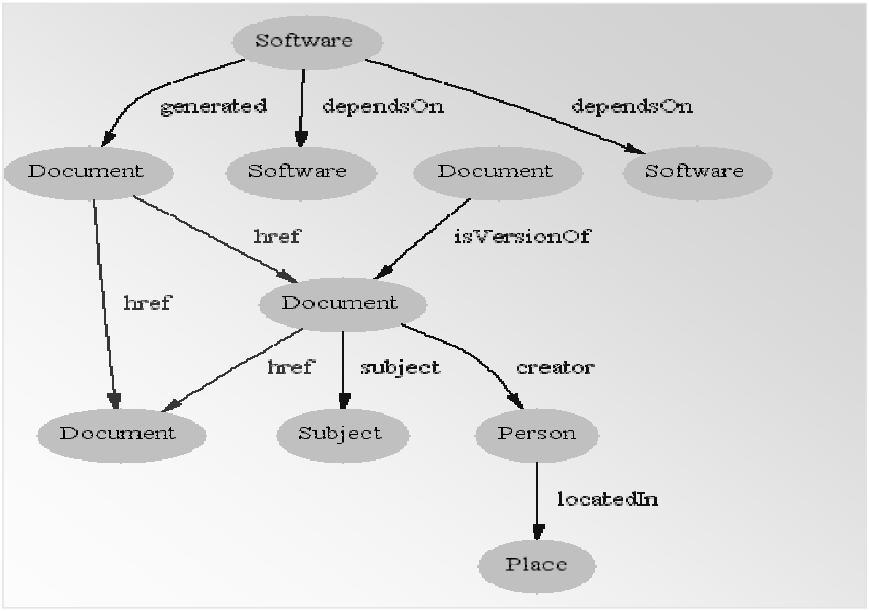
☆ 2001 Berners-Lee :*"Il Web Semantico è un'estensione del Web di oggi, in cui all'informazione viene dato un preciso significato, permettendo ai computer ed alle persone di lavorare in cooperazione."*

L'organizzazione delle pagine web attualmente

collegamenti sintattici => legati al funzionamento di un qualche codice => deboli



Organizzazione delle pagine con collegamenti semantici



↳Oltre a portare in un determinato luogo un collegamento dovrebbe descrivere il luogo verso cui porta.

Semantico è un meccanismo che sa predire il valore della sua azione

L'ingegneria della conoscenza e il web

- *Collegamenti Semantici: Non è un concetto nuovo:*
- Quando si interroga una base di dati si possono eseguire ricerche imponendo precise relazioni:
  - Es. In una biblioteca on-line si può chiedere "quali autori hanno scritto almeno due libri di "Intelligenza artificiale"
  - Le relazioni sono stabilite fra *concetti* ("autore" e "libro") anziché fra parole chiave (non si ricerca la stringa "autore" o "libro").
- Questo è possibile perché esiste uno *schema* del DB, cioè un modello ed un insieme di regole che stabiliscono come devono essere organizzati i dati

Il web semantico …

- *Non è Intelligenza Artificiale*
  - Il concetto di documenti riconoscibili dalle macchine indica solo una capacità delle macchine a risolvere un problema ben definito con operazioni ben definite su dati ben definiti ed esistenti.
  - Non richiede alle macchine di riconoscere e capire il linguaggio delle persone, ma richiede alle persone di andare nella loro direzione.
- Non richiederà che ogni applicazione debba usare espressioni di complessità arbitraria
  - Sebbene il linguaggio lo permetta le applicazioni saranno limitate in pratica a generare semplici espressioni come un elenco di controllo di accesso, criteri di ricerca. ..
- Non è un nuovo tentativo di un precedente esperimento già fallito
  - Molti sistemi di rappresentazione della conoscenza hanno un problema nell'unire due basi separate di conoscenza dove ogni concetto aveva un solo posto nell'albero di conoscenza.
  - Il mondo del Web semantico è invece disegnato avendo in mente proprio questo: la documentazione retrospettiva di relazioni fra concetti originalmente indipendenti.

I modelli semantici

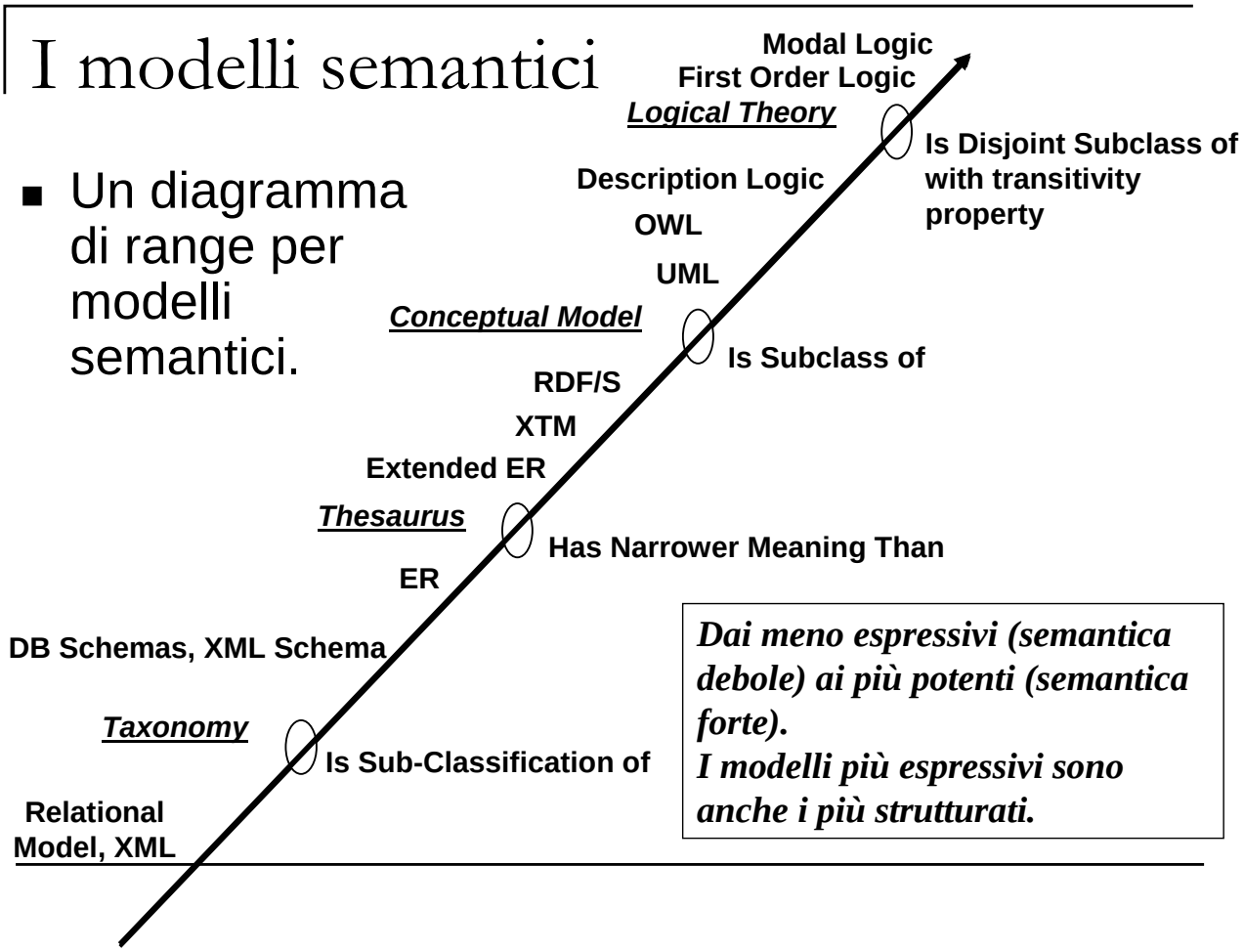
I modelli semantici

☆Problema attuale: cercare una *modellizzazione adeguata per la conoscenza in un domino specifico o per descrivere le relazioni entro una organizzazione*

- Il problema è accentuato se si vogliono ottenere degli approcci utilizzabili anche dalle macchine
- Problemi di inconsistenza terminologica e l'uso diversi vocabolari e modelli sono all'ordine del giorno
- L'identificazione e la riconciliazione di queste distinzioni semantiche è una ragione fondamentale per usare i modelli semantici.

I modelli semantici

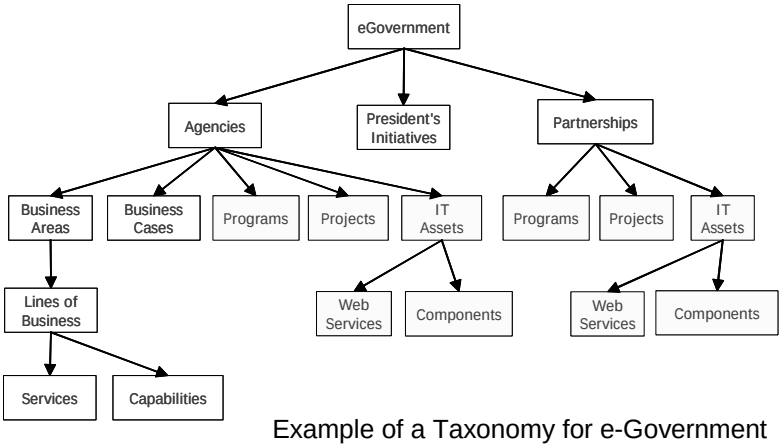
- Un diagramma di range per modelli semantici.



Dai meno espressivi (semantica debole) ai più potenti (semantica forte).
I modelli più espressivi sono anche i più strutturati.

Tassonomia

- Uno dei modelli semantici più semplici
- Cattura il fatto che esistono delle connessioni fra i termini ma non la loro natura



Example of a Taxonomy for e-Government

- Classifica le informazioni entro una ben definita struttura associativa. Normalmente queste classificazioni sono in forma di gerarchie.
- Una gerarchia è una struttura ad albero.
- Come un comune albero ha una radice, e delle biforcazioni. Ogni punto di biforcazione è chiamato nodo.

↳ Serve un linguaggio di indicizzazione (vocabolario controllato)

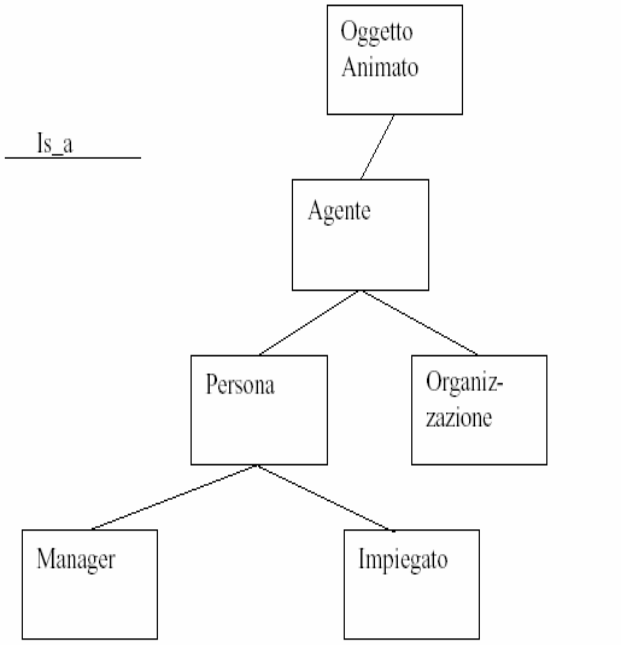
Tassonomia

- Una definizione da vocabolario di tassonomia suonerebbe circa così: (from Merriam-Webster OnLine http://www.m-w.com):
- *Lo studio dei principi generali di classificazione scientifica: classificazione sistematica. In particolare: classificazione ordinata di piante ed animali in accordo con le loro presunte relazioni naturali.*

- Riportata alle tecnologie dell'informazione una tassonomia potrebbe essere definita come:
- *La classificazione di entità di informazione in forma di gerarchia in accordo con le relazioni presunte tra le entità del mondo reale che esse rappresentano.*

Tassonomia

- **Esempio**
- Sono presenti le la classi *Oggetto Animato*, *Agente*, *Persona*, *Organizzatore*, *Manager* e *Impiegato*.
- La freccia indica la relazionew *Is_a* o *é_sottoclasse_di*.
- Così *Persona* è sottoclasse di *Agente*. *Impiegato* è sottoclasse di *Persona*.
- Si può anche dire che *Persona* è superclasse di *Impiegato* e *Agente* è superclasse di *Persona*.



Tassonomia

- Quando cerchiamo qualcosa dobbiamo sapere dove cercarlo (cioè sapere dove l'argomento è stato classificato).
  - Potremmo cercare i dinosauri ipotizzando che siano stati classificati nella categoria Vertebrati fossili.
- Alcuni motori di ricerca usano tassonomie per rintracciare i documenti al posto di combinazioni booleane di parole chiave.

Tassonomie e Tesauri

- Se nelle tassonomie abbiamo solo la relazione di sottoclasse.
- Nei *tesauri* abbiamo dei vocabolari controllati di termini fra cui possono esistere più relazioni.
  - Indica l'esigenza di trovare un punto di incontro tra lessico dell'autore e del ricercatore, una relazione biunivoca tra termine e concetto, così da ottenere *univocità semantica*.
  - Si eliminano i problemi connessi con l'uso del linguaggio naturale, in cui ridondanze, ambiguità, omonimie ed altre caratteristiche che ne garantiscono espressività, rendono difficile l'organizzazione funzionale dei motori di ricerca.

Tesauro

Definizione di tesauro (ISO 2788-1986) «*il thesaurus è il vocabolario di un "linguaggio di indicizzazione" controllato, organizzato in maniera formale, in maniera cioè da rendere esplicite le relazioni "a priori" fra i concetti*»

- Un esempio noto è WordNet (http://www.cogsci.princeton.edu/~wn). Si può anche usare in linea. (http://www.cogsci.princeton.edu/cgi-bin/webwn).
- Ogni parola di WordNet puo avere uno o più significati (word senses o ***SynSets***).
- Il fatto che una parola possa avere più significati viene chiamata polisemia.

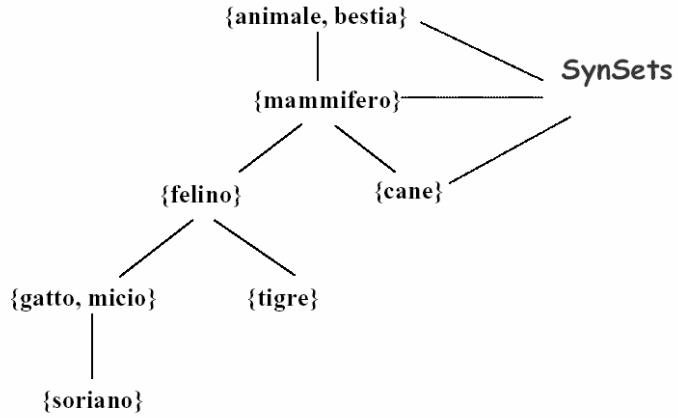
Tesauri

WordNet

Da un elenco di parole:

<tigre, cane, animale, mammifero, bestia, micio, soriano, gatto, felino>

A un dizionario strutturato:



La stessa parola può appartenere a più SynSets

Tesauri

- I SynSets sono legati da alcune relazioni
  - Sinonimia: lega i nodi che hanno lo stesso significato.
  - Iperonimia: Nodi che hanno un significato più generale
  - Iponimia: Synsets che hanno un significato più ristretto
  - Olonimia: è parte di
  - Meronimia: ha come parti

Tesauri

- Un esempio di uso di WordNet.
- Cerchiamo la parola bank che secondo WordNet ha 10 significati come nome e 8 come verbo (non riportati).

WordNet 2.0 Search

Search word: [ ] Find senses

Overview for "bank"

The **noun** "bank" has 10 senses in WordNet.

1. depository financial institution, **bank**, banking concern, banking company -- (a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home")
2. **bank** -- (sloping land (especially the slope beside a body of water); "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents")
3. **bank** -- (a supply or stock held in reserve for future use (especially in emergencies))
4. **bank**, bank building -- (a building in which commercial banking is transacted; "the bank is on the corner of Nassau and Witherspoon")
5. **bank** -- (an arrangement of similar objects in a row or in tiers; "he operated a bank of switches")
6. savings bank, coin bank, money box, **bank** -- (a container (usually with a slot in the top) for keeping money at home; "the coin bank was empty")
7. **bank** -- (a long ridge or pile; "a huge bank of earth")
8. **bank** -- (the funds held by a gambling house or the dealer in some gambling games; "he tried to break the bank at Monte Carlo")
9. **bank**, cant, camber -- (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
10. **bank** -- (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning); "the plane went into a steep bank")

Tesauri

- *Un esempio d'uso di WordNet.*
- *Cerchiamo degli iperonimi del senso 1*

Results for "Hypernyms (this is a kind of...)" search of noun "bank"

Sense 1
depository financial institution, bank, banking concern, banking company --
(a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home")
=> financial institution, financial organization, financial organisation --
(an institution (public or private) that collects funds (from the public or other institutions) and invests them in financial assets)
=> institution, establishment -- (an organization founded and united for a specific purpose)
=> organization, organisation -- (a group of people who work together)
=> social group -- (people sharing some social relation)
=> group, grouping -- (any number of entities (members) considered as a unit)

Da tassonomie e tesauri alle Ontologie

- Tassonomie e tesauri fissano una semantica.
- *Per arricchire la semantica si deve passare a modelli concettuali e teorie logiche.*
- Un ***modello concettuale*** è il modello di una particolare area di conoscenza o di attività, chiamata dominio, che rappresenta le entità del dominio, le relazioni che intercorrono fra queste, espresse sotto forma di attributi (proprietà) delle entità e dei valori che questi attributi possono avere.
- Inoltre possiamo dare delle regole che riguardano le classi, gli attributi e le loro relazioni.

Ontologie

- Un modello concettuale è simile a quello che si usa nella programmazione orientata agli oggetti dove possiamo definire classi e sottoclassi.
- Anche le teorie logiche possono essere impiegate per descrivere modelli concettuali.
  - In questo caso si hanno assiomi e regole di inferenza.
  - Da questi si possono dimostrare dei teoremi per far vedere che qualcosa è vero.
  - Per esempio si può dire che la relazione *è_sottoclasse_di* è transitiva.
  - Che la relazione *è_superclasse_di* è la sua inversa,
  - che due sottoclassi di un medesimo concetto sono disgiunte (cioè non hanno elementi in comune).
  - ↳ La logica che si occupa di questo viene chiamata "***logica descrittiva".***

Logiche descrittive

- La sintassi della logica dei predicati è progettata per rendere facile parlare degli oggetti
- Le logiche descrittive sono notazioni progettate per facilitare la descrizione della definizione e delle proprietà delle categorie.
- I sistemi basati sulla logica descrittiva si sono evoluti dalle reti semantiche per rispondere alla necessità di formalizzare il significato delle reti, mantenendo l'enfasi sulla struttura tassonomica come principio organizzativo
- Si parla di logiche descrititve già negli anni '80

Logiche descrittive vs logiche dei predicati

- In logica dei predicati si possono esprimere conoscenze molto articolate e, almeno in linea di principio, eseguire in modo automatico ragionamenti complessi.
- Ci sono però due problemi.
  - la procedura di deduzione (*procedura di prova* o *calcolo*) non è una procedura di decisione, ma soltanto di *semidecisione*. Cioè:
  - Se la conclusione è deducibile delle premesse, la procedura termina in un numero finito di passi producendo una prova;
  - se la conclusione non è deducibile delle premesse, la procedura può non terminare (un sistema informatico può "andare in ciclo" quindi)
  - Il secondo problema è che la procedura, anche nei casi in cui termina, è spesso molto costosa in termini di risorse di calcolo.

Logiche descrittive vs logiche dei predicati

- In logica descrittiva si è cercato un linguaggio meno espressivo ma in grado di:
  - essere comunque abbastanza espressivo per le applicazioni;
  - la deduzione sia basata su una procedura di decisione (che quindi termina in ogni caso dopo un numero finito di passi, sia quando la conclusione è deducibile dalle premesse, sia quando non lo è);
  - la procedura di deduzione abbia complessità computazionale accettabile (ovvero richieda una quantità accettabile di risorse di calcolo).

Logiche descrittive

- Termini: espressioni che descrivono concetti
  - Atomici: uomo
    - Sono detti concetti (descrivono concetti) o classi (denotano insiemi di oggetti della realtà)
  - Complessi: persona $\cap$ maschio
  - Equivalenza terminologica: uomo$\equiv$ persona $\cap$ maschio
    - Nel caso uno dei due termini è complesso e l'altro atomico si dice *definizione terminologica*
- *Terminologia* od *ontologia* un insieme finito di definizioni terminologiche.

Logiche descrittive

- Ontologia: assegna un *significato* non ambiguo ai termini atomici, *in base al significato (in funzione) di altri termini. I quali i*n generale avranno un significato grazie a ulteriori definizioni terminologiche, e così via.
- Ogni ontologia è finita: prima o poi si arriva a termini privi di una definizione, che vanno considerati come *primitivi*.
- Termini primitivi: affinché l'ontologia abbia una qualche utilità pratica questi devono essere ancorati a, e definiti in, qualche dominio (problema del ***symbol grounding***)

Logica descrittiva

- Semantica dei termini
- Per le espressioni di una DL è possibile specificare una *semantica formale*, che associa a ogni termine una interpretazione definita in modo insiemistico.
- Termini
  - Data una certa porzione predefinita della realtà (che chiameremo *dominio*), ogni termine determina un'*estensione*, definita come l'insieme di tutti gli individui del dominio cui il termine si applica.
    - Es: il dominio comprende tutti gli individui presenti in un'aula durante una lezione, l'estensione in quel dominio del termine UOMO è costituito da tutte gli uomini presenti nell'aula
- Termini predefiniti
  - T Concetto universale (totalità degli individui esistenti)
  - $\perp$ Concetto vuoto (insieme vuoto di individui)

Logica descrittiva

- Operatori
- Intersezione '$\cap$'
- Equivalenza '$\equiv$'

$$DONNA \equiv PERSONA \cap FEMMINA$$

- Complemento '$\neg$' $UOMO \equiv PERSONA \cap \neg FEMMINA$,
- Unione '$\cup$',

$$NATURALE \equiv MINERALE \cup (VEGETALE \cup ANIMALE)$$

Logica descrittiva

- Termini basati su ruoli
  - Oltre ai termini corrispondenti a predicati con un argomento (detti concetti o classi), le DL utilizzano termini corrispondenti a predicati a due argomenti, che esprimono relazioni binarie fra individui della realtà; tali termini vengono detti *ruoli*, *proprietà*, *attributi* o *relazioni*.
  - Es: definizione terminologica che utilizza un ruolo

    GENITORE ≡ ∃Figlio, (in FOL -> ∃y Figlio(x,y))
  - che intuitivamente significa "genitore è chi ha un figlio".
  - L'espressione ∃Figlio è un termine complesso, formato dal *quantificatore esistenziale* '∃' e dal *ruolo* Figlio.

Logica descrittiva

- Termini basati su ruoli
- Le espressioni come $\exists R$ possono essere combinate come concetti:
- $MADRE \equiv DONNA \cap \exists Figlio$
- $MADRE\text{-}F \equiv MADRE \cap \forall Figlio.FEMMINA$
  - Madre con sole figlie femmine
- $GENITORE\text{-}UN\text{-}M \equiv \exists Figlio.\neg FEMMINA$
  - Genitore che ha almeno un figlio maschio

Logiche Descrittive

- ***Principali metodi inferenziali***
- Sussunzione
  - Consiste nel determinare se una categoria è il sottoinsieme di un'altra confrontando le loro definizioni
- Classificazione
  - Consiste nella verifica che un oggetto appartiene ad una categoria
- Verifica di consistenza di una definizione di categoria
  - Una definizione è consistente se i criteri di appartenenza sono logicamente soddisfacibili

Logica descrittiva

- Sussunzione

- Supponiamo di voler specificare che i gatti sono animali domestici.
- Non è possibile utilizzare un'equivalenza, perché naturalmente esistono animali domestici che non sono gatti.
- La relazione fra i due termini si esprime allora nel modo seguente:

  GATTO $\subseteq$ ANIM-DOM.
- L'operatore '$\subseteq$' è detto operatore di *sussunzione*;
  - il termine GATTO è *sussunto* dal (è un *iponimo* del, è una *specializzazione* del) termine ANIM-DOM;
  - il termine ANIM-DOM *sussume* il (è un *iperonimo* del, è una *generalizzazione* del) termine GATTO.

Logica descrittiva

- Le conoscenze fattuali in DL

- Nelle DL si possono esprimere due tipi di conoscenze fattuali. Sono infatti ammesse asserzioni del tipo
  - $C(a)$,
    - dove C è un termine arbitrario e a è un nominale (nomi d'individuo)
  - $R(a,b)$,
    - dove R è un ruolo ed a, b sono nominali (non necessariamente distinti).
- Esempi:
  - MADRE(anna)
  - DONNA $\cap$ $\exists$Figlio(anna)(dove C = DONNA $\cap$ $\exists$Figlio)
  - Figlio(anna,bruno)

Logiche Descrittive

Esempi

- Scapoli sono maschi adulti non sposati
- *Scapolo=And(NonSposato,Adulto,Maschio)*
- In logica dei predicati
- *Scapolo(x) ⇔ NonSposato(x) & Adulto(x) & Maschio(x)*
- *La logica descrittiva permette di applicare direttamente operazioni logiche ai predicati*
- *L'insieme degli uomini con almeno tre figli tutti disoccupati e sposati con dottori e con al più due figlie tutte professoresse di fisica o di matematica*
- *And(Uomo,AtLeast(3,Figlio),AtMost(2,Figlia), All(Figlio,And(disoccupato,Sposato,All(Coniuge,Dottore))), All(Figlia,And(Professore,Fills(Dipartimento,Fisica,Matematica))))*

Logiche Descrittive: *discussione*

- Enfasi sulla trattabilità dell'inferenza
  - Un'istanza di problema è risolta descrivendola e poi chiedendo al sistema se è sussunta tra molte possibili categorie di soluzioni
  - Nei sistemi basati sulla logica dei predicati, predire il tempo richiesto è quasi sempre impossibile
  - In logica descrittiva si cerca di assicurare che la verifica di sussunzione sia effettuata in un tempo che cresce con un polinomio della dimensione delle descrizioni

Logiche Descrittive: *discussione*

- Conseguenze
  - Problemi difficili non potranno essere espressi
  - O le loro descrizioni dovranno essere esponenzialmente grandi
  - I risultati aiutano comunque l'utente a riconoscere i costrutti che causano problemi

Logiche Descrittive: *discussione*

- *La logica descrittiva permette di applicare direttamente operazioni logiche ai predicati*
  - ✓ dato un termine atomico A, la semantica del termine ci deve consentire di identificare l'estensione di A nel dominio; tale estensione è rappresentata da tutti e soli gli individui del dominio che rendono vera la formula A(x) quando siano assunti come valori della x.

Specifichiamo meglio le Ontologie

- Una definizione di ontologia:
  Un'ontologia è una *descrizione formale esplicita di un dominio* di interesse.
  - ***Descrizione***: una forma di rappresentazione della conoscenza
  - ***Formale***: simbolica e meccanizzabile
  - ***Esplicita:*** elenchi estensionali di frammenti di conoscenza
  - ***Dominio***: ristretta ad un determinato sottoinsieme dello scibile, affrontato da un certo punto di vista.

Ontologie

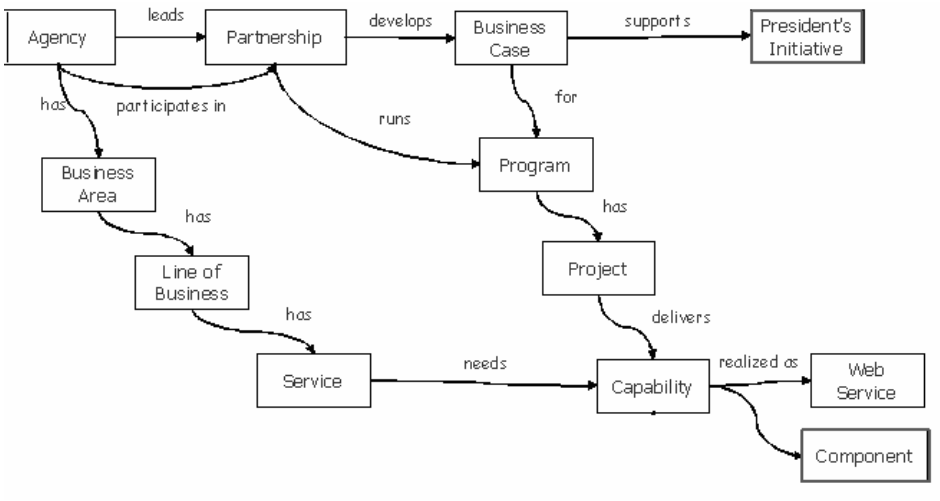
Quindi una ontologia è costituita da:

- Classi (concetti generali del dominio di interesse.)
- Relazioni semantiche tra queste classi
- Proprietà (attributi, ruoli) assegnate a ciascun concetto, che ne descrivono vari tipi di attributi o proprieta'
  - Restrizioni sulle proprietà Impongono il tipo di dato sul valore che la proprietà può assumere
- un eventuale *livello logico* che permetta di inferire nuovi fatti a partire da quelli codificati all'interno della risorsa
  - (ad esempio, un insieme di *assiomi* o *micro-teorie*).

Ontologie

- Se una tassonomia è schematizzata come un albero un'ontologia è schematizzabile come un grafo

- La flessibilità del modello e le possibilità di connessioni la rendono molto espressiva, ma bisogna saper trattare le contraddizioni ed arrivare a compromessi



Ontologie e tassonomie

- Tassonomia è la forma più semplice che un'ontologia può assumere che consiste nella semplice classificazione *gerarchica* delle entità di un campo applicativo.
- Definendo *partizione* la definizione di una classe come unione di un numero finito di sottoclassi disgiunte fra loro, ovvero:

  $A \equiv B1 \cup ... \cup Bn$,

  $Bi \cap Bj \equiv \perp$ per $1 \leq i < j \leq n$.
- Una tassonomia è una insieme di partizioni che inizia da un'unica classe, detto *radice* della tassonomia, e prosegue poi per livelli.

Ontologie e tesauri

- I *tesauri* sono dizionari che descrivono relazioni semantiche fra le parole di una lingua.
- Tutte queste relazioni possono essere dedotte da un'ontologia, pur di associare alle parole i concetti che ad esse corrispondono.
- Più precisamente, assumiamo che a ogni parola sia associato un concetto, e supponiamo che alle due parole $p1$ e $p2$ corrispondano rispettivamente i concetti $C1$ e $C2$; in tal caso:
  - se $C1 \equiv C2$, allora $p1$ e $p2$ si dicono *sinòmini* (ad es., "gatto" e "micio");
  - se $C1 \subseteq C2$, allora $p1$ si dice *ipònimo* di $p2$ e $p2$ si dice *iperònimo* di $p1$ (ad es., "gatto" e "felino");
  - se $C1 \equiv C \cap D$ e $C2 \equiv C \cap \neg D$, allora $p1$ e $p2$ si dicono *antònimi* (ad es., "femmina" e "maschio");
  - $C1 \subseteq \exists$Parte.$C2$, allora $p1$ si dice *olònimo* di $p2$ e $p2$ si dice *meròmimo* di $p1$ (ad es., "padella" e "manico").

Ontologie e il Web

- Una ontologia descrive le parole comuni e i concetti (significati) usati per descrivere e rappresentare un'area di conoscenza (dominio).
- Una ontologia può essere usata da persone, applicazioni, database etc. per condividere una conoscenza comune riguardo ad un certo dominio (educazione, medicina, riparazione di automobili etc.).
- L'ontologia include le definizioni dei concetti del dominio e delle loro relazioni in un modo usabile dal computer (ma anche comprensibile agli umani).

Ontologie e il web

- Si possono distinguere tre livelli di generalità per le ontologie:
  - ☆ ontologia superiore (upper ontology),
    - i termini più astratti e di uso praticamente universale e indipendente dalla singola area applicativa (es. termini come "oggetto fisico", "prodotto" "evento", "azione" …);
  - ☆ *ontologia mediana* (*middle ontology*),
    - un'area applicativa ben definita (es, la chimica organica, il commercio elettronico, la telefonia mobile, …);
  - ☆ ontologia inferiore (lower ontology),
    - utilizzata per interfacciare una singola applicazione con un'ontologia mediana (es, un servizio web per la vendita on line di strumenti musicali dovrà combinare in modo specifico un'ontologia del commercio elettronico e un'ontologia degli strumenti musicali).

Ontologie e il web

- In un'applicazione, la maggior parte dei termini tende ad essere definita nel livello mediano, i livelli superiore e inferiore riguardano un numero di termini più limitato.
- Naturalmente, la maggior parte delle ontologie tende a presentare nel livello superiore gli stessi termini:
  - "oggetto fisico" ed "evento" compariranno nell'*upper ontology* di molte applicazioni
- L'ideale, sarebbe definire un'ontologia superiore standard, da utilizzare come punto di partenza per lo sviluppo di ontologie mediane e inferiori.
  - Attualmente sono in corso vari tentativi:
    - la *Standard Upper Ontology* (SUO, http://suo.ieee.org);
    - *Open Cyc* (http://www.opencyc.org), derivata dall'ontologia Cyc.

Ontologie e Knowledge Base

↳Istanze

- A partire dalle classi dell'ontologia, è possibile definire delle istanze, che rappresentano specifici oggetti del mondo reale
- Le istanze ereditano attributi e relazioni dalle classi

☆Knowledge base = Ontologia + insieme delle istanze delle classi

Ontologie

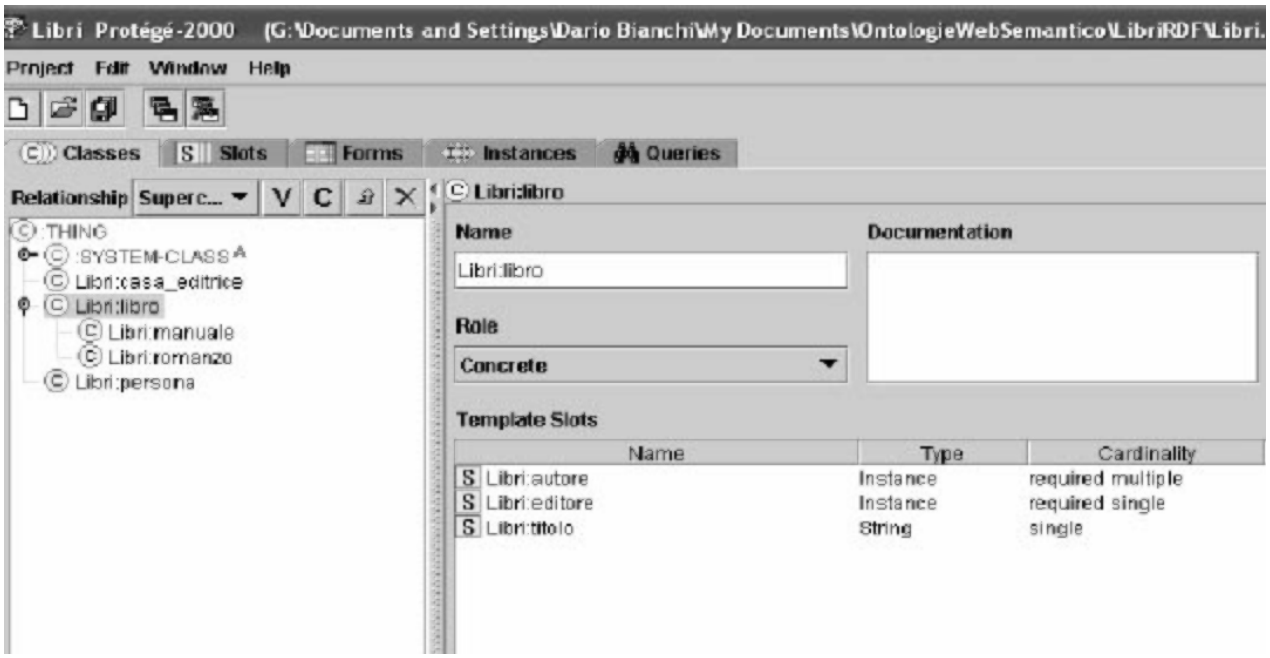
↳Un esempio di ontologia

- Si vogliono descrivere i libri, le case editrici e le persone
- l'esempio è illustrato con riferimento al tool per la costruzione di ontologie Protege http://protege.stanford.edu:

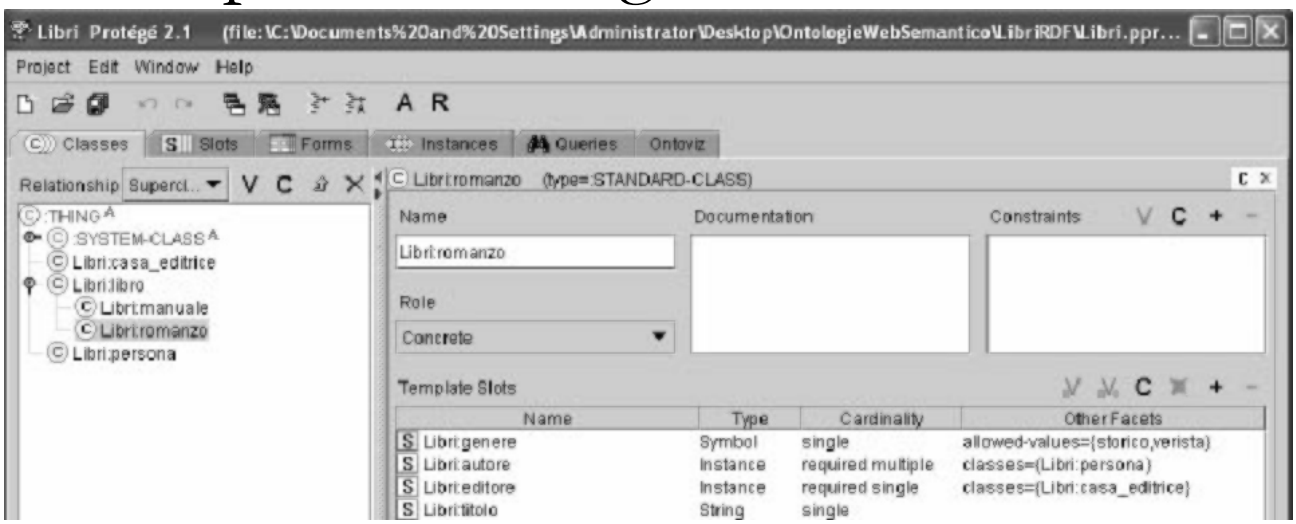
Esempio di ontologia

- La classe libri ha due sottoclassi manuale e romanzo.
- La classe libro ha tre slot autore, editore e titolo.
- Ci sono delle restrizioni sugli slot:
  - Autore deve essere una istanza di persona (una assunzione semplificativa)
  - deve esserci almeno 1 autore, ma anche più di uno (required multiple in protege).
  - Editore ha come valore una istanza obbligatoria e unica di casa editrice.
  - Titolo è un tipo base cioè stringa ed è obbligatorio ed unico.

Esempio di ontologie



Esempio di ontologia

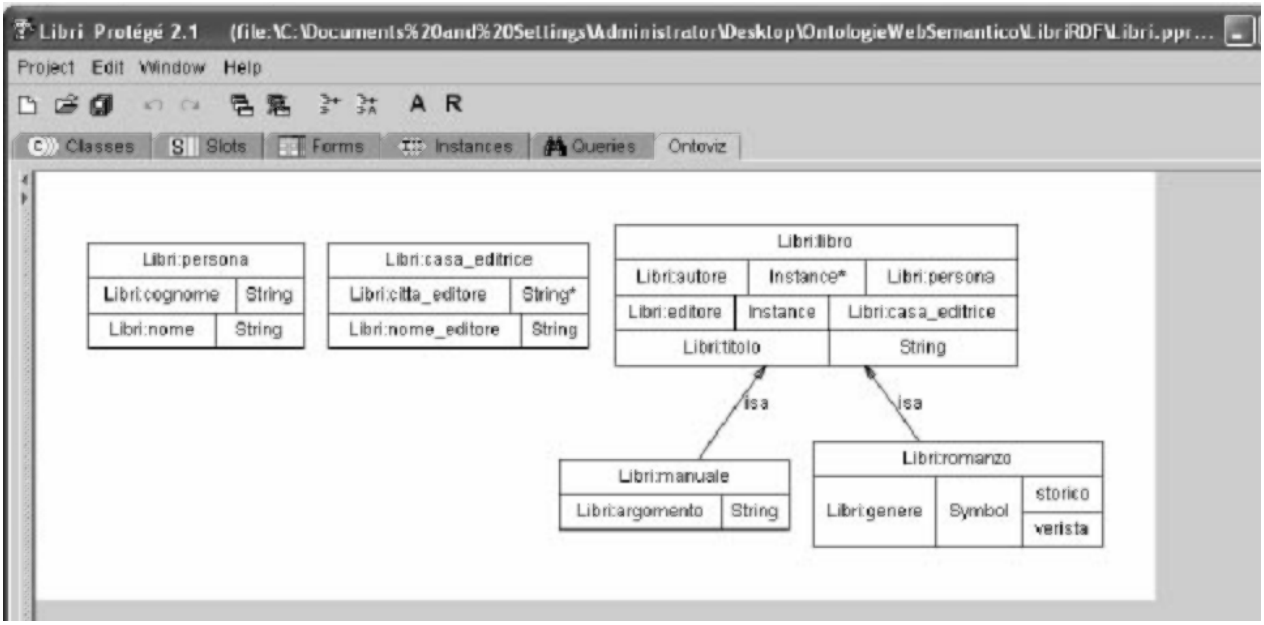


- Le sottoclassi manuale e romanzo ereditano gli stessi slot ma possono aggiungerne altri:
  - La classe romanzo eredita questi tre slot ma ne aggiunge uno, genere di tipo simbolo che può assumere solo i valori elencati (storico, verista ..) e di numerosità single (cioè solo 1).

Esempio di Ontologia

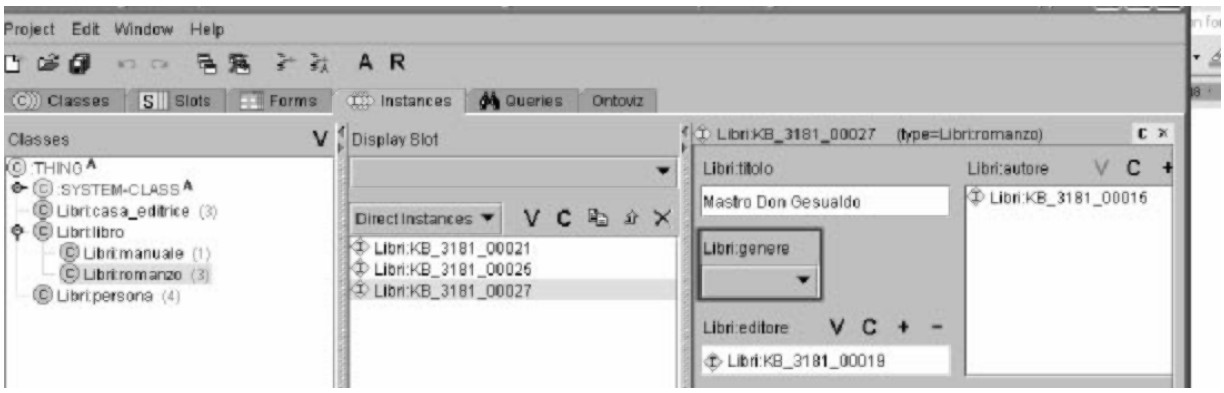
- In maniera analoga possiamo definire le classi persona e casa_editrice.
- Quella che segue è una immagine della struttura delle classi ottenuta con ontowiz (un plugin di Protege).
  - Sotto il nome della classe (concetto) si vedono gli slot (proprietà) ed i valori che possono assumere.

Esempio di ntologia



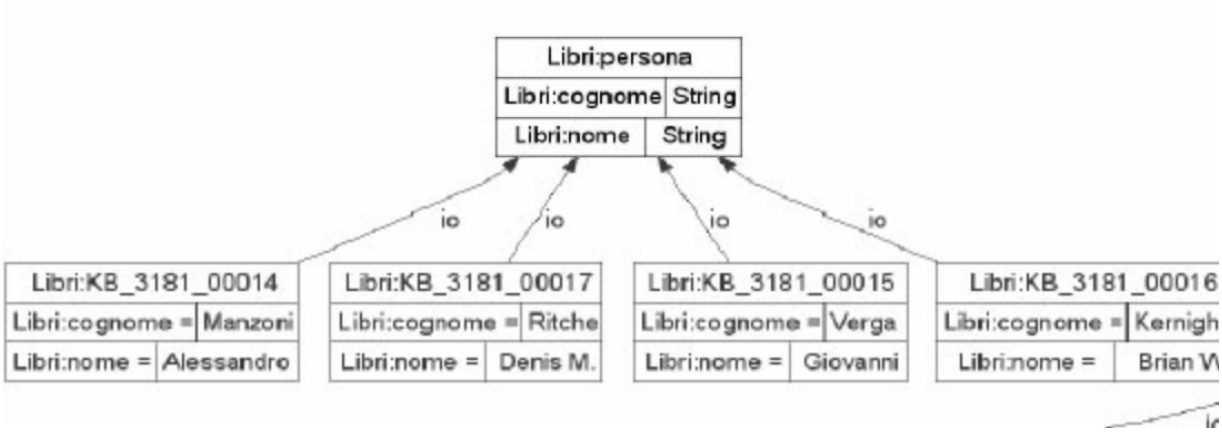
Esempio di ontologia e Knowledge Base

Definita l'ontologia possiamo passare a definire le istanze delle varie classi.
In questo esempio vediamo le istanze di romanzo.



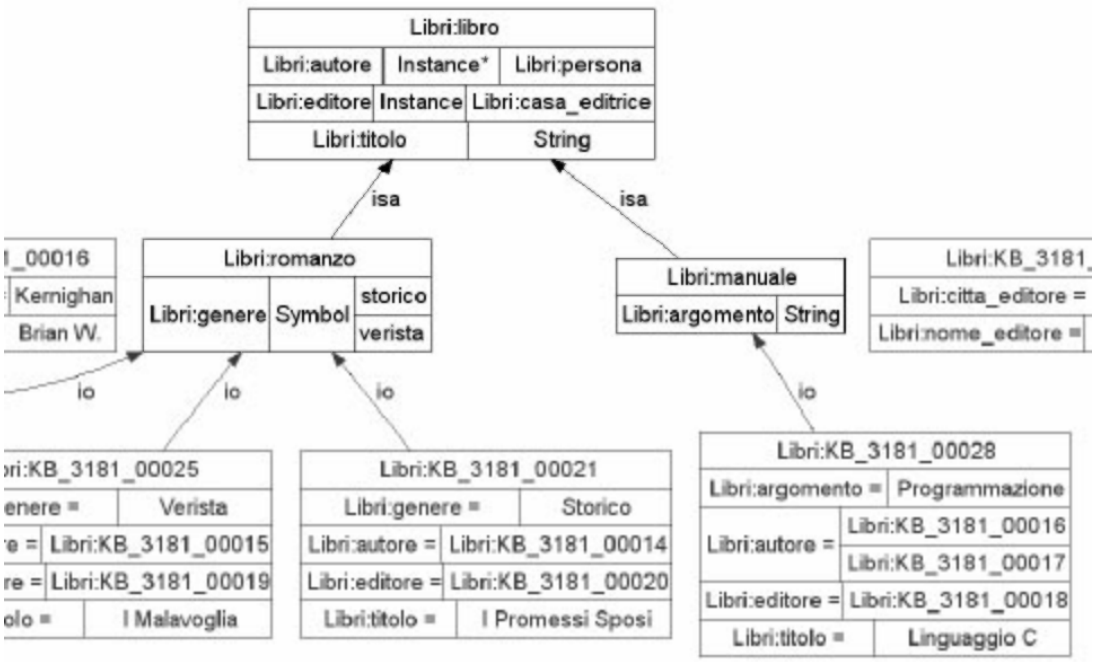
Esempio di ontologia e Knowledge Base

Possiamo vedere le istanze in modo grafico .



Le istanze di persona.

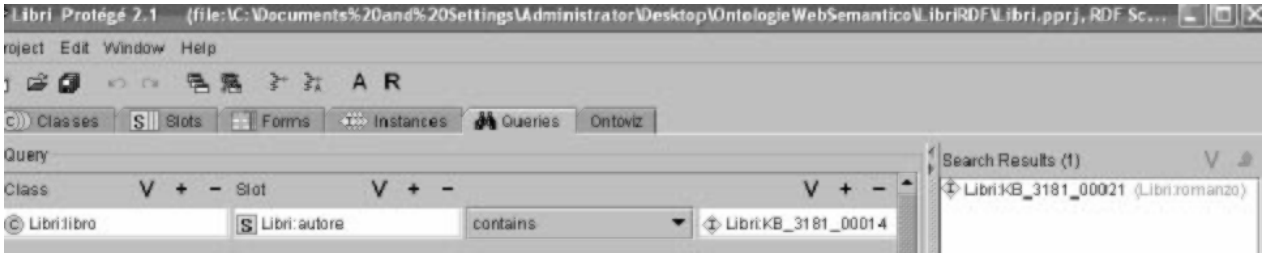
Esempio di ontologia e Knowledge Base



Le istanze di libri.

Esempio di ontologia e Knowledge Base

- A questo punto abbiamo una base di conoscenza e possiamo fare delle interrogazioni (o query).
- Sempre usando le Protege possiamo fare delle query.
- Vogliamo cercare un libro il cui autore sia Manzoni. Come possiamo vedere dalle figure precedenti
  - Libri:KB_3181_00014 è una istanza di persona che corrisponde a Manzoni.
  - Il risultato Libri:KB_3181_00021 corrisponde a I Promessi Sposi.
    - (purtroppo dobbiamo usare gli identificatori che Protege assegna automaticamente alle istanze).



Esempio di ontologia e Knowledge Base

- E’ opportuno notare che per rispondere bisogna fare una inferenza.
  - Infatti “I Promessi Sposi” sono una istanza di romanzo, non di libro.
  - Ma sapendo che romanzo è una sottoclasse di libro possiamo anche dire che sono una istanza libro.
  - Protege segnala nella risposta che l’istanza trovata appartiene alla classe romanzo.

Progettare Ontologie

Definire un'ontologia

- *Per definire una ontologia dobbiamo:*
  - Definire le classi
  - Organizzare le classi un una gerarchia tassonomica (sottoclassi-superclassi)
  - Definire le proprietà e descrivere i valori leciti per ciascuna di esse
  - Attribuire i valori alle proprietà per tutte le istanze create.

Progettare Ontologie

1. Elencare i termini più importanti

- *Si può iniziare scrivendo una lista di tutti i termini del dominio su cui vogliamo fare affermazioni*
  - Cerchiamo do rispondere alle seguenti domande:

1. Quali sono i termini di cui vogliamo parlare?
2. Quali sono le proprietà ad essi associate?
3. Cosa vogliamo dire a proposito di essi?

Progettare Ontologie

2. Definire le classi e la gerarchia

- *Ci sono diversi metodi per sviluppare l'organizzazione gerarchica dell'ontologia*

1. **Top-down**: si parte dalla definizione dei concetti più generali e si specializzano tali definizioni in concetti più specifici.
2. **Bottom-up**: si parte dalla definizione delle classi più specifiche, e si cerca di raggrupparle in gruppi concettuali più astratti.

➢ Quasi sempre si segue una combinazione dei metodi Top-down e Bottom-up.

Progettare Ontologie

3. Definire le proprietà delle classi

- *Le proprietà degli oggetti verranno modellate nelle classi relative:*

1. Proprietà "intrinsiche" properties (il gusto di un vino)
2. Proprietà "estrinsiche" (il nome di un vino)
3. Parti costituenti, se l'oggetto è strutturato (le parti del corpo)
4. Relazioni ad altri soggetti (costruttore di ...)

➢ Gli attributi sono ereditati dalle sottoclassi.

Progettare Ontologie

- 4. Definire i vincoli sulle proprietà

- *Le proprietà (slot) possono avere diverse "facet": tipo dati, valori permessi, numero di valori, ecc.:*

  1. Cardinality: quanti valori posso avere in uno slot
  2. Value type: tipo di dato
  3. Domain: a quali classi si applica questo slot
  4. Range: classi permesse per gli slot definibili sulle istanze.

Progettare Ontologie

- 5. Creare le istanze

1. *L'ultimo passo di sviluppo consiste nella creazione delle istanze delle classi presenti nell'ontologia.*

2. *A questo punto spesso si scoprono errori di modellazione nei passi precedenti.*

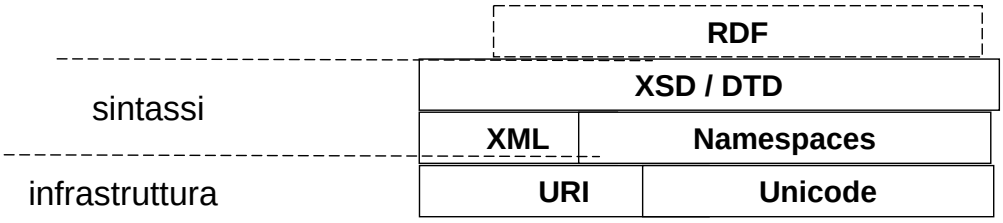
Riuso di Ontologie

- Conviene sempre analizzare se esistono ontologie disponibili che possano essere adottate, estese, raffinate, adattate, nel nostro dominio e obiettivo particolare.
- Alcuni esempi:
  - Ontolingua ontology library http://www.ksl.stanford.edu/software/ontolingua
  - DAML ontology library http://www.daml.org/ontologies
  - Ontologie OWL http://protege.stanford.edu/plugins/owl/ontologies.html
  - Sito OWL http://www.w3.org/2004/OWL/

I linguaggi di supporto ai modelli semantici sul web

Web semantico: protocolli e linguaggi

- Tre sono le tecnologie di supporto per il suo sviluppo:
  - ***URI*** (Uniform Resource Identifiers), un meccanismo generico per identificare risorse
  - ***XML*** (eXstensible MarkUp Language) una meta-sintassi utilizzabile da ogni applicazione, che assieme ad Unicode risolve il problema di distinguere dati da metadati.
  - ***RDF***, un linguaggio per esprimere affermazioni



Web Semantico: URI

➨ *Possiamo dare un URI a ogni cosa, ed ogni cosa che ha un URI può dire di "essere sul Web"*

- Gli URI sono decentralizzati.
  - Nessuna persona o organizzazione controlla chi li fa e il loro uso
  - Possibile creazione di URI per cose inesistenti o non proprie.

- URI strumenti molto flessibili ma problematici
  - Poiché ognuno può creare un URI, inevitabilmente avremo vari URI che rappresentano la stessa cosa.
  - Non c'è modo di verificare se due URI si riferiscono alla stessa risorsa.
  - Non c'è la possibilità di dire con certezza cosa significa un URI

Web Semantico:XML

- L'XML è stato scritto per essere un modo semplice di inviare documenti nel Web.
- XML è anche la sintassi di base del web semantico
  - ✓ Elimina ambiguità tra contenuto e markup
  - ✓ Permette ad ognuno di scrivere il proprio formato di documento e poi scrivere un documento in quel formato.
  - ✓ Il markup dei documenti li rende "leggibili da una macchina" che può rispondere in maniera appropriata alle sue modalità al significato codificati nel tag:
    - Es: documento con alcune che sono marcate come "":

      Un browser Web potrebbe semplicemente mostrarle in italic laddove un browser vocale potrebbe indicare l'enfasi cambiando il tono della sua voce.

Web Semantico:XML e namespace

- Problema:
- Cosa succede se si utilizzano le stesse parole in diversi linguaggi di markup, anche con significati diversi?
- Per prevenire confusione, devo in maniera unica **identificare** i miei elementi di markup
  - Si può assegnare un URI ad ognuno degli elementi ed attributi del proprio linguaggio di markup.
  - Lo si fa utilizzando gli XML Namespace.
    - In questo modo, ognuno può creare i propri tag e mescolarli con tag fatti da altri.

Web semantico: RDF e RDFS

- Un modello generale per fare asserzioni semantiche su oggetti web proposto dal W3C
- Un “oggetto web” è detto risorsa.
  - Una risorsa è qualsiasi entità identificata mediante un URI, ciò include pagine web e documenti XML

Web Semantico: RDF

- RDF permette di esprimere ogni affermazione come una tripla (Soggetto, Predicato, Oggetto)
  - (ad es.: "il documento http://www.host.org/~mrossi è stato creato da Mario Rossi"), dove il soggetto è un URI, il predicato esprime una relazione, e l'oggetto è un'altra risorsa, oppure un valore letterale.

- Oltre alle affermazioni, RDF permette di esprimere anche citazioni, ovvero reificazioni, ovvero meta-affermazioni, vale a dire affermazioni su altre affermazioni
  - (es.: "Andrea dice che il documento http://www.host.org/~mrossi è stato creato da Mario Rossi").

Web semantico: RDF e XML

- RDF non è un formato XML, ma un modello astratto (uno schema logico)
  - Esistono però *linearizzazioni in XML* di RDF (che però non sono univoche).
- RDF permette schemi multipli di metadati leggibili sia da macchine che da umani
  - Usa XML per esprimere la struttura perciò permette alle comunità di metadati di definire la vera semantica.
  - Questo approccio decentralizzato riconosce che nessun schema è appropriato per tutte le situazioni e che gli schemi hanno bisogno di un meccanismo di linking per avere una descrizione , un'identificazione ed un'usabilità nei vari contesti.

Il modello di RDF

- Il modello di RDF è basato su tre concetti:
  - Risorse: tutto ciò che viene descritto. Ogni risorsa è identificata da un URI; può quindi essere anche un oggetto non accessibile da web.
  - Proprietà: un attributo che voglio associare alla risorsa. E' una coppia attributo-valore. Ogni proprietà ha un significato specifico, una serie di valori leciti, è associabile ad uno o più tipi di risorsa.
  - Asserzioni (*statement*): l'associazione di una proprietà ad una risorsa. Ogni asserzione ha una struttura obbligata del tipo "soggetto", "predicato", "oggetto".

Soggetto (risorsa)	http://www.host.org/~mrossi
Predicato (proprietà)	Autore
Oggetto (letterale)	"Mario Rossi"

Il modello di RDF

- Per esprimere una affermazione dobbiamo identificare:
  - L'oggetto che vogliamo descrivere
  - La specifica **proprietà** dell'oggetto (o **relazione** tra oggetti) su cui vogliamo predicare
  - Il **valore** assunto dalla proprietà o **l'oggetto** con cui viene messa in relazione l'entità su cui stiamo predicando

http://www.host.org/article1.pdf ha un autore il cui valore è Mario Rossi

- Per disambiguare i termini del discorso si utilizzano URIref anche per rappresentare le relazioni (o predicati)

Soggetto	http://www.host.org/article1.pdf
Predicato	http://www.host.org/terms/author
Oggetto	Mario Rossi

http://www.host.org/article1.pdf http://www.host.org/terms/author Mario Rossi

Il modello di RDF: notazione N3

affermazione

Autore(http://www.mywebsite.com)=JohnDoe

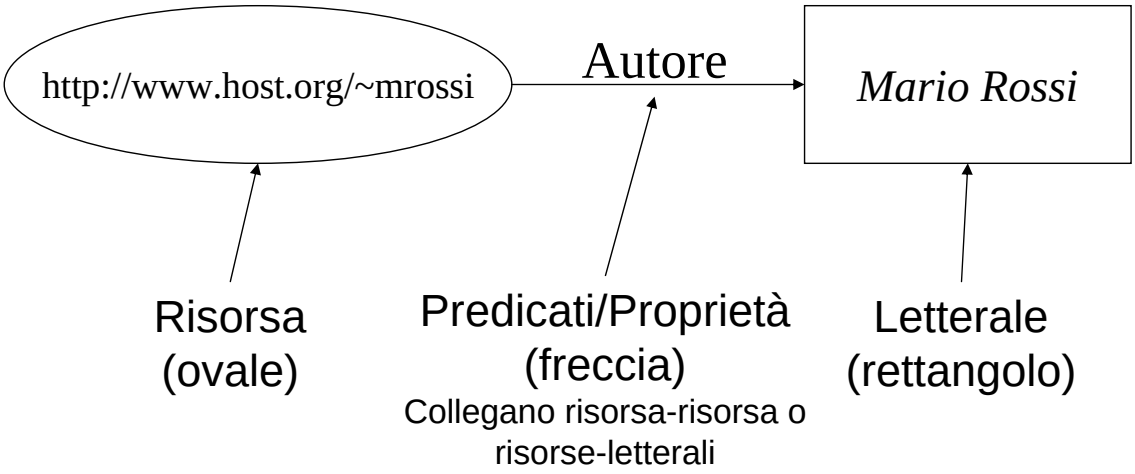
proprietà risorsa soggetto risorsa oggetto

Si può esprimere come tripla del tipo (soggetto, predicato, oggetto):

(http://www.mywebsite.com, Autore, JohnDoe)

RDF Rappresentazione grafica

- *La proprietà "Autore" della risorsa "http://www.host.org/~mrossi" vale "Mario Rossi"*



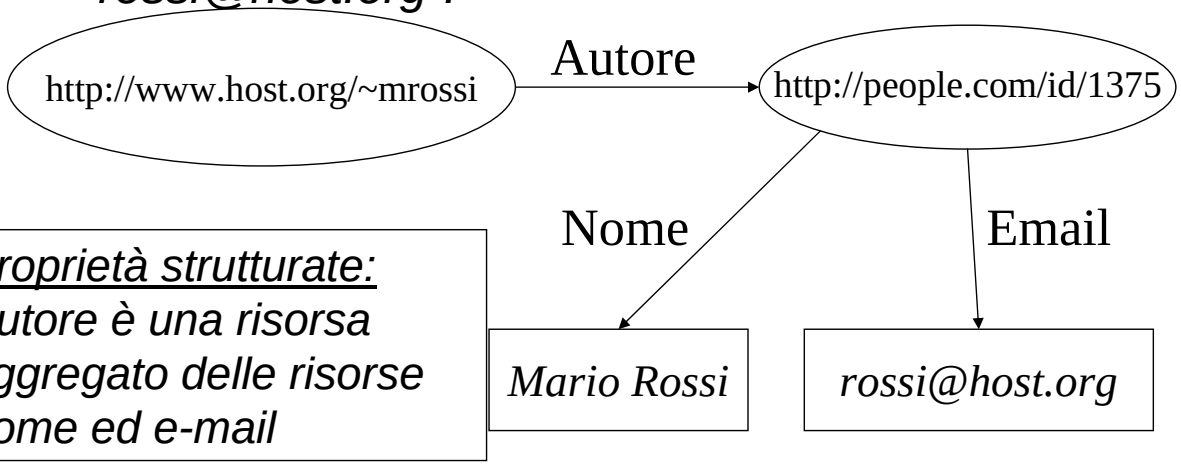
RDF - Sintassi estesa RDF/XML

- XML si propone come scelta naturale come supporto sintattico alla descrizione delle risorse
  - L'elemento radice è *rdf:RDF*
  - Ogni asserzione può essere rappresentata da un elemento *Description*
  - Un documento RDF in formato RDF/XML è composto da una sequenza di elementi Description
  - Le asserzioni riguardanti uno stesso soggetto possono essere raggruppate
- L'esempio in sintassi estesa:

```
<rdf:Description
rdf:about="http://www.host.org/~mrossi">
   <s:Autore>Mario Rossi</s:Autore>
</rdf:Description>
```

RDF: Proprietà strutturate

- *La proprietà "Autore" della risorsa "http://www.host.org/~mrossi" è quell'ente il cui nome è "Mario Rossi" e che ha e-mail "rossi@host.org".*



Proprietà strutturate:
Autore è una risorsa aggregato delle risorse nome ed e-mail

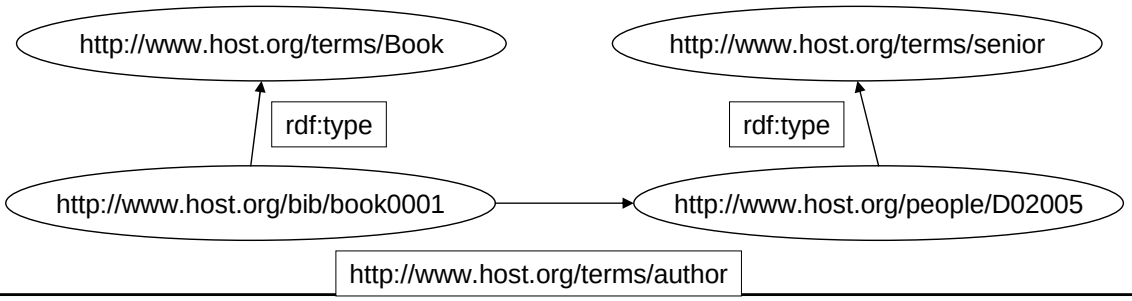
RDF: XML Sintassi estesa

- Esempio proprietà strutturate:

```
<rdf:Description
rdf:about="http://www.host.org/~mrossi">
 <s:Autore>
  <rdf:Description
rdf:about="http://people.com/id/1375">
    <s:Nome>Mario Rossi</s:Nome>
    <s:Email>rossi@host.org</s:Email>
  </rdf:Description>
 </s:Autore>
</rdf:Description>
```

Tipizzazione

- Per descrivere compiutamente una risorsa occorre specificare la categoria di risorse a cui appartiene
- Ad esempio, se il mio agente vuole cercare un libro deve poter sapere che http://www.host.org/bib/book0001 rappresenta un libro
- In RDF l'associazione tra istanza e tipo è rappresentata dalla particolare relazione *rdf:type*
- L'interpretazione del tipo viene lasciato all'applicazione



Tipizzazione

- E' possibile assegnare ad ogni risorsa un tipo appartenente ad uno schema di meta informazioni:

```
<rdf:Description
rdf:about="http://www.host.org/~mrossi">
  <s:Autore>
    <rdf:Description
rdf:about="http://people.com/id/1375">
      <rdf:type rdf:resource="/myschema.rdf#Persona"/>
      <s:Nome>Mario Rossi</s:Nome>
      <s:Email>rossi@host.org</s:Email>
    </rdf:Description>
  </s:Autore>
</rdf:Description>
```

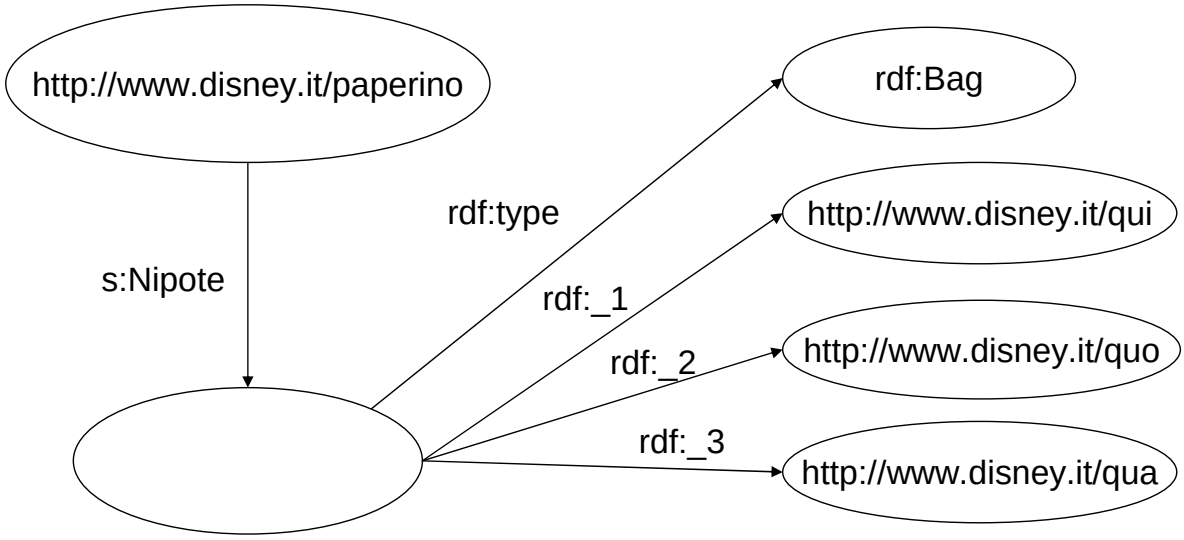
- L'attributo rdf:type specifica l'URI della definizione del tipo.

Contenitori

- A volte è importante fare riferimento ad un insieme di risorse (es, se un documento è stato creato da più autori, o se lo stesso autore ha fatto più di un documento)
- In questo caso tali risorse devono essere inserite all'interno di un contenitore che sarà l'oggetto dello statement.
- RDF definisce tre tipi di contenitori:
  - **Bag**. E' un insieme con ripetizioni. L'ordine non è rilevante.
  - **Sequence**. E' un insieme con ripetizioni ed un ordine definito tra le risorse presenti.
  - **Alternative**. E' un insieme senza ripetizioni in cui può essere scelto uno solo degli elementi. L'ordine degli elementi può essere usato per esprimere preferenza.

Rappresentazione dei contenitori

- *I nipoti di Paperino sono Qui, Quo, Qua.*



Sintassi XML dei contenitori

```
<rdf:Description
rdf:about="http://www.disney.it/paperino">
   <s:Nipote>
      <rdf:Description>
        <rdf:type rdf:resource=
          "http://www.w3.org/1999/02/22-rdf-syntax-
ns#Bag"/>
        <rdf:_1 rdf:resource="http://www.disney.it/qui"/>
        <rdf:_2 rdf:resource="http://www.disney.it/quo"/>
        <rdf:_3 rdf:resource="http://www.disney.it/qua"/>
      </rdf:Description>
   </s:Nipote>
</rdf:Description>
```

- Analogamente si useranno i tipi rdf:Seq per le sequenze e rdf:Alt per le alternative.

RDF: mondo aperto

- RDF fa l'assunzione di mondo aperto, questo significa che chiunque può effettuare asserzioni su risorse definite ovunque.
  - Il modello complessivo è dato dall'unione delle asserzioni
- Ognuno può sempre introdurre un elemento Description con un attributo rdf:about che lo associa ad una risorsa descritta altrove
  - E se l'informazione introdotta da terze parti è scorretta?
    - Il sistema gestirà in maniera scorretta le risorse
  - E se l'informazione è addirittura contraddittoria?
    - Questo significa che possiamo derivare qualunque cosa
- Servono quindi dei criteri di certificazione dell'informazione
- Questo però richiede delle meta-asserzioni, cioè delle asserzioni che riguardano altre asserzioni

Reificazione

- Come è possibile fornire meta-informazioni su una meta-informazione? Ad esempio come posso esprimere la frase *«Andrea afferma che Mario Rossi è l'autore della risorsa "http://www.host.org/~mrossi"»*?
- Questo in breve significa attribuire la proprietà *«afferma»* allo statement *«Mario Rossi è l'autore della risorsa "http://www.host.org/~mrossi"»*. Occorre pertanto considerare la meta-informazione come una risorsa da descrivere.
- Questa procedura si chiama *reificazione* (riduzione a ente/oggetto) della asserzione (o statement). Dopo avere reificato l'asserzione potrò esprimere ulteriori proprietà su di essa.

Reificazione

- Uno statement di cui vado a considerare esplicitamente l'identificatore diventa uno statement reificato.
- E può essere usato come oggetto di un altro predicato:

```
<rdf:Description>
 <rdf:subject
rdf:resource="http://www.host.org/~mrossi"/>
 <rdf:predicate
rdf:resource="/myschema.rdf#Autore"/>
 <rdf:object>Mario Rossi</rdf:object>
 <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#Statement"/>

 <s:AffermatoDa>Andrea</s:AffermatoDa>
</rdf:Description>
```

RDF e RDF Schema

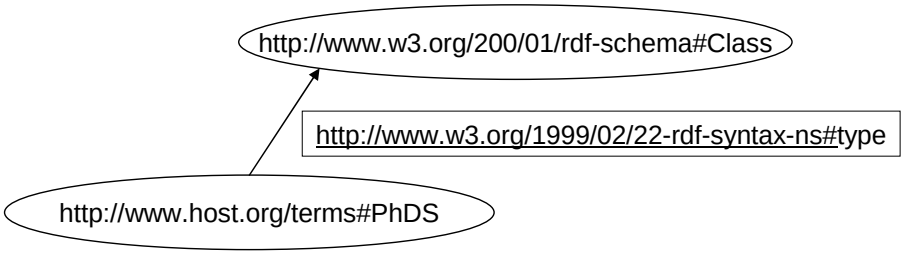
- RDF è un modello che permette di esprimere sul Web affermazioni sul mondo
- Le proprietà RDF rappresentano relazioni tra risorse e possono essere viste come attributi di risorse ai quali viene associato un valore
- RDF non fornisce alcun meccanismo per descrivere tali proprietà, nè per descrivere le relazioni tra le proprietà ed altre risorse
- Questo è il ruolo di RDF Schema
- *RDF Schema definisce classi e proprietà che possono essere usate per descrivere classi, proprietà ed altre risorse*

RDF Schema

- RDF Schema è una estensione semantica di RDF
  - Noto anche come RDF Vocabulary Description Language
- Non definisce proprietà descrittive come *author* ma specifica come fare per dare un nome e una descrizione a proprietà e classi
- Fornisce meccanismi per descrivere gruppi di risorse correlate e le relazioni tra queste risorse
- Il sistema di classi e proprietà che definisce è simile ai sistemi di tipizzazione di un linguaggio OO come Java
  - La differenza è che RDF Schema definisce le proprietà in base alle classi di risorse a cui si applicano (non viceversa)

RDFS Vocabolary: Classi

- Le classi sono aggregati di individui (istanze). Ogni classe rappresenta un tipo di risorsa su cui si costruisce il modello RDF
- Il *namespace* di riferimento è: *http://www.w3.org/2000/01/rdf-schema#*
- Ogni classe è una risorsa in relazione *rdf:type* con la risorsa *http://www.w3.org/2000/01/rdf-schema#Class*

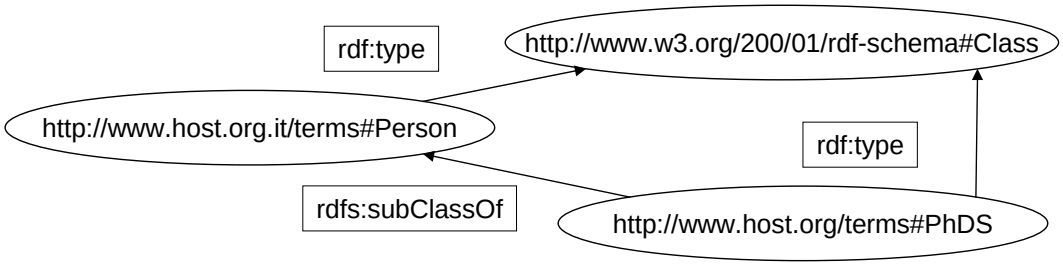


Data una classe, l'insieme delle sue istanze è detta *class extension*

RDFS Vocabolary: Classi

- Analogamente ai linguaggi OO, RDF(S) permette di organizzare le classi in gerarchie
- Il vocabolario RDF(S) mette a disposizione la relazione *subClassOf*
  - Transitiva, antisimmetrica e non riflessiva (niente cicli)
  - Es: un PhD Student è (is-a) una Persona
- La semantica di RDF/RDF(S) è studiata in modo da poter effettuare inferenze anche assumendo di non possedere tutta la conoscenza sull'argomento
  - Posso usare nell'RDF classi e relazioni che non compaiono nello schema

RDFS Vocabolary: Classi



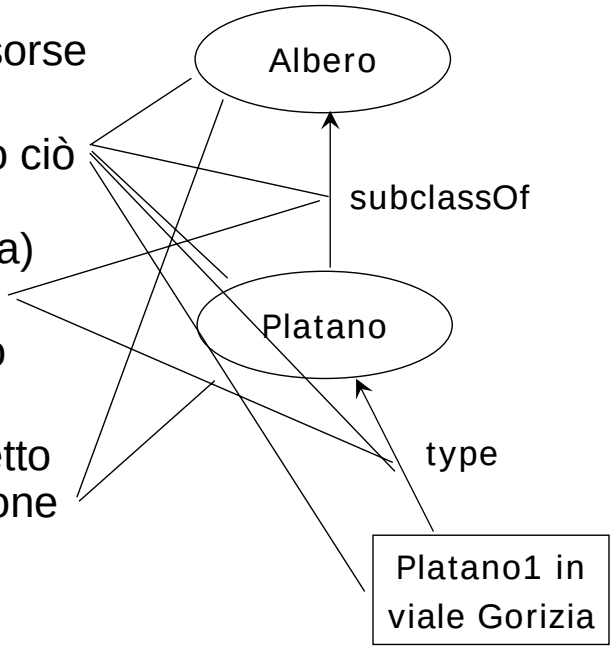

```
<rdfs:Class rdf:ID="Person"/>
<rdfs:Class rdf:ID="PhDS">
        <rdfs:subClassOf rdf:resource="#Person"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="PhDS">
        <rdfs:subClassOf>
                <rdfs:Class rdf:ID="Person"/>
        </rdfs:subClassOf>
</rdfs:Class>
```

RDFS Vocabolary: classi base

Sono definite le seguenti classi (risorse di tipo rdfs:Class):

- rdfs:Resource: rappresenta tutto ciò che viene descritto in RDF (corrispondente di Object in Java)
- rdfs:Property: rappresenta il sottoinsieme di risorse che sono proprietà
- rdfs:Class: corrisponde al concetto di tipo (corrispondente alla nozione di classe nei linguaggi OO)



RDFS Vocabolary: proprietà base

➢ Una proprietà (ovvero risorse di tipo rdfs:Property)
è una relazione tra risorse *subject* e risorse *object*

RDFS definisce le proprietà:

- rdfs:type: indica che una risorsa è istanza di una classe (1 risorsa, molti tipi)
- rdfs:subClassOf: relazione “sottoinsieme - sovrainsieme” tra classi (una classe può essere sottoclasse di molte classi)
- rdfs:subPropertyOf: una proprietà è una specializzazione di un’altra (1 proprietà, 0-n specializzazioni)
- rdfs:seeAlso: una risorsa contiene informazioni su un’altra risorsa
- rdfs:isDefinedBy: sottoproprietà di rfds:seeAlso, indica quale risorsa definisce un’altra risorsa (es. quale schema?)

RDFS Vocabolary: vincoli

rdfs:ConstraintProperty è una sottoclasse di rdf:Property. Le sue istanze sono proprietà utilizzate all’interno di vincoli.

RDF Schema permette di imporre:

- Vincoli di dominio (rdfs:domain)
  - Istanze di rdfs:ConstraintProperty
  - Vincolano l’applicazione di una proprietà a una o più classi
- Vincoli di intervallo (rdfs:range)
  - Istanze di rdfs:ConstraintProperty
  - Vincolano il valore di una proprietà a un determinato intervallo scelto su istanze di classi

RDFS vocabulary: esempio di vincoli

```
<rdf:Description ID="appartieneA">
<rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#Property"/>
<rdfs:domain rdf:resource="Oggetto"/>
<rdfs:range rdf:resource="Persona"/>
</rdf:Description>
```

Vincola la proprietà appartieneA:

- All'applicazione su istanze della classe Oggetto
- Ad assumere valori che siano istanze della classe Persona

RDFS Vocabolary Usare domain e range

- Hanno l'obiettivo di supportare un uso significativo delle proprietà e delle classi nei dati RDF
  - Limitazioni sui valori delle proprietà
  - Quali classi ha senso che abbiano determinate proprietà
- Forniscono uno strumento per descrivere queste informazioni senza stabilire se e come una applicazione deve farne uso
- I vocabolari RDF possono descrivere relazioni tra elementi di vocabolari diversi e sviluppati in modo indipendente
  - Posso creare nuove proprietà con *domain* e *range* il cui valore è una classe definita in un altro namespace

RDF e RDFS *discussione*

- Nei linguaggi di programmazione lo schema dei tipi ha sempre uno scopo *prescrittivo*. Il programma che non rispetta i vincoli semplicemente è scorretto
- In RDF lo Schema fornisce informazioni aggiuntive, ma si lascia all'applicazione la scelta dell'uso di queste informazioni
  - *Scenario 1* – uso prescrittivo: l'applicazione interpreta lo schema come dei vincoli (constraints) sui modelli leciti
  - Scenario 2 – uso deduttivo sul modello: si possono usare le informazioni sullo schema per dedurre ulteriore conoscenza. Es: ho un libro di cui conosco l'autore, posso dedurre che la risorsa autore è una persona
  - *Scenario 3* – uso deduttivo sullo schema: incontro un libro che ha per autore una *Company*, in questo caso potrebbe esserci una inconsistenza, oppure posso trovare che Company subClassOf Person

Limiti di RDFS: *Ragionare sul web*

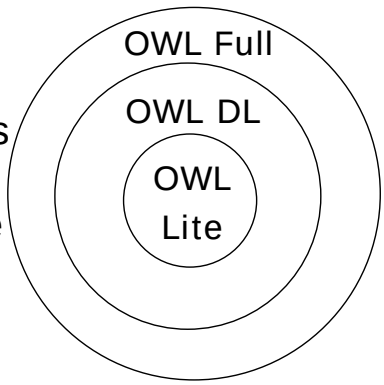
- Il livello successivo è la possibilità di trarre conclusioni dalle affermazioni.
  - RDF non basta,
    - Possiede un limitato potere espressivo (subClassOf, subPropertyOf, range, domain)
    - Ed ha una semantica non ben definita
  - ci vuole un linguaggio per esprimere inferenze (creazione di nuove informazioni a partire da quello che si ha o si trova)
    - Linguaggi per ontologie
      - Intuitivi / espressivi
      - Sintassi ben specificata, semantica formale, adeguato potere espressivo
  - Una proposta di linguaggio per inferenze è OWL (Ontology Web Language) all'interno del W3C nel contesto RDF

OWL: Ontology Web Language

- Standard del W3C
- Si basa su RDF (per le istanze) e RDFS (per classi e proprietà) e ne estende l'espressività
- Possiede una semantica formale e supporto al ragionamento efficiente (*Description Logic*)

Tre livelli di espressività (e complessità) crescente

- ***OWL Lite:***
  - semplice da usare e implementare ma scarsamente espressivo .
- ***OWL DL***:
  - abbastanza espressivo
  - la stessa espressività delle Description Logics
  - dotato di procedure di ragionamento di complessità nota, approfonditamente studiate e ormai ben ottimizzate
- ***OWL Full***:
  - consente rappresentazioni che vanno al di là della logica predicativa del primo ordine
  - è molto espressivo ma non decidibile e quindi problematico dal punto di vista della meccanizzazione del ragionamento



Struttura delle ontologie in OWL

- Poichè il Web Semantico è distribuito, OWL deve permettere la raccolta di informazioni da risorse distribuite.
  - Questo è parzialmente raggiunto in quanto si permette alle ontologie di essere correlate tra loro, includendo la possibilità di effettuare una importazione esplicita delle informazioni da altre ontologie.
- Inoltre OWL effettua una assunzione di *mondo aperto* (OWA - Open World Assumption) cioè assume che la descrizione delle risorse non sia confinata ad un unico file o ad un unico obiettivo.
  - Sebbene la classe C1 sia definita originariamente in una ontologia O1, questa può essere estesa anche in altre ontologie. Le conseguenze di queste nuove asserzioni circa la classe C1 sono *monotone*. Le nuove informazioni infatti non possono ritrattare le informazioni precedenti; possono essere però contraddittorie, ma possono solo *aggiungere* fatti e conseguenze e mai *cancellarli*.
- La possibilità di queste contraddizioni è un qualcosa che il progettista di ontologie deve bene tenere in considerazione. Ci si aspetta inoltre che gli strumenti di supporto aiutino a rilevare questi casi.

Struttura delle ontologie in OWL

- Spazio dei nomi
  - Prima di poter usare un insieme di termini, abbiamo bisogno di una precisa indicazione di quali saranno i vocabolari specifici che saranno utilizzati.

```
<rdf:RDF
xmlns     ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xml:base  ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

Struttura delle ontologie in OWL

- Intestazione dell'ontologia
  - Una volta che gli spazi dei nomi sono stati dichiarati, normalmente includiamo una collezione di affermazioni riguardanti l'ontologie e raggruppate sotto una etichetta owl:Ontology. Queste etichette svolgono compiti critici molto comuni quali i commenti, il controllo della versione e l'inclusione di altre ontologie.

```
<owl:Ontology rdf:about="">
<rdfs:comment>An example OWL ontology</rdfs:comment>
<owl:priorVersion rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide20031215/wine"/>
<owl:imports rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food"/>
<rdfs:label>Wine Ontology</rdfs:label>
```

Principali caratteristiche di OWL

- Classi
  - subClassOf, intersectionOf, unionOf, complementOf, enumeration, equivalence, disjoint
- Proprietà
  - symmetric, transitive, functional, inverse Functional
  - range, domain, subPropertyOf, inverseOf, equivalentProperty
- Affermazioni *sulle istanze*
  - sameIndividualAs, differentFrom, AllDifferent

Classi in OWL

- In OWL i termini sono denominati *descrizioni di classi*,
- gli operatori per la definizione di termini sono denominati *costruttori di classi*
- e i ruoli sono denominati *proprietà*.
- OWL prevede sei tipi di descrizioni di classi:
  - identificatore
  - enumerazione
  - restrizione di proprietà
  - intersezione
  - unione
  - complemento.

Classi in OWL

- Ogni descrizione di classe descrive una risorsa di tipo owl:Class.
- Nel caso più semplice la descrizione consta di un *identificatore* della classe (un URI).
- OWL prevede due identificatori predefiniti per la *classe universale* e la *classe vuota*:
  - owl:Thing, corrispondente a T;
  - owl:Nothing, corrispondente a ⊥.
  - Ogni individuo (*a*1, ... , *an*, ...) è istanza della classe owl:Thing.
- ***Enumerazione***
  - Una classe finita può essere descritta dall'*enumerazione* di (owl:oneOf) tutti gli individui che le appartengono: {*a*1,...,*an*}.
  - Ogni nominale (*a*1, ... , *an*, ...) va interpretato come un URI.

Classi in OWL

- E' possibile esprimere una classe come sottoclasse:

```
<owl:Class rdf:ID="Fiume">
      <rdfs:subClassOf rdf:resource="#CorsoDAcqua"/>
</owl:Class>
```

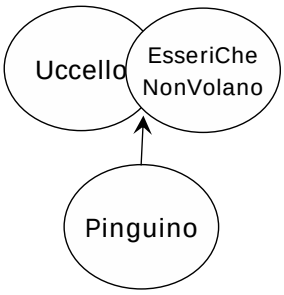
- E' possibile definire una sottoclasse esprimendo delle restrizioni rispetto a un'altra classe:

```
<rdfs:subClassOf rdf:ID="Fiume">
      <owl:Restriction>
            <owl:onProperty rdf:resource="#sfocia"/>
            <owl:allValuesFrom rdf:resource="#Mare"/>
      </owl:Restriction>
</rdfs:subClassOf>
```

Altri modi di definire una classe

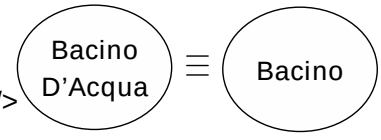
- **intersectionOf:**

```
<owl:Class rdf:ID="Pinguino">
    <rdfs:subclassOf>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Uccello"/>
          <owl:Class rdf:about="#EsseriCheNonVolano"/>
        </owl:intersectionOf>
    </rdfs:subclassOf>
</owl:Class>
```



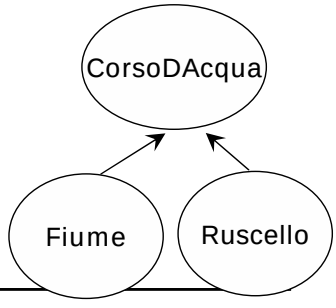
- **equivalentClass:**

```
<owl:Class rdf:ID="BacinoDAcqua">
    <owl:equivalentClass rdf:resource="http://www.other.org#Bacino"/>
</owl:Class>
```



- **disjointWith:**

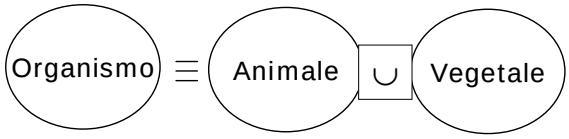
```
<owl:Class rdf:ID="Fiume">
      <rdfs:subClassOf rdf:resource="#CorsoDAcqua"/>
      <owl:disjointWith rdf:resource="#Ruscello"/>
      <owl:disjointWith rdf:resource="#Rivolo"/>
      <owl:disjointWith rdf:resource="#Torrente"/>
</owl:Class>
```



Altri modi di definire una classe

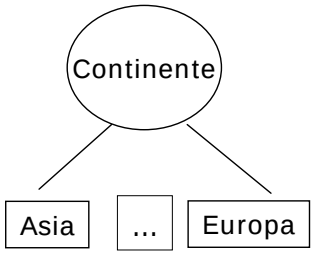
- unionOf:

```
<owl:Class rdf:ID="Organismo">
    <rdfs:subclassOf>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Animale"/>
            <owl:Class rdf:about="#Vegetale"/>
        </owl:intersectionOf>
    </rdfs:subclassOf>
</owl:Class>
```



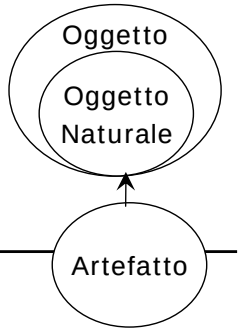
- oneOf (definizione estensionale di classe):

```
<owl:Class rdf:ID="Continente">
    <rdfs:subClassOf rdf:resource="#Regione"/>
    <owl:oneOf rdf:parseType="Collection">
        <geo:Continente rdf:about="http://www.asia.org"/>
        <geo:Continente rdf:about="http://www.australia.org"/>
        <geo:Continente rdf:about="http://www.europa.org"/>
        <geo:Continente rdf:about="http://www.america.org"/>
        <geo:Continente rdf:about="http://www.africa.org"/>
    </owl:oneOf>
</owl:Class>
```



- complementOf:

```
<owl:Class rdf:ID="Artefatto">
    <rdfs:subClassOf rdf:resource="#Oggetto"/>
    <owl:complementOf rdf:resource="#OggettoNaturale"/>
</owl:Class>
```



Proprietà in OWL

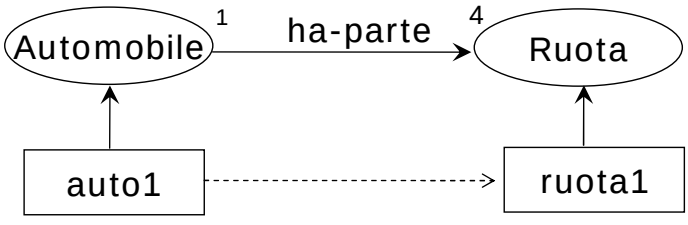
- Coerentemente con quanto previsto da RDFS, in OWL anche le *proprietà* (corrispondenti ai ruoli nelle DL) possono essere visti come particolari classi.
- Come tali le proprietà possono avere sottoproprietà ed essere combinate con vari costruttori.
- *Così come ogni classe di individui è una risorsa di tipo owl:Class, tutte le proprietà sono risorse di tipo rdf:Property.*

Proprietà in OWL

- In OWL si possono specificare i seguenti aspetti relativi alle proprietà:
  - identificatore di proprietà
    - Un *identificatore di proprietà* corrisponde a un ruolo
  - dominio e codominio (*range*)
    - Di una proprietà possono essere specificati il *dominio* (rdfs:domain) e il *codominio* (rdfs:range).
  - Sottoproprietà
    - Una proprietà può essere definita come *sottoproprietà* di (rdfs:subPropertyOf) un'altra proprietà.
  - proprietà equivalente
    - Una proprietà può essere definita come *equivalente* (owl:equivalentProperty) a un'altra proprietà
  - proprietà inversa.
    - Data una proprietà *R* si può definire la *proprietà inversa* (owl:inverseOf) *R*–.

Istanze in OWL

- Si istanzia creando un tag con il nome della classe:
  <Fiume rdf:ID="Tevere"/>
- Si lega l'istanza ad altre istanze mediante le ObjectProperty definite:



<Ruota rdf:ID="ruota1">
<Automobile rdf:ID="auto1">
  <ha-parte rdf:resource="#ruota1"/>
</Automobile>

Individui e fatti in OWL

- In OWL le asserzioni sul mondo reale sono detti *fatti*. Introduciamo qui due tipi di fatti:
  - fatti relativi all'appartenenza di un individuo a una classe o ai valori di una proprietà di un individuo;
  - fatti relativi all'identità di uno o più individui.
- *Appartenenza a una classe e valori di una proprietà*
  - In OWL è possibile specificare che un individuo *a* appartiene a una classe *C*, $C(a)$,
  - oppure che una proprietà *R* di un individuo *a* ha valore *b*, $R(a,b)$.
- *Identità*
  - Il linguaggio OWL non assume che gli individui abbiano nome unico. Quindi è possibile asserire che due nomi fanno riferimento allo stesso individuo (owl:sameAs): $a = b$.
  - Analogamente è possibile asserire che due nomi fanno riferimento al individui distinti (owl:differentFrom): $a \neq b$.
  - È anche possibile asserire che n individui sono tutti distinti fra loro (owl:AllDifferent).

Conseguenze dell'uso di ontologie sul Web

Conseguenze uso ontologie

➨Dal recupero di un'informazione da una base di dati si passa al recupero dell'informazione su una base di conoscenza

☆La diffferenza principale fra una base di conoscenze e una base di dati è la possibilità di condurre *ragionamenti* in modo automatico

➩Come si realizzano questi ragionamenti?

Conseguenze uso ontologie

➨In logica, principalmente, il "ragionamento" è di tipo deduttivo

- procedimento che porta a verificare se un enunciato *X* (es, l'equivalenza di due termini) è *conseguenza logica* di una base di conoscenze.
- non si prende quindi in considerazione i ragionamenti di tipo
  - induttivo
    - va dai casi singolari (fatti) alle affermazioni generali;
  - abduttivo
    - abduttivo va dagli effetti alle possibili cause

Conseguenze uso ontologie

☆ *Approfondiamo il "meccanismo di ragionamento" e suddividiamolo in tre aspetti*

- *compito di ragionamento* i tipi di enunciati che si desidera dedurre da una base di conoscenze;
- *procedura di ragionamento* l'algoritmo per la deduzione degli enunciati;
- *servizio di ragionamento*: i servizi basati su procedure di ragionamento che uno strumento mette a disposizione delle applicazioni che accedono alla base di conoscenze.

Conseguenze uso ontologie

☆ *I compiti di ragionamento più significativi per le applicazioni, basate sulle Description Logic, sono*

- *sussunzione*: data una base di conoscenza, stabilire se una sussunzione $C \subseteq D$ è conseguenza logica della base,
- *equivalenza*: data una base di conoscenza, stabilire se un'equivalenza $C \equiv D$ è conseguenza logica di essa,
- *soddisfacibilità:* data una base di conoscenza, stabilire se un termine C è soddisfacibile, nel senso che applicare il termine a un individuo del dominio non comporta una contraddizione logica;
- *disgiunzione*: data una base di conoscenza, stabilire se due termini C e D sono disgiunti, nel senso applicare i due termini contemporaneamente allo stesso individuo comporta una contraddizione logica

Conseguenze uso ontologie

- Si può dimostrare che è possibile ***ridurre*** i quattro compiti di ragionamento fondamentali ad una sola procedura di ragionamento
  - In particolare la riduzione dei compiti di ragionamento alla sola procedura di soddisfacibilità è la strada che si segue per implementare i servizi di ragionamento per le DL molto espressive ma decidibili

Ciò significa che, almeno in linea di principio, tutti i compiti di ragionamento possono essere ridotti al solo tipo di problema di soddisfacibilità da implementare con un solo algoritmo.

Conseguenze uso ontologie

☆*Ragionare sul web: TRUST*

- Per realizzare inferenze corrette si deve partire da informazioni corrette
  - Che deduzioni posso ottenere dalla combinazione di due o più collezioni di informazioni, se tra di loro esistono affermazioni contraddittorie?
  - Inoltre un limite fondamentale delle logiche di questo tipo è dato dal fatto che non solo le affermazioni contraddittorie non determinano nuova informazione, ma possono essere usate per generare qualunque inferenza (es:$A \wedge \neg A \Rightarrow$?)

☆*Problema della veridicità ed affidabilità delle informazioni*

  - Il passo successivo all'uso di ontologie è cercare di ottenere una rete di affermazioni di affidabilità e fiducia (trust) sulle collezioni di informazioni, in cui viene espresso il valore di affidabilità delle affermazioni contenute.

Web semantico: dove siamo?

- http://www.w3.org/DesignIssues/Semantic.html
  - An attempt to give a high-level plan of the architecture of the **Semantic Web** by Tim Berners-Lee.
- http://www.w3.org/2001/sw/
  - Semantic web World Wide Web consortium
- http://www.semanticweb.org/
  - Portal of the **Semantic Web** Community. Projects, tools and ongoing events.
- http://www.websemantico.org/
  - La risorsa italiana del web semantico
- http://jena.sourceforge.net/
  - *Jena – A Semantic Web Framework for Java*