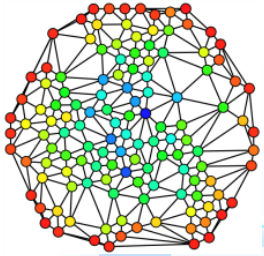


Sistemi Collaborativi e di Protezione (SCP)

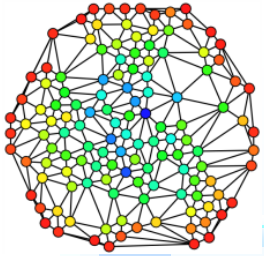
Corso di Laurea in Ingegneria

Knowledge Management and Ontology Languages

Eng. Ph.D. Michela Paolucci
Eng. Ph.D. Gianni Pantaleo
Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796363

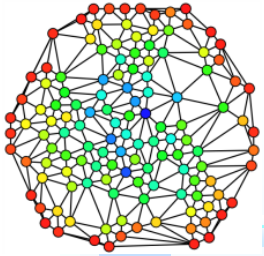
Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet

michela.paolucci@unifi.it gianni.pantaleo@unifi.it
http://www.disit.dsi.unifi.it/



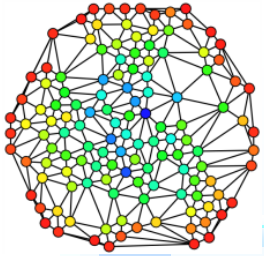
Outline

- Rappresentazione della Conoscenza e Ontologie
  - La conoscenza
  - Linguaggi di Rappresentazione
  - Ragionamento Automatico
  - Sistemi di rappresentazione della conoscenza
- OWL: *Ontology Web Language*
  - Introduzione
  - Descrizione di Classi e Assiomi
  - Proprietà
  - Individui e Fatti
  - Servizi di Ragionamento
  - Interrogare la Conoscenza: SPARQL



Outline

- Rappresentazione della Conoscenza e Ontologie
  - La conoscenza
  - Linguaggi di Rappresentazione
  - Ragionamento Automatico
  - Sistemi di rappresentazione della conoscenza
- OWL: *Ontology Web Language*
  - Introduzione
  - Descrizione di Classi e Assiomi
  - Proprietà
  - Individui e Fatti
  - Servizi di Ragionamento
  - Interrogare la Conoscenza: SPARQL



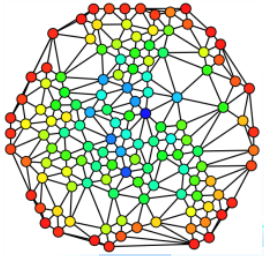
Rappresentazione della Conoscenza e Ontologie

La conoscenza

Linguaggi di rappresentazione

Ragionamento automatico

Sistemi di rappresentazione della conoscenza



La Conoscenza

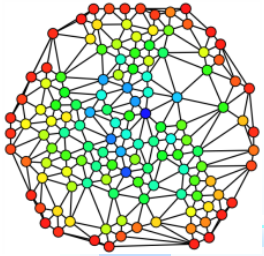
Un **sistema basato su conoscenza** è un **sistema informativo** in grado di sfruttare l'**informazione** contenuta in una **base di conoscenza** *(knowledge base, KB)* mediante **procedure automatiche di ragionamento**

Conoscenza = Informazione disponibile per l'Azione
(Dretske, 1981)

Nei computer risiede una grande quantità di informazione …

… solo una piccola parte di essa può essere sfruttata per agire …

… poiché la maggior parte dei file memorizzati nei computer è in **linguaggio naturale**



Forme di conoscenza

Implicita

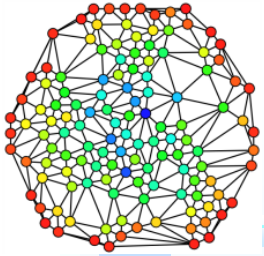
- Posseduta dalle persone
- Comunicabile in forma verbale o scritta

Tacita

- Presente nelle menti degli individui
- Difficile da comunicare verbalmente

Esplicita

- Strutturata (data base, XML+DTD, XML+XML-Schema, ecc.)
- Semi-strutturata (XML, ecc.)
- Debolmente strutturata (HTML, testi tabulati, ecc.)
- Non strutturata (documenti in linguaggio naturale, disegni, ecc.)



Acquisizione e Conservazione

Fonti di Conoscenza:

Esperienza diretta

- Interazione del soggetto con il suo ambiente

Ragionamento

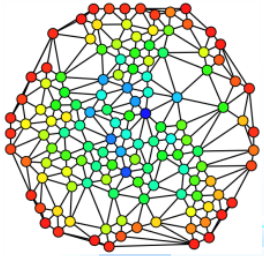
- Deduttivo/inferenza (conclusioni ← premesse)
- Induttivo (regole generali ← fatti specifici)
- Abduttivo (possibili cause ← effetti osservati)

Comunicazione

- Uso di sistemi di segni *(in particolare il linguaggio naturale)* per trasferire informazioni da un soggetto ad un altro

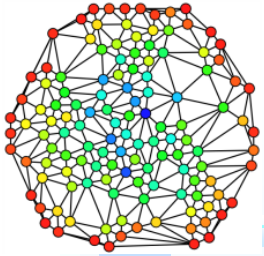
Funzione della memoria:

Capacità di Conservare nel tempo elementi di conoscenza e soprattutto di reperirli con efficienza quando occorre farne uso



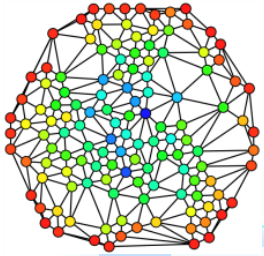
Interoperabilità (1)

- Per molti anni le aziende hanno sviluppato e utilizzato applicazioni "**chiuse**" (formato proprietario)
- Difficile interazione tra applicazioni distinte
- Le tecnologie basate su XML offrono un buon livello di interoperabilità all'interno di una singola azienda
- Intranet aziendali e integrazione di ***dati*** e ***servizi web*** (web services)



Interoperabilità (2)

- Le applicazioni "**aperte**" devono essere in grado di interoperare con applicazioni di aziende diverse (ambito extranet)
- Le tecnologie basate su XML sono necessarie ma non sufficienti:
  - XML codifica e standardizza la sintassi dei dati ma non la loro semantica
- Ad esempio, due documenti XML possono contenere le seguenti stringhe:
  - `"<spesa>85</spesa>"` `"<costo>85</costo>«`
- Una applicazione informatica non può scoprire che si tratta di due diverse codifiche della stessa informazione, a meno che non sia disponibile la conoscenza del fatto che "prezzo" e "costo" sono sinonimi (conoscenza terminologica), almeno nel contesto dell'applicazione in esame



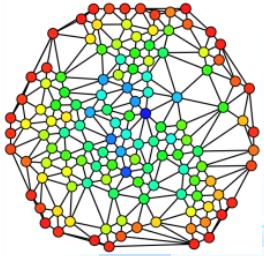
Rappresentazione della Conoscenza e Ontologie

La conoscenza

Linguaggi di rappresentazione

Ragionamento automatico

Sistemi di rappresentazione della conoscenza



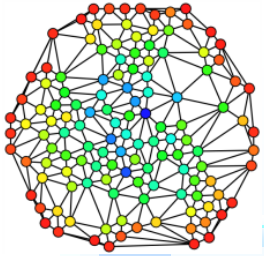
La Logica simbolica

Problema:

- **Rappresentare** la conoscenza in formato **machine-readable** *(in modo che un computer possa "leggere" la conoscenza rappresentata e utilizzarla per eseguire compiti d'interesse applicativo)*

Soluzione:

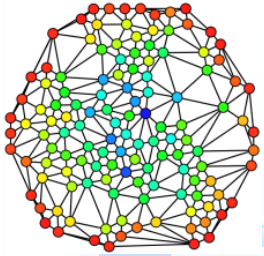
- **Rappresentazione dichiarativa** tramite **logica simbolica** *(formale)*, ed in particolare la **logica dei predicati del primo ordine** *(First Order Logic, FOL)*



First Order Logic (FOL)

- In FOL tutte le rappresentazioni riguardano un insieme non vuoto di ***individui***, detto ***universo*** (o dominio)
- Per tutti gli individui è possibile rappresentare:
  - Proprietà
  - Relazioni che li legano

- Concetto di '***fatto***'. Un fatto è dato dal sussistere:
  - di una proprietà di un determinato individuo (es. "Barbara è bionda", "Luigi ha 21 anni")
  - oppure di una relazione fra più individui (es. "Alberto è più alto di Barbara", "Alberto ha dato il suo cellulare a Barbara", ecc.)



Linguaggi di rappresentazione

- Un linguaggio per la rappresentazione di conoscenza è un linguaggio formale, con sintassi testuale o grafica, le cui espressioni sono utilizzate per rappresentare elementi di conoscenza
- **Esempio:** Rappresentazione del significato del **termine** "madre" come "donna con almeno un figlio"
  - ♣ Linguaggio naturale:

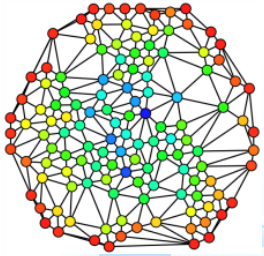
    (x è una madre)se e solo se(*x* è una donna ed esiste almeno un *y* tale che *x* è genitore di *y*)

  - ♣ First Order Logic (FOL):

    ∀ x (MADRE(x) ↔ DONNA(x) ∧ ∃ y GenDi(x,y))

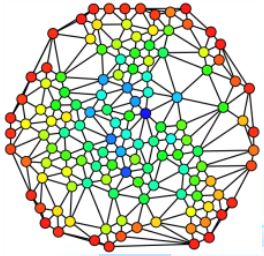
  - ♣ Logic Programming (LP):

    madre(X) :- donna(X), genDi(X,Y).



Notazione classica per FOL

- Quantificatore Universale:
  - ∀ ***x*** … *(per ogni* ***x*** *…)*
- Quantificatore Esistenziale:
  - ∃ ***y*** … *(esiste un* ***y*** *tale che …)*
- Connettivo Bicondizionale:
  - … ↔ … *(… se e solo se …)*
- Connettivo di Congiunzione:
  - … ∧ … *(… e …)*
- Predicati Mono-argomentali:
  - **`MADRE(`*x*`)`** *(proprietà su* ***x****)*
  - **`DONNA(`*x*`)`** *(proprietà su* ***x****)*
- Predicati Bi-argomentali:
  - **`GenDi(`*x*`,`*y*`)`** *(relazione binaria tra* ***x*** *e* ***y****)*



Altre possibili notazioni

Notazione classica per FOL:

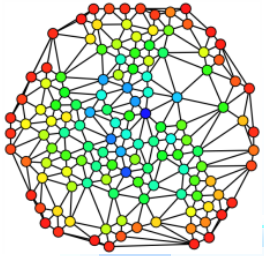
$\forall x$ **(MADRE(**x**)** $\leftrightarrow$ **DONNA(**x**)** $\wedge \exists y$ **GenDi(**x**,**y**))**

Notazione ASCII per FOL:

```
(forall (?x)
  (iff (MADRE ?x)
     (and (DONNA ?x)
        (exists (?y) (GenDi ?x ? y)))))
```

Notazione XML per FOL:

```
- <formula>
  - <forall>
    - <varlist>
        <var>x</var>
      </varlist>
    - <iff>
      - <atom>
          <con>MADRE</con>
          <var>x</var>
        </atom>
      - <and>
        - <atom>
            <con>DONNA</con>
            <var>x</var>
          </atom>
        - <exists>
          - <varlist>
              <var>y</var>
            </varlist>
          - <atom>
              <con>GenDi</con>
              <var>x</var>
              <var>y</var>
            </atom>
          </exists>
        </and>
      </iff>
    </forall>
  </formula>
```



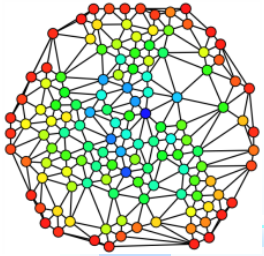
Rappresentazione della Conoscenza e Ontologie

La conoscenza

Linguaggi di rappresentazione

Ragionamento automatico

Sistemi di rappresentazione della conoscenza



La deduzione

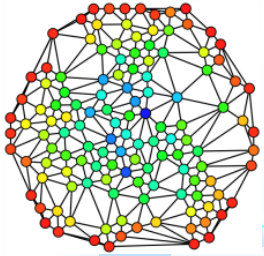
- Nel contesto in cui ci stiamo muovendo, per "**ragionamento**" si intende il ragionamento **deduttivo** *(o deduzione)*

- Una **deduzione** è un processo che fa passare da alcune espressioni (dette premesse o ipotesi) a un'espressione (detta conclusione o tesi), in modo tale da conservare l'eventuale verità delle premesse: in altre parole, se le premesse sono vere, lo sarà anche la conclusione.

- Ad esempio, dati come premesse:
  1. la **definizione** di "madre"
  2. il fatto che ***laura*** è una **`DONNA`**
  3. il fatto che ***laura*** è **`GenitoreDi`** di ***franco***

  Si può dedurre come conclusione che:

  laura è una **`MADRE`**



Deduzione naturale (calcolo)

1. $\forall x\ (\mathtt{MADRE}(x) \leftrightarrow \mathtt{DONNA}(x) \wedge \exists y\ \mathtt{GenDi}(x,y))$
2. $\mathtt{DONNA}(\mathit{laura})$
3. $\mathtt{GenDi}(\mathit{laura}, \mathit{franco})$

Deduzione *(da 1 per eliminazione di ∀)*

4. $\mathtt{MADRE}(\mathit{laura}) \leftrightarrow \mathtt{DONNA}(\mathit{laura}) \wedge \exists y\ \mathtt{GenDi}(\mathit{laura}, y)$

Deduzione *(da 3 per introduzione di ∃)*

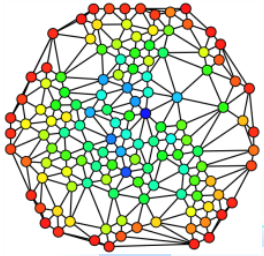
5. $\exists y\ \mathtt{GenDi}(\mathit{laura}, y)$

Deduzione *(da 2 e 5 per introduzione di ∧)*

6. $\mathtt{DONNA}(\mathit{laura}) \wedge \exists y\ \mathtt{GenDi}(\mathit{laura}, y)$

Deduzione *(da 4 e 6 per eliminazione di ↔)*

7. $\mathtt{MADRE}(\mathit{laura})$



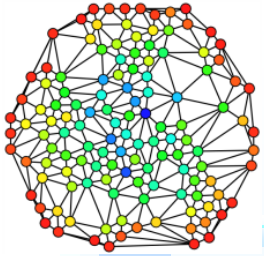
Semi-decidibilità di FOL (1)

Utilizzando FOL si può:

- esprimere conoscenza in modo molto articolato
- eseguire *(in linea di principio)*, in modo automatico, dei ragionamenti complessi

C'è però un problema: in FOL la procedura di deduzione non è una procedura di decisione, ma soltanto di **semi-decisione.** Ciò significa che:

- 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**



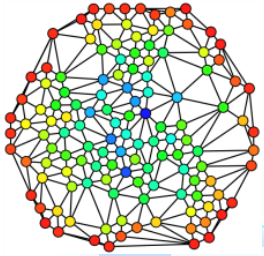
Semi-decidibilità di FOL (2)

La semi-decidibilità di FOL dipende dalla sua notevole espressività:

- più un linguaggio di rappresentazione è espressivo, più sono problematiche le procedure di ragionamento

Molte ricerche nel campo dei linguaggi di rappresentazione della conoscenza hanno l'obiettivo di identificare un sotto-linguaggio di FOL tale che:

- il linguaggio sia comunque abbastanza espressivo per le applicazioni
- la deduzione si basi su una procedura di **decisione**
- tale procedura di decisione abbia complessità computazionale accettabile



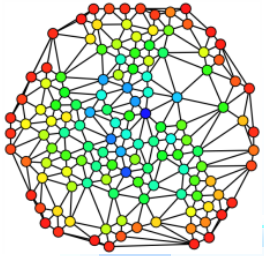
Rappresentazione della Conoscenza e Ontologie

La conoscenza

Linguaggi di rappresentazione

Ragionamento automatico

Sistemi di rappresentazione della conoscenza



Le rappresentazioni

Esempi di Sistemi di Rappresentazione sono:

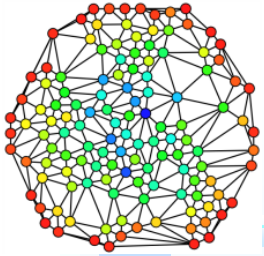
- le basi di dati
- i diagrammi UML
- i documenti XML

Al di là delle ovvie differenze, tutti i sistemi di rappresentazione condividono alcune caratteristiche comuni:

Modello concreto: conoscenza fattuale su un certo frammento di realtà *(es. "Luigi Rossi ha 10 anni")*

Modello concettuale: conoscenza terminologica e nomologica *(es. definire i concetti di persona - avente nome, cognome e altre proprietà - ecc.)*

Metamodello: la specifica degli strumenti formali utilizzabili per definire i primi due *(es. linguaggio FOL, schemi E-R, diagrammi delle classi in UML, ecc)*



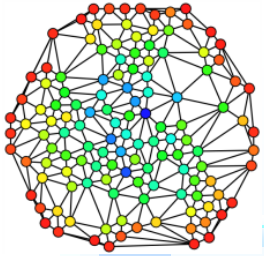
Knowledge Representation Systems

Differenza di fondo tra un **KBS** (***Knowledge Based System***) ed atri sistemi di rappresentazione *(basi di dati, UML)*:

nei campi delle basi di dati e della specifica del software, i **modelli concettuali**

NON fanno parte del prodotto finale

nei **KBS** i **modelli concettuali** **sono parte integrante del prodotto finale** in quanto disponibili e utilizzati a runtime dai sistemi software.



Architettura di un KBS

Base di conoscenze (KB):

Terminological Box (TBox) → modello concettuale *(schemi, classi…)*

Assertion Box (ABox) → modello concreto *(dati, istanze…)*

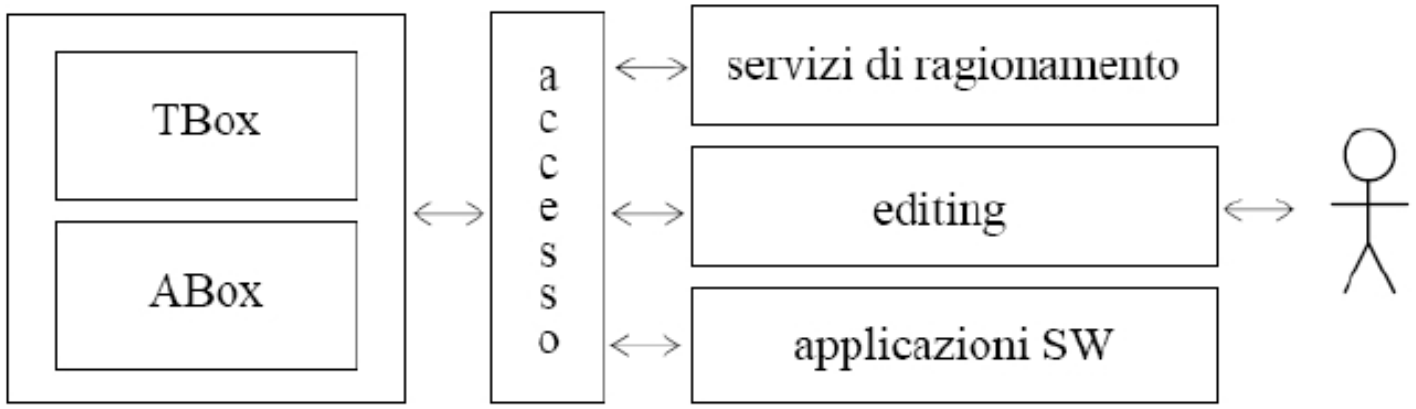
Moduli, Servizi e Applicazioni Software:

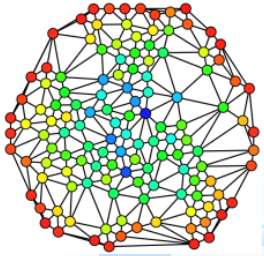
Servizi di ragionamento per dedurre conoscenza partendo dalla KB

Interfaccia di **editing** per gestire *(lato utente)* i contenuti della KB

Applicazioni software specifiche

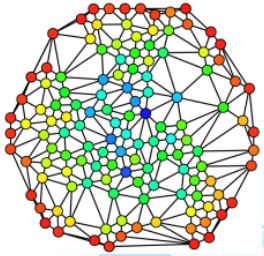
Interfaccia di accesso per la comunicazione tra **KB** e **Applicazioni**





Elementi di Logica Descrittiva

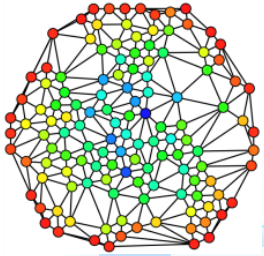
- Termini, equivalenze e sussunzioni
- Ruoli
- Nominali, termini enumerativi e domini concreti
- TBox e ABox



Logiche Descrittive

Un linguaggio descrittivo **L** consiste di una terna di insiemi finiti **(C;R;Ob)**.

- Gli elementi di **C** sono indicati con le lettere A, B, ecc. e sono chiamati *concetti atomici* di **L**;
- gli elementi di **R** sono indicati con le lettere r, s, ecc. e sono detti *ruoli* di **L**
- gli elementi di **Ob** sono indicati con le lettere a, b, ecc. e sono detti (*nomi di*) *oggetti* di **L**.



Logiche Descrittive

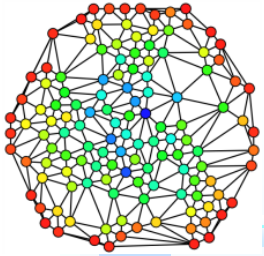
Logica Descrittiva di base : **ALC** ***(Attribute Language with Complement)***

La semantica è definita mediante entità chiamate **Interpretazioni**:

$$\boldsymbol{I} = (\Delta, \boldsymbol{i})$$

Δ: Dominio non vuoto; $\boldsymbol{i}$: funzione di interpretazione

- $\boldsymbol{A} \in \boldsymbol{C} \rightarrow \boldsymbol{A}_{\Delta} \subseteq \Delta$, ($\boldsymbol{A}_{\Delta}$ è un insieme)
- $\boldsymbol{r} \rightarrow \boldsymbol{r}_{\boldsymbol{x}} \subseteq \Delta \times \Delta$, ($\boldsymbol{r}_{\boldsymbol{x}}$ è una relazione binaria)
- $\boldsymbol{Ob} \rightarrow \boldsymbol{a} \in \Delta$. ($\boldsymbol{a}$ è un elemento)



Operatori Logici

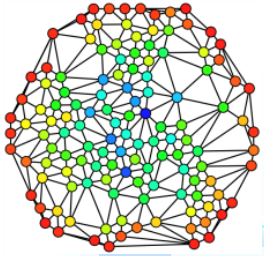
Diverse **Logiche Descrittive** *(DL) derivate da* ***ALC***

- ***S*** → ALC estesa con proprietà di sussunzione/equivalenza e transitività dei ruoli
- ***H*** → gerarchia tra ruoli (es., hasDaughter ⊑ hasChild)
- ***F*** → proprietà funzionali
- ***O*** → possibilità di definire termini per enumerazione di nominali {*a1*,...,*an*}
- ***N*** → restrizioni numeriche di cardinalità (es., ≤2hasChild)
- ***Q*** → restrizione di cardinalità qualificata (es., GEN3F ≡ ≥3GenDi.FEMMINA)
- ***I*** → presenza di ruoli inversi (es., isChildOf ≡ hasChild^{-})

SHIQ (alla base del linguaggio ***DAML+OIL****)*

SHOIN(Dn) *(alla base del linguaggio* ***OWL****, lo standard attualmente sostenuto dal W3C)*

Ogni *DL* si caratterizza per l'utilizzo di un certo numero di ***operatori logici*** scelti da un repertorio di operatori possibili



Termini

Atomici *(indicati spesso con le lettere* **A** *e* **B***)*

DONNA

intuitivamente significa *"DONNA"*

Complessi *(indicati spesso con le lettere* **C** *e* **D***)*

PERSONA ⊓ FEMMINA

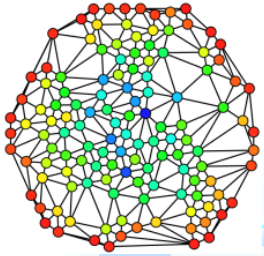
si legge *"PERSONA e FEMMINA"* o *"PERSONA intersezione FEMMINA"*

intuitivamente significa *"persona di genere femminile"*

Spesso chiamati

concetti *(poiché descrivono concetti)*

classi *(poiché denotano insiemi di oggetti della realtà)*



Equivalenze

Equivalenza Terminologica

DONNA ≡ PERSONA ⊓ FEMMINA

intuitivamente significa *"DONNA equivale a PERSONA FEMMINA"*

In generale

C ≡ D

si legge *"C equivale a D"*

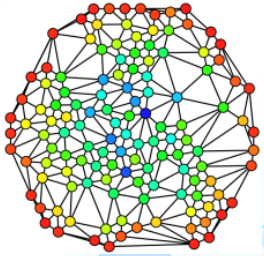
esprime l'equivalenza fra i 2 termini *"C"* e *"D"*

Definizione Terminologica

A α **C** (con "*A*" è atomico)

DONNA ≡ PERSONA ⊓ FEMMINA

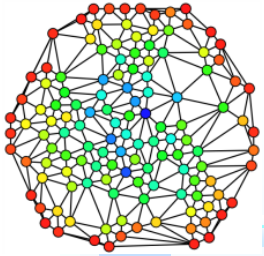
definizione del termine *"DONNA"* a partire dai termini *"PERSONA"* e *"FEMMINA"*



Funzioni di una ontologia

- **Definire** *relazioni* di **equivalenza** e **sussunzione** fra un certo numero di **termini**

- **Assegnare** un *significato non ambiguo* a un certo numero di **termini atomici** (non primitivi) in base al *significato* di **altri termini**



Sussunzioni

Sussunzione Terminologica

RAGAZZA ⊑ DONNA

si legge *"RAGAZZA è sussunto da DONNA"* o *"DONNA sussume RAGAZZA"*

intuitivamente significa *"una ragazza è una donna"*

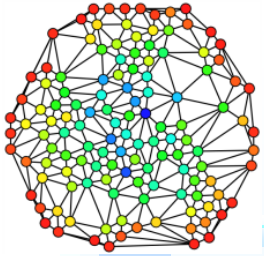
In generale

C ⊑ D

ogni individuo descritto da *"C"* è anche descritto da *"D"*

Simmetria

C ≡ D coincide con la doppia sussunzione **C ⊑ D** e **D ⊑ C**



Enunciati terminologici

Espressioni che esprimono

equivalenze fra **termini**

sussunzioni fra **termini**

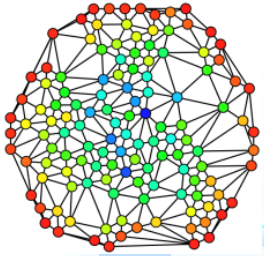
Assioma Terminologico

Enunciato Terminologico assunto come vero

Terminologia o **Ontologia**

insieme finito di **Assiomi Terminologici**

una teoria del primo ordine esprimibile in una *DL*



Negazione e disgiunzione

Gli *"uomini"* sono il ***complemento*** delle *"donne"* rispetto alla totalità delle "persone". Utilizzando l'operatore ¬ di ***complemento*** *(negazione)* è quindi possibile definire:

UOMO ≡ PERSONA ⊓ ¬FEMMINA diventa

[UOMO ≡ PERSONA ⊓ ¬FEMMINA] =

= ∀*x* ([UOMO]*x* ↔ [PERSONA ⊓ ¬FEMMINA]*x*) =

= ∀*x* (UOMO(*x*) ↔PERSONA(*x*) ∧ ~FEMMINA(*x*))

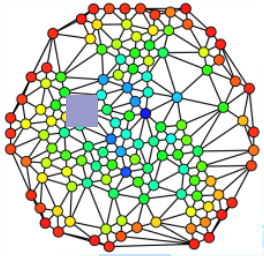
Un altro operatore utile è l'operatore di **unione** *(disgiunzione*, **or** *non esclusivo)*, che ci permette ad esempio di definire gli *"esseri viventi"* come **unione** di *"vegetali"* e *"animali"*:

VIVENTE ≡ VEGETALE ⊔ ANIMALE diventa

[VIVENTE ≡ VEGETALE ⊔ ANIMALE] =

= ∀ *x* ([VIVENTE]*x* ↔ [VEGETALE ⊔ ANIMALE]*x*) =

= ∀ *x* (VIVENTE(*x*) ↔ VEGETALE(*x*) ∨ ANIMALE(*x*))

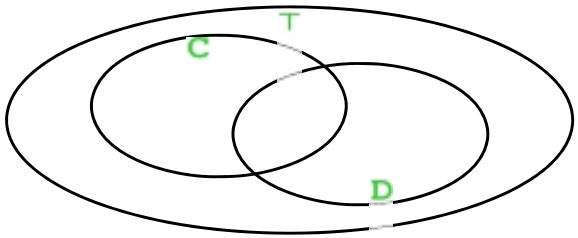


DL vs. Algebra booleana

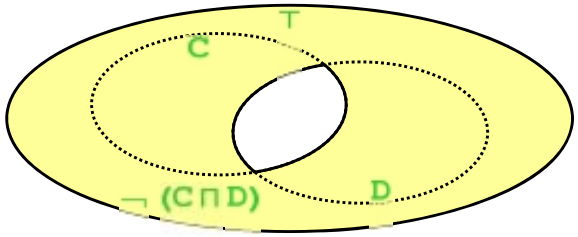
Si noti che gli operatori ¬, ⊓, ⊔, ⊤, ⊥ formano un'algebra booleana:

¬ ⊤ equivale a ⊥

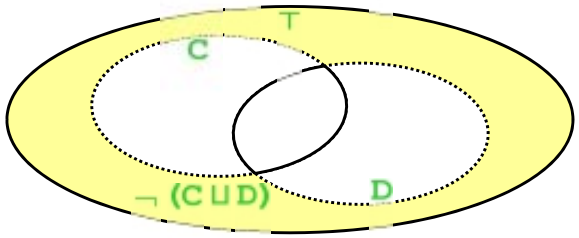
¬ ¬ C equivale a C

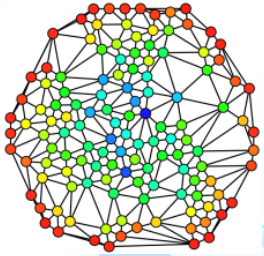


¬ (C ⊓ D) equivale a ¬ C ⊔ ¬ D



¬ (C ⊔ D) equivale a ¬ C ⊓ ¬ D





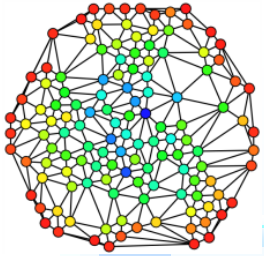
I Ruoli

Oltre ai termini corrispondenti a predicati con ***un*** argomento *(detti, come abbiamo visto,* ***concetti*** *o* ***classi****)*, le *DL* utilizzano termini corrispondenti a **predicati a *due* argomenti**

I predicati a due argomenti esprimono ***relazioni binarie*** fra individui della realtà

Tali **termini** vengono detti **ruoli**

ruoli ≡ ***proprietà*** ≡ ***attributi*** ≡ ***relazioni***



Quantificatore esistenziale

Esempio di enunciato terminologico con un ***ruolo***

MADRE $\sqsubseteq$ $\exists$**GenDi**

intuitivamente significa *"ogni madre è genitore di almeno un individuo"*

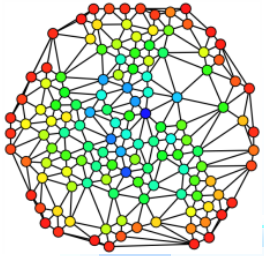
si legge *"MADRE è sussunto dall'INSIEME DEGLI INDIVIDUI che sono GENITORI di QUALCUNO"*

$\exists$**GenDi** = termine complesso formato dal *quantificatore esistenziale* $\exists$ ed il *ruolo* **GenDi**

Traduzione in ***FOL*** *(una formula dotata di un'unica variabile libera* x*, mentre* y *è vincolata da* $\exists$*)*

MADRE $\sqsubseteq$ $\exists$**GenDi** diventa

$$[\mathbf{MADRE} \sqsubseteq \exists \mathbf{GenDi}] = \forall x\, ([\mathbf{MADRE}]_x \rightarrow [\exists \mathbf{GenDi}]_x) =$$
$$= \forall x\, (\mathbf{MADRE}(x) \rightarrow \exists y\, \mathbf{GenDi}(x, y))$$



Quantificatore esistenziale qualificato

Esempio:

$\exists$**GenDi.FEMMINA**

denota l'insieme di *"tutti gli individui"* dell'universo che sono *"genitori"* di almeno un *"individuo"* di sesso *"femminile"*

si legge *"l'INSIEME DEGLI INDIVIDUI che sono GENITORI di almeno una FEMMINA"*

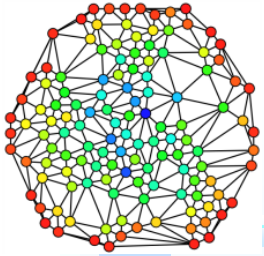
Traduzione:

$\exists$**R.C** diventa

$$[\exists R.C]_x = \exists y\,(R(x,y) \wedge [C]_y)$$

$\exists$**GenDi.FEMMINA** diventa

$$[\exists GenDi.FEMMINA]_x = \exists y\,(GenDi(x,y) \wedge [FEMMINA]_y) =$$
$$= \exists y\,(GenDi(x,y) \wedge FEMMINA(y))$$



Quantificatore universale

Esempio:

$\forall$GenDi.FEMMINA

si legge *"l'insieme degli INDIVIDUI dell'universo che sono GENITORI di sole FEMMINE"*

Traduzione:

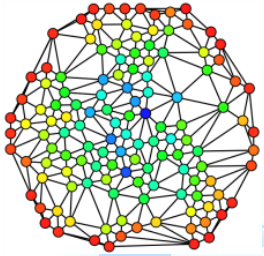
$\forall$R.C diventa

$[\forall R.C]_x = \forall y\,(R(x,y) \rightarrow [C]_y)$

$\forall$GenDi.FEMMINA diventa

$[\forall GenDi.FEMMINA]_x = \forall y\,(GenDi(x,y) \rightarrow [FEMMINA]_y) =$

$= \forall y\,(GenDi(x,y) \rightarrow FEMMINA(y))$



Il ruolo inverso

Esprimiamo il ruolo *"FiglioDi"* partendo dal ruolo *"GenDi"* attraverso la notazione *"GenDi –"*

FiglioDi ≡ **GenDi-**

Esempio

∃ **GenDi-.FEMMINA** diventa

[∃ GenDi-.FEMMINA]$_x$ = ∃ y **(GenDi** (y, x) ∧ **[FEMMINA]**$_y$**)**

∀ **GenDi-.FEMMINA** diventa

[∀ GenDi-.FEMMINA]$_x$ = ∀ y **(GenDi** (y, x) → **[FEMMINA]**$_y$**)**

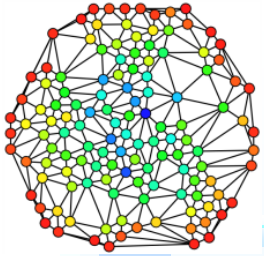
In generale

R esprime la relazione binaria **R** (x, y)

R- esprime la relazione binaria **R** (y, x)

∃ **R-.C** diventa **[∃ R-.C]**$_x$ = ∃ y **(R** (y, x) ∧ **[C]**$_y$**)**

∀ **R-.C** diventa **[∀ R-.C]**$_x$ = ∀ y **(R** (y, x) → **[C]**$_y$**)**



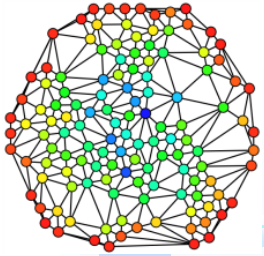
Dominio e codominio (1)

I ruoli, in generale, hanno **senso** solo limitatamente a certi sottoinsiemi dell'universo

Esempio

> `GenDi` mette in relazione fra loro due persone, mentre non ha senso se applicato ad altre categorie inanimate, ad esempio oggetti.

Ad un ruolo `R` si associano quindi due insiemi di individui, detti il ***dominio*** e il ***codominio*** del ruolo, che rappresentano gli insiemi di individui sui cui pensiamo variare le variabili ***x*** e ***y*** nell'espressione `R(x,y)`



Dominio e codominio (2)

Definizione di dominio **D** e il codominio **C** di un ruolo **R**

T $\sqsubseteq \forall$ **R.C** *(definizione del dominio* **D***)* - in FOL $\forall x\, \forall y$ **(R(***x***,** y**)** $\rightarrow$ **[C]**$_y$**)**

T $\sqsubseteq \forall$ **R-.D** *(definizione del codominio* **C***)* - in FOL $\forall x\, \forall y$ **(R(***y***,** x**)** $\rightarrow$ **[D]**$_y$**)**

Esempio

T $\sqsubseteq \forall$ **ProprietarioDi.BENE**

"definizione di ***dominio*** *dato dall'insieme degli INDIVIDUI (PERSONE) che sono PROPRIETARI di* ***soli*** *BENI"*

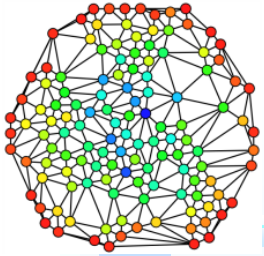
T $\sqsubseteq \forall$ **ProprietarioDi-.PERSONA**

"definizione di ***codominio*** *dato l'insieme degli INDIVIDUI (BENI) che sono POSSEDUTI da* ***sole*** *PERSONE"*

Notazione abbreviata

R : D $\rightarrow$ **C**

ProprietarioDi : PERSONA $\rightarrow$ **BENE**



Vincoli di cardinalità

È possibile esprimere sui ***ruoli***

vincoli di *cardinalità semplice*

$\leq nR$ $\geq nR$

oppure vincoli di *cardinalità qualificata*

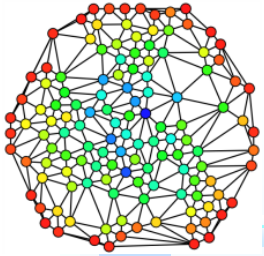
$\leq nR.C$ $\geq nR.C$

Il simbolo **n** rappresenta un intero senza segno

Esempio

$\mathbf{GEN3F} \equiv \geq 3\mathbf{GenDi.FEMMINA}$

"definizione di GEN3F dall'insieme degli individui che sono genitori di almeno tre figlie"



Vincoli di cardinalità (2)

Traduzioni in FOL

- $\leq nR$ diventa

  $[\leq nR]_x = \exists_{\leq n} y\ R(x, y)$

- $\geq nR$ diventa

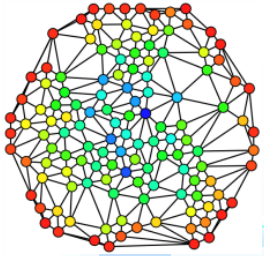
  $[\geq nR]_x = \exists_{\geq n} y\ R(x, y)$

- $\leq nR.C$ diventa

  $[\leq nR.C]_x = \exists_{\leq n} y\ (R(x, y) \wedge [C]_y)$

- $\geq nR.C$ diventa

  $[\geq nR.C]_x = \exists_{\geq n} y\ (R(x, y) \wedge [C]_y)$

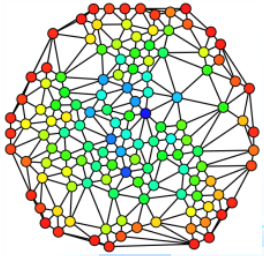


Vincoli di cardinalità (3)

Riprendiamo l'esempio precedente:

GEN3F ≡ ≥3GenDi.FEMMINA diventa:

$$\forall x\ (GEN3F(x) \leftrightarrow \exists y1, \exists y2, \exists y3,$$
$$(\ GenDi(x,y1) \wedge FEMMINA(y1)$$
$$\wedge\ GenDi(x,y2) \wedge FEMMINA(y2)$$
$$\wedge\ GenDi(x,y3) \wedge FEMMINA(y3)$$
$$\wedge\ y1 \neq y2$$
$$\wedge\ y1 \neq y3$$
$$\wedge\ y2 \neq y3\ ).$$



Cardinalità esatta

Definizioni:

$=nR \triangleq \leq nR \sqcap \geq nR$

$=nR.C \triangleq \leq nR.C \sqcap \geq nR.C$

Osservazioni:

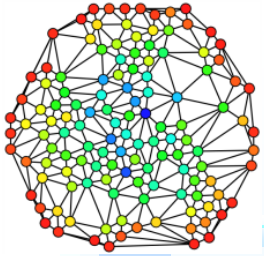
$\geq 1R.C$ equivale a $\exists R.C$

$\leq 0R.C$ equivale a $\neg\exists R.C$

$\geq 0R.C$ equivale a $\top$

"l'insieme di tutti gli individui che hanno una relazione di tipo R con zero o più individui"

indipendentemente dal ruolo R, tutti gli individui del dominio soddisfano questa condizione



Ruoli funzionali (1)

Una ***relazione funzionale*** è una ***relazione binaria*** t.c. ogni elemento del dominio è in relazione con ***al più*** un elemento del codominio.

Un tale ruolo è detto ***ruolo funzionale***

Esempio:

$\texttt{MoglieDi} : \texttt{DONNA} \rightarrow \texttt{UOMO}$

Definizione del Dominio:

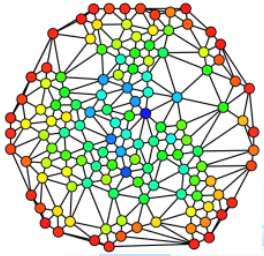
$\top \sqsubseteq \forall\, \texttt{MoglieDi}.\texttt{UOMO}$

$\top \sqsubseteq \forall\, \texttt{MoglieDi}^{-}.\texttt{DONNA}$

Nel nostro ordinamento legale, il **ruolo** $\texttt{MoglieDi}$ è **funzionale** perché ogni donna può avere al più un marito (per volta, s'intende)

$\texttt{DONNA} \sqsubseteq \leq 1\,\texttt{MoglieDi}$

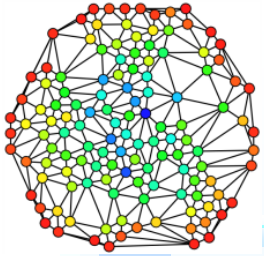
è sufficiente affermare che la classe delle donne coincide con la classe degli individui che hanno ***al più*** *un marito.*



Ruoli funzionali (2)

Si noti che $\texttt{DONNA} \sqsubseteq \leq 1\texttt{MoglieDi}$ non elimina la necessità di definire comunque il dominio di $\texttt{MoglieDi}$ attraverso $\top \sqsubseteq \forall\, \texttt{MoglieDi}.\texttt{UOMO}$

Da sola $\texttt{DONNA} \sqsubseteq \leq 1\texttt{MoglieDi}$ non esclude la possibilità che il dominio di $\texttt{MoglieDi}$ sia più ampio dell'insieme delle donne



Sussunzioni tra ruoli

In molte DL è consentito esprimere sussunzioni ed equivalenze fra ruoli con espressioni della forma:

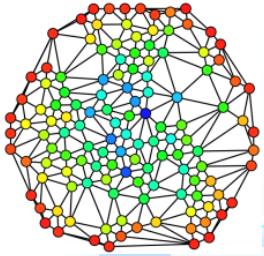
- $\mathtt{R} \sqsubseteq \mathtt{S}$ diventa in FOL $\forall x\, \forall y\, (\mathtt{R}(x, y) \rightarrow \mathtt{S}(x, y))$
- $\mathtt{R} \equiv \mathtt{S}$ diventa in FOL $\forall x\, \forall y\, (\mathtt{R}(x, y) \leftrightarrow \mathtt{S}(x, y))$

Esempio

- $\mathtt{GenitoreDi} \sqsubseteq \mathtt{ParenteDi}$
  - *genitore è un tipo di parentela*
- $\mathtt{FilgioDi} \equiv \mathtt{GenitoreDi}^-$
  - *FiglioDi è l'inverso di GenitoreDi*

Proprietà Simmetrica

- $\mathtt{R} \sqsubseteq \mathtt{R}^-$ diventa in FOL $\forall x\, \forall y\, (\mathtt{R}(x, y) \rightarrow \mathtt{R}(y, x))$
- $\mathtt{FratelloDi} \sqsubseteq \mathtt{FratelloDi}^-$



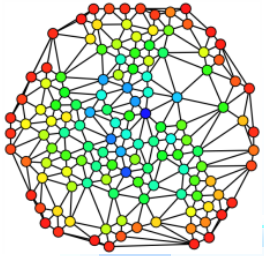
Composizioni di ruoli

In alcune ***DL*** è possibile costruire ***ruoli complessi*** utilizzando l'operatore ∘ di **composizione**.

Dati due ruoli **R** ed **S**:

R ∘ **S** diventa in FOL $\exists z\ (\mathbf{R}(x, z) \wedge \mathbf{S}(z, y))$

La composizione di ruoli è molto utile, ma spesso non viene ammessa perché è problematica per la decidibilità della logica.



Proprietà transitiva

Definizione

$(R \circ R) \sqsubseteq R$ diventa in FOL

$\forall x \forall y \, (\exists z \, (R(x, z) \wedge R(z, y)) \rightarrow R(x, y))$

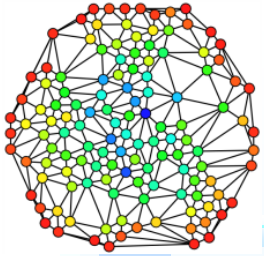
Molte ***DL*** (tra cui $\mathcal{SHOIN}(D_n)$ per OWL) pur non consentendo la composizione di ruoli, prevedono un assioma terminologico per *dichiarare* un ruolo come transitivo

$Tr(R)$

Equivalenza

$\forall (R \circ S).C$ equivale a $\forall R.(\forall S.C)$

$\exists (R \circ S).C$ equivale a $\exists R.(\exists S.C)$

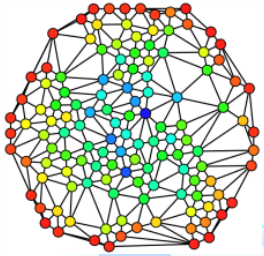


Sistema di rappresentazioni

Un sistema di rappresentazione della conoscenza è costituito di una **TBox** e di una **ABox**

La **TBox** contiene ***assiomi terminologici*** (Classi) e definisce un'ontologia

La **ABox** contiene invece ***conoscenze fattuali*** (Istanze) espresse sotto forma di asserzioni



ABox

Nelle ***DL*** si possono esprimere diversi tipi di conoscenze fattuali

`C(a)` *(C = termine arbitrario, a = nominale)*

`R(a,b)` *(R = ruolo, a,b = nominali)*

Esempio

`MADRE(elisa)` diventa in FOL

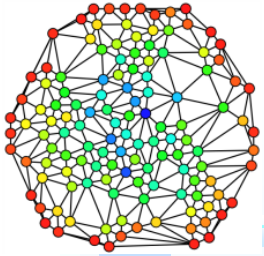
`MADRE(elisa)`

`DONNA` ⊓ ∃`GenDi(laura)` diventa in FOL

`DONNA(laura)` ∧ ∃*y* `GenDi(laura,`*y*`)`

`GenDi(elisa,gino)` diventa in FOL

`GenDi(elisa,gino)`



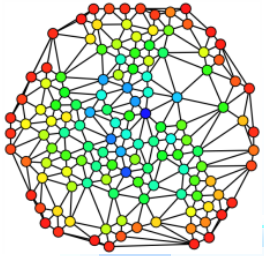
Esempio

TBox

Faculty ⊑ Organization
Professor ⊑ Researcher
Professor ⊑ ∃hasSDS.ScientificDisciplinarySector
Course ⊑ ∃courseTakenAtTime.Interval
Person ⊑ ∃hasCompetence.ConceptSkill
Course ⊑ ∃subject.ConceptSkill

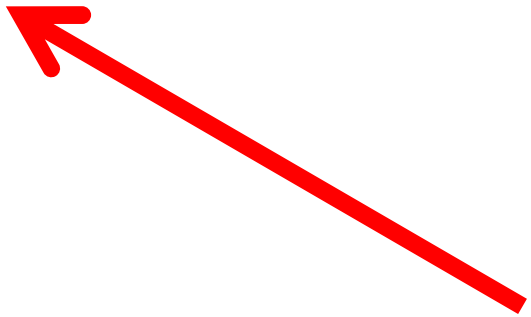
ABox

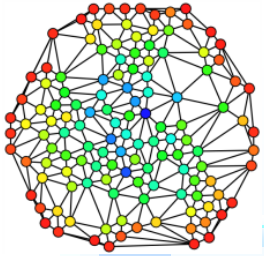
Paolo Nesi: FullProfessor,
Distributed Systems I: Course,
Security Systems: ConceptSkill,
Cloud Computing: ConceptSkill,
(Paolo Nesi, Distribuited Systems I):courseTaken,
(Paolo Nesi, Cloud Computing):hasCompetence,
(Distributed Systems I, Security Systems):subject



Outline

- Rappresentazione della Conoscenza e Ontologie
  - La conoscenza
  - Linguaggi di Rappresentazione
  - Ragionamento Automatico
  - Sistemi di rappresentazione della conoscenza
- OWL: *Ontology Web Language*
  - Introduzione
  - Descrizione di Classi e Assiomi
  - Proprietà
  - Individui e Fatti
  - Servizi di Ragionamento
  - Interrogare la Conoscenza: SPARQL





OWL: Ontology Web Language

Introduzione

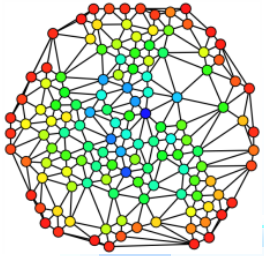
Descrizione di Classi e Assiomi

Le Proprietà

Individui e Fatti

Servizi di Ragionamento

Interrogare la Conoscenza: SPARQL



Il linguaggio OWL (1)

OWL *(Web Ontology Language)* è lo standard proposto dal **W3C** per la definizione di ontologie per il web semantico

http://www.w3.org/TR/owl-ref/

Sviluppato a partire da **DAML+OIL**, il quale

implementa la logica $\mathcal{SHIQ}$

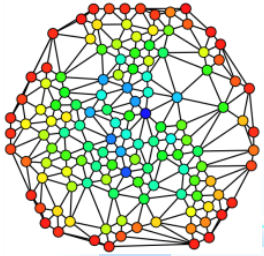
è basato sui linguaggi **OIL** e **DAML-ONT**

Prevede tre livelli di complessità crescente

OWL Lite, semplice da usare e implementare ma scarsamente espressivo

OWL DL, su logica $\mathcal{SHOIN}(D_n)$, abbastanza espressivo, decidibile e dotato di procedure di ragionamento di complessità nota, studiate ed ottimizzate

OWL Full, oltre FOL, molto espressivo ma ***"semi-decidibile"***

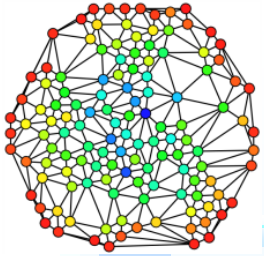


Note di complessità

Il linguaggio delle formule valide della logica del primo ordine non è decidibile, bensì **semi-decidibile**: esiste un algoritmo in grado di valutare la validità di una formula

- Nel caso in cui la formula sia valida l'algoritmo è in grado di terminare restituendo come prova la dimostrazione della sua validità
- In caso contrario, se la formula non è valida, l'algoritmo non è in grado di accorgersene e continua a eseguire calcoli (si dice che diverge), senza mai fornire una risposta

Il linguaggio di tutte le formule universalmente valide della logica del secondo ordine non è **neppure semi-decidibile**.

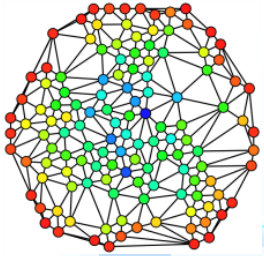


Il linguaggio OWL (2)

Un'ontologia **OWL** si articola in una **TBox** e una **ABox** ambedue rappresentate come **grafi RDF** *(insiemi di **triple RDF**)*

Diversi costrutti di **RDFS** sono direttamente adottati da **OWL**

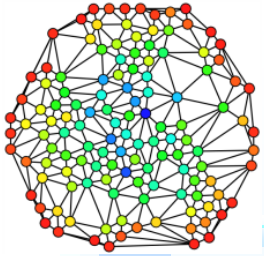
OWL introduce inoltre costrutti propri, non presenti in **RDFS**, comunque rappresentati come **triple RDF**



Terminologia

In **OWL**

- I **Termini** sono denominati ***descrizioni di classi***
- Gli **Operatori** per la definizione di termini sono denominati ***costruttori di classi***
- I **Ruoli** sono denominati ***proprietà***
- Le **Definizioni Terminologiche** della *TBox* sono dette ***assiomi di classe***
- Le **Asserzioni** dell'*ABox* sono detti ***fatti***



OWL: Ontology Web Language

Introduzione

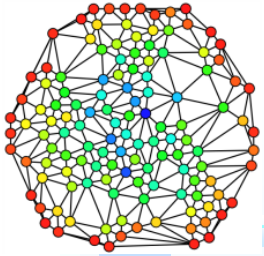
Descrizione di Classi e Assiomi

Le Proprietà

Individui e Fatti

Servizi di Ragionamento

Interrogare la Conoscenza: SPARQL



Identificatore

Ogni ***descrizione di classe*** descrive una **risorsa** di tipo

`owl:Class`

Nel caso più semplice la descrizione consta di un *identificatore della classe (Unique Resource Identifier = URI)*, corrispondente a un **termine atomico** delle ***DL***

Sintassi ***RDF***

```
<owl:Class  rdf:ID="ClassName"/>
<owl:Class  rdf:about="ClassName"/>
```

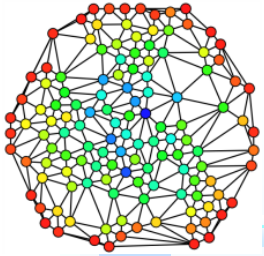
Due identificatori di classi sono già predefiniti in OWL

`owl:Thing` = l'insieme di tutti gli individui ($\top$ per la *classe universale*)

Ogni classe OWL è sottoclasse di **`owl:Thing`**

`owl:Nothing` = l'insieme vuoto ($\bot$ per la *classe vuota*)

La classe **`owl:Nothing`** è una sottoclasse di ogni classe



Enumerazione

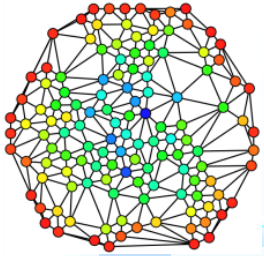
Una classe **A** può essere descritta *dall'enumerazione* di un numero finito di nominali a_1, ..., a_n tramite l'operatore **owl:oneOf**

Sintassi ***DL***

$A \equiv \{a_1, \ldots, a_n\}$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#a1" />
    ...
    <owl:Thing rdf:about="#an" />
  </owl:oneOf>
</owl:Class>
```



Restrizioni di proprietà (1)

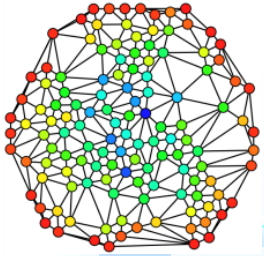
Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **almeno** un individuo della classe **C** tramite l'operatore **owl:someValuesFrom**

Sintassi ***DL***

$A \equiv \exists R.C$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:someValuesFrom rdf:resource="#C" />
  </owl:Restriction>
</owl:Class>
```



Restrizioni di proprietà (2)

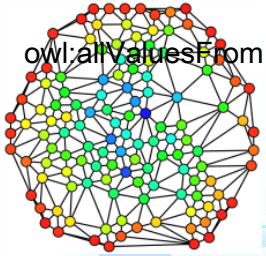
Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **soli individui** di classe **C** tramite l'operatore **owl:allValuesFrom**

Sintassi ***DL***

$A \equiv \forall R.C$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:allValuesFrom rdf:resource="#C" />
  </owl:Restriction>
</owl:Class>
```

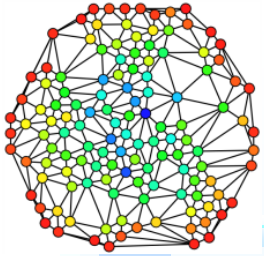


Restrizioni di proprietà (3)

ESEMPI (da W3C OWL Guide: http://www.w3.org/TR/owl-ref/)

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

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



Restrizioni di proprietà (4)

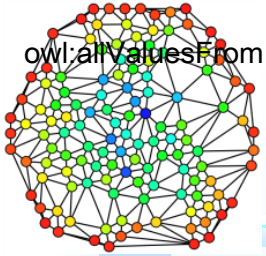
Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** **sul solo individuo** a tramite l'operatore **owl:hasValue**

Sintassi ***DL***

$A \equiv \forall R.\{a\}$

Sintassi ***RDF***

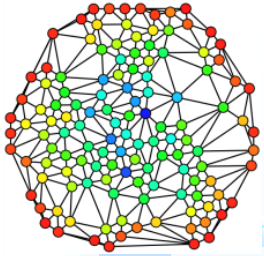
```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:hasValue rdf:resource="#a" />
  </owl:Restriction>
</owl:Class>
```



Restrizioni di proprietà (5)

ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/)

```
<owl:Class rdf:ID="Burgundy">
  ...
  <rdfs:subClassOf rdf:resource="Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTaste" />
      <owl:hasValue rdf:resource="#Dry" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```



Restrizioni di proprietà (6)

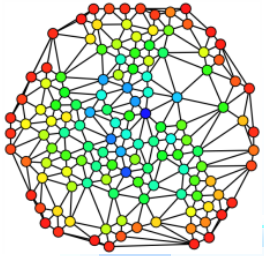
Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **al massimo n individui** tramite l'operatore **owl:maxCardinality**

Sintassi ***DL***

$A \equiv \leq nR$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:maxCardinality rdf:datatype=
      "&xsd;nonNegativeInteger">n</owl:maxCardinality>
  </owl:Restriction>
</owl:Class>
```



Restrizioni di proprietà (7)

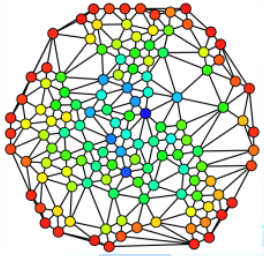
Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **almeno n individui** tramite l'operatore **owl:minCardinality**

Sintassi ***DL***

$A \equiv \geq nR$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:minCardinality rdf:datatype=
      "&xsd;nonNegativeInteger">n</owl:minCardinality>
  </owl:Restriction>
</owl:Class>
```



Restrizioni di proprietà (8)

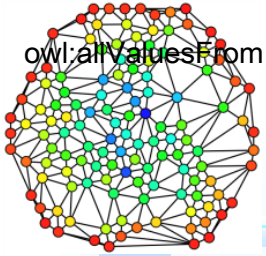
Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **esattamente n individui** tramite l'operatore **owl:cardinality**

Sintassi ***DL***

$A \equiv =nR$

Sintassi ***RDF***

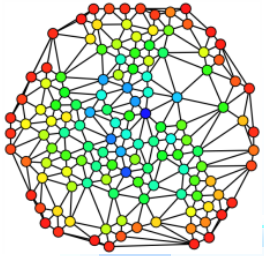
```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:cardinality rdf:datatype=
      "&xsd;nonNegativeInteger">n</owl:cardinality>
  </owl:Restriction>
</owl:Class>
```



Restrizioni di proprietà (9)

ESEMPI (da W3C OWL Guide: http://www.w3.org/TR/owl-ref/)

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxCardinality>
</owl:Restriction>

<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:minCardinality>
</owl:Restriction>

<owl:Restriction>
  <owl:onProperty rdf:resource="#hasIDFiscalCode" />
  <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
```



Intersezione

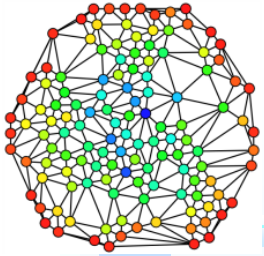
Una classe **A** può essere descritta come *intersezione* di un numero finito di classi **C1**, ..., **Cn** tramite l'operatore **owl:intersectionOf**

Sintassi ***DL***

$A \equiv C_1 \sqcap \ldots \sqcap C_n$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#C1" />
    ...
    <owl:Class rdf:about="#Cn" />
  </owl:intersectionOf>
</owl:Class>
```



Unione

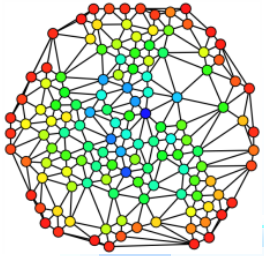
Una classe **A** può essere descritta come *unione* di un numero finito di classi **C1**, ..., **Cn** attraverso l'operatore **owl:unionOf**

Sintassi ***DL***

$A \equiv C_1 \sqcup \ldots \sqcup C_n$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#C1" />
    ...
    <owl:Class rdf:about="#Cn" />
  </owl:unionOf>
</owl:Class>
```



Complemento

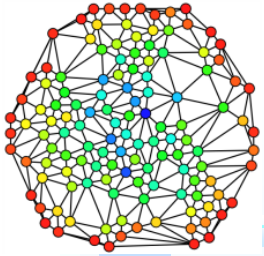
Una classe **A** può essere descritta come *complemento* di un'altra classe **B** attraverso l'operatore **owl:complementOf**

Sintassi ***DL***

$A \equiv \neg B$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:complementOf>
    <owl:Class rdf:about="#B" />
  </owl:complementOf>
</owl:Class>
```



Assiomi di Classe: *Sottoclassi*

Fra due descrizioni di classi **C** e **D** si può definire una relazione di sottoclasse attraverso l'operatore

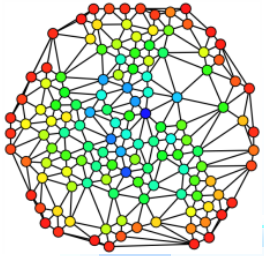
rdfs:subClassOf

Sintassi ***DL***

$C \sqsubseteq D$

Sintassi ***RDF***

```
<owl:Class rdf:about="#C">
  <rdfs:subClassOf rdf:resource="#D" />
</owl:Class>
```



Assiomi di Classe: *Equivalenza*

Fra due descrizioni di classi **C** e **D** si può definire una relazione di equivalenza attraverso l'operatore

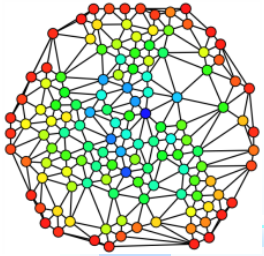
owl:equivalentClass

Sintassi ***DL***

$C \equiv D$

Sintassi ***RDF***

```
<owl:Class rdf:about="#C">
  <owl:equivalentClass rdf:resource="#D" />
</owl:Class>
```



Assiomi di Classe: *Disgiunzione*

Fra due descrizioni di classi **C** e **D** si può dichiarare una relazione di disgiunzione attraverso l'operatore

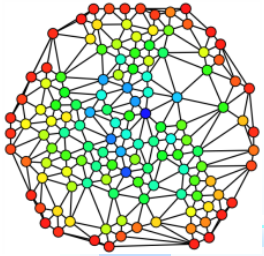
owl:disjointWith

Sintassi ***DL***

$C \sqcap D \equiv \bot$

Sintassi ***RDF***

```
<owl:Class rdf:about="#C">
  <owl:disjointWith rdf:resource="#D" />
</owl:Class>
```



OWL: Ontology Web Language

Introduzione

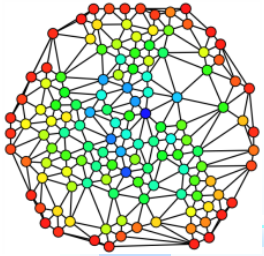
Descrizione di Classi e Assiomi

Le Proprietà

Individui e Fatti

Servizi di Ragionamento

Interrogare la Conoscenza: SPARQL



Proprietà

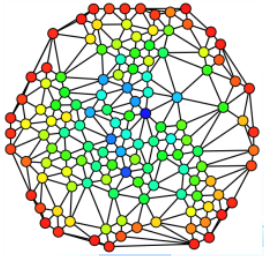
Coerentemente con quanto previsto da *RDFS*, in **OWL** anche le ***proprietà*** *(corrispondenti ai ruoli nelle DL)* possono essere viste come ***particolari classi***

Possono quindi 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`

In OWL le proprietà possono essere risorse di due tipi

- `owl:ObjectProperty` *(proprietà di individui, cioè fra elementi di classi OWL)*
- `owl:DatatypeProperty` *(proprietà di dati appartenenti a tipi di dati RDFS)*



Sottoproprietà

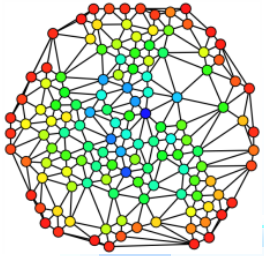
Una proprietà **R** può essere definita come sottoproprietà di un'altra proprietà **S** attraverso l'operatore **rdfs:subPropertyOf**

Sintassi ***DL***

$R \sqsubseteq S$

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <rdfs:subPropertyOf rdf:resource="#S" />
</owl:ObjectProperty>
```



Dominio

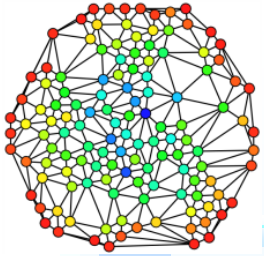
Di una proprietà **R** può essere specificato il dominio attraverso l'operatore **rdfs:domain**

Sintassi ***DL***

$\top \sqsubseteq \forall$ **R.C** *(definizione del dominio* **D***)*

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <rdfs:domain>
    <owl:Class rdf:about="#C" />
  </rdfs:domain>
</owl:ObjectProperty>
```



Codominio

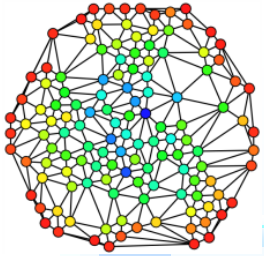
Di una proprietà **R** può essere specificato il codominio attraverso l'operatore **rdfs:range**

Sintassi ***DL***

$\top \sqsubseteq \forall$ **R-.D** *(definizione del codominio* **C***)*

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <rdfs:range>
    <owl:Class rdf:about="#D" />
  </rdfs:range>
</owl:ObjectProperty>
```



Proprietà equivalente

Una proprietà **R** può essere definita come equivalente a un'altra proprietà **S** attraverso l'operatore

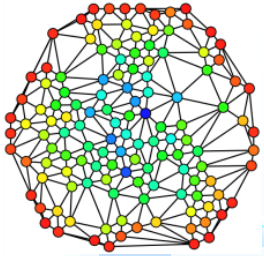
owl:equivalentProperty

Sintassi ***DL***

$R \equiv S$

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <owl:equivalentProperty rdf:resource="#S" />
</owl:ObjectProperty>
```



Proprietà inversa

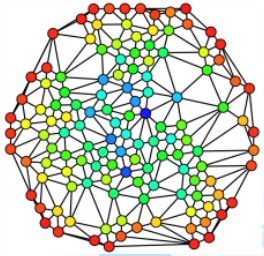
Data una proprietà **R** si può definire la proprietà inversa **S** attraverso l'operatore **owl:inverseOf**

Sintassi ***DL***

$S \equiv R^-$

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="S">
  <owl:inverseOf rdf:resource="#R" />
</owl:ObjectProperty>
```



Funzionalità (1)

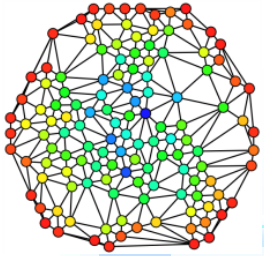
Una proprietà **R** è funzionale se soddisfa il vincolo di cardinalità unitaria globale espresso dall'operatore **owl:FunctionalProperty**, ovvero può assumere un unico valore per ogni istanza del dominio.

Sintassi ***DL***

$\top \sqsubseteq \leq 1 R$

Sintassi ***RDF***

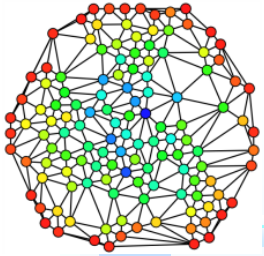
```
<owl:FunctionalProperty rdf:about="#R" />
...
<owl:ObjectProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#C" />
</owl:ObjectProperty>
```



Funzionalità (2)

ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/):

```
<owl:ObjectProperty rdf:ID="hasHusband">
  <rdf:type     rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Woman" />
  <rdfs:range   rdf:resource="#Man" />
</owl:ObjectProperty>
```



Funzionalità inversa

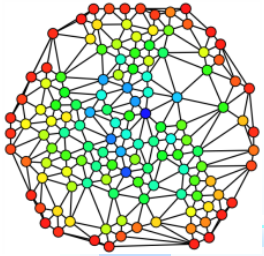
Una proprietà **R** è funzionale inversa se soddisfa il vincolo di cardinalità globale espresso dall'operatore **owl:InverseFunctionalProperty**

Sintassi ***DL***

$\top \sqsubseteq \leq 1 R^-$

Sintassi ***RDF***

```
< owl:InverseFunctionalProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#C" />
</owl:InverseFunctionalProperty>
```



Transitività (1)

In OWL è possibile dichiarare che una proprietà è transitiva tramite l'operatore

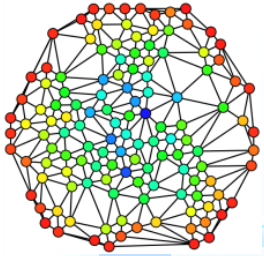
owl:TransitiveProperty

Sintassi ***DL***

$Tr(R)$

Sintassi ***RDF***

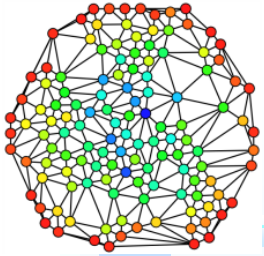
```
<owl:TransitiveProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#D" />
</owl:TransitiveProperty>
```



Transitività (2)

ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/):

```
<owl:ObjectProperty rdf:ID="subRegionOf">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Region"/>
  <rdfs:range  rdf:resource="#Region"/>
</owl:ObjectProperty>
```



Simmetria (1)

In OWL è possibile dichiarare che una proprietà simmetrica tramite l'operatore

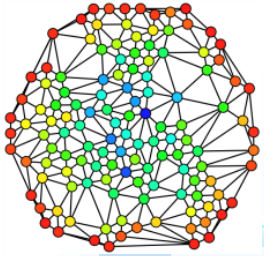
owl:SymmetricProperty

Sintassi ***DL***

$R \sqsubseteq R^-$

Sintassi ***RDF***

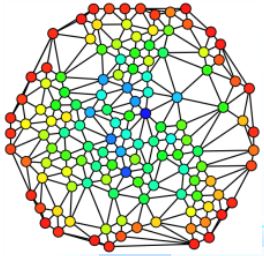
```
<owl:SymmetricProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#D" />
</owl:SymmetricProperty>
```



Simmetria (2)

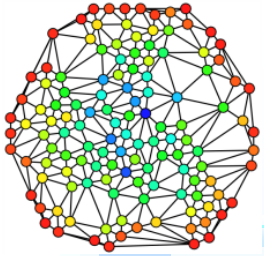
ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/):

```
<owl:SymmetricProperty rdf:ID="friendOf">
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range  rdf:resource="#Human"/>
</owl:SymmetricProperty>
```



Riepilogo

```
<owl:ObjectProperty rdf:about="&UniFI#isCoordinatorOf">
        <rdfs:subPropertyOf rdf:resource="&UniFI#isWorkingFor"/>
        <rdfs:domain rdf:resource="&UniFI#Coordinator"/>
        <owl:inverseOf rdf:resource="&UniFI#hasCoordinator"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="&UniFI#Center"/>
                    <rdf:Description rdf:about="&UniFI#ResearchProject"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
</owl:ObjectProperty>
```



OWL: Ontology Web Language

Introduzione

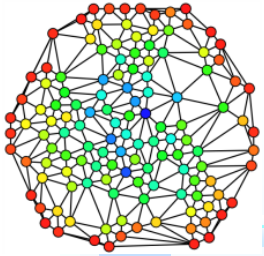
Descrizione di Classi e Assiomi

Le Proprietà

Individui e Fatti

Servizi di Ragionamento

Interrogare la Conoscenza: SPARQL



Appartenenza ad una classe

In OWL è possibile specificare che un individuo **a** appartiene a una classe **C**

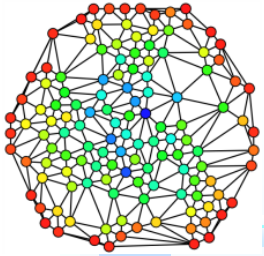
Sintassi ***DL***

```
C(a)
```

Sintassi equivalenti ***RDF:***

```
<C rdf:ID="a">
  ...
</C>
```

```
<C rdf:about="a">
  ...
</C>
```



Valori di proprietà

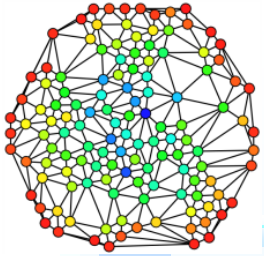
In OWL è possibile specificare che una proprietà **R** di un individuo **a** ha valore **b**

Sintassi ***DL***

```
R(a,b)
```

Sintassi ***RDF***

```
<C rdf:ID="a">
  <R rdf:resource="#b" />
  ...
</C>
```



Identità (1)

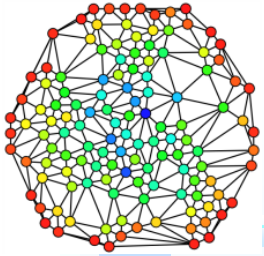
Il linguaggio **OWL** non assume che gli individui abbiano nome unico. Quindi è possibile asserire che due nomi fanno riferimento allo stesso individuo tramite l'operatore **owl:sameAs**

Sintassi ***DL***

```
a = b
```

Sintassi ***RDF***

```
<rdf:Description rdf:about="#a">
  <owl:sameAs rdf:resource="#b" />
</rdf:Description>
```



Identità (2)

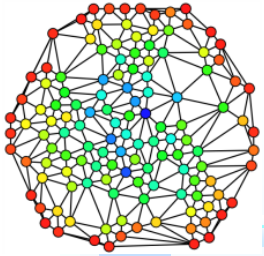
Analogamente è possibile asserire che due nomi fanno riferimento ad individui distinti tramite l'operatore **owl:differentFrom**

Sintassi ***DL***

$a \neq b$

Sintassi ***RDF***

```
<C rdf:ID="a">
  <owl:differentFrom rdf:resource="#b"/>
  …
</C>
```



Identità (3)

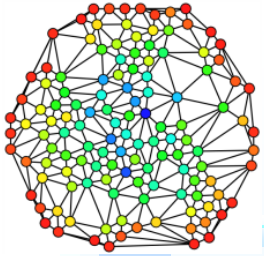
È anche possibile asserire che **n** individui sono tutti distinti fra loro tramite l'operatore **owl:AllDifferent**

Sintassi ***DL***

$a_1 \neq \ldots \neq a_n$

Sintassi ***RDF***

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <C rdf:about="#a1"/>
    ...
    <C rdf:about="#an"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```



OWL: Ontology Web Language

Introduzione

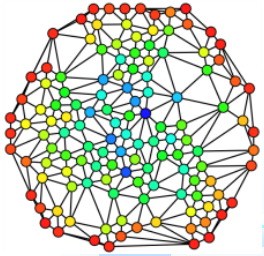
Descrizione di Classi e Assiomi

Le Proprietà

Individui e Fatti

Servizi di Ragionamento

Interrogare la Conoscenza: SPARQL



Reasoning

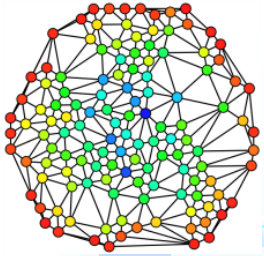
L'aspetto che distingue nettamente una base di conoscenze da una base di dati è la possibilità di condurre ***ragionamenti*** in modo automatico

Poiché una base di conoscenze **KB** è costituita da una *TBox* **T** e da una *ABox* **A** scriveremo in generale

KB = ‹T,A›

Nel contesto della logica, quando si parla di "*ragionamento*" ci si riferisce sempre a ragionamenti di tipo **deduttivo**, o più semplicemente **deduzioni**

In generale, quindi, un ragionamento è un procedimento che porta a verificare se un enunciato **X** *(ad esempio la sussunzione o l'equivalenza di due termini)* è ***conseguenza logica*** di una base di conoscenza



Conseguenza logica (1)

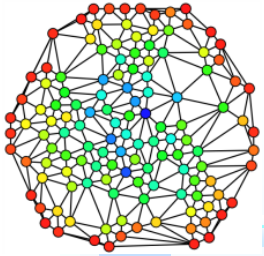
Intuitivamente un enunciato ***X*** è conseguenza logica di una base di conoscenze **KB** quando ***X*** è certamente vero in ogni situazione in cui siano veri gli assiomi terminologici e le asserzioni contenuti in **KB**.

Più precisamente, un enunciato ***X*** è conseguenza logica di una base di conoscenze **KB** quando ***X*** è vero in ogni modello *(nel senso di FOL)* degli assiomi terminologici e delle asserzioni contenuti in **KB**

In tal caso scriviamo

KB $\models$ ***X***

KB implica logicamente ***X*** *(**X** è conseguenza logica di **KB**)*



Conseguenza logica (2)

Consideriamo ad esempio la *TBox* **T** contenente le definizioni seguenti:

T1. $\mathtt{GENITORE} \equiv \mathtt{PERSONA} \sqcap \exists \mathtt{GenDi}$

T2. $\mathtt{GenDi} : \mathtt{PERSONA} \rightarrow \mathtt{PERSONA}$

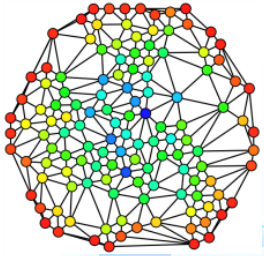
T3. $\mathtt{DONNA} \equiv \mathtt{PERSONA} \sqcap \mathtt{FEMMINA}$

T4. $\mathtt{UOMO} \equiv \mathtt{PERSONA} \sqcap \neg\mathtt{FEMMINA}$

T5. $\mathtt{MADRE} \equiv \mathtt{GENITORE} \sqcap \mathtt{FEMMINA}$

T6. $\mathtt{PADRE} \equiv \mathtt{GENITORE} \sqcap \neg\mathtt{FEMMINA}$

Il contenuto di **T** implica logicamente che certi enunciati, pur non essendo contenuti esplicitamente in **T**, sono necessariamente veri sotto l'ipotesi che sia vero il contenuto di **T**. Ad esempio:



Conseguenza logica (3)

Ogni madre è una persona nonché una donna

$\mathtt{MADRE} \sqsubseteq \mathtt{PERSONA}$

$\mathtt{MADRE} \sqsubseteq \mathtt{DONNA}$

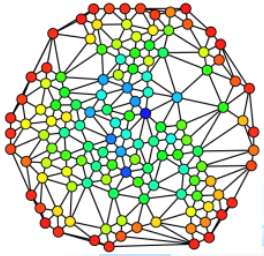
Ogni padre è una persona nonché un uomo

$\mathtt{PADRE} \sqsubseteq \mathtt{PERSONA}$

$\mathtt{PADRE} \sqsubseteq \mathtt{UOMO}$

La classe delle madri e la classe dei padri sono disgiunte

$\mathtt{MADRE} \sqcap \mathtt{PADRE} \equiv \bot$



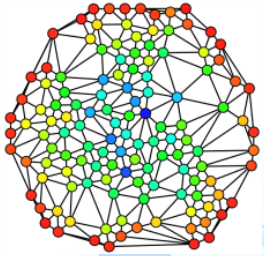
Conseguenza logica (4)

Per segnalare che questi enunciati sono conseguenze logiche di **T** scriviamo ad esempio

$$\mathbf{T} \models \texttt{MADRE} \sqsubseteq \texttt{PERSONA}$$

Altri enunciati, invece, non sono conseguenza logica di **T**. Ad esempio dalla *TBox* precedente non segue logicamente che una persona abbia almeno due genitori. Per esprimere questo fatto scriveremo:

$$\mathbf{T} \not\models \texttt{PERSONA} \sqsubseteq \texttt{=2GenDi-}$$



Tipi di ragionamento

Compito di *ragionamento (reasoning task)*

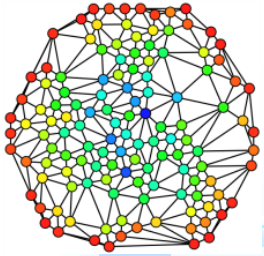
è caratterizzato dal tipo di enunciati che si desidera dedurre da una base di conoscenze

Procedura di *ragionamento*

l'algoritmo che consente la deduzione degli enunciati

Servizio di *ragionamento*

un servizio effettivamente implementato da uno strumento e messo a disposizione delle applicazioni che accedono alla base di conoscenze.



Riduzione alla sussunzione

Si vede facilmente che i compiti di ragionamento fondamentali per le *TBox* possono essere ridotti alla sola sussunzione

Equivalenza

$\mathbf{T} \models C \equiv D$ equivale a $\mathbf{T} \models C \sqsubseteq D$ e $\mathbf{T} \models D \sqsubseteq C$

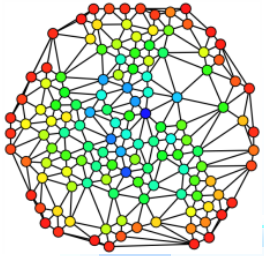
Soddisfacibilità

$\mathbf{T} \not\models C \sqsubseteq \bot$

Disgiunzione

$\mathbf{T} \models C \sqcap D \sqsubseteq \bot$

Questa è la strada che si segue per implementare i servizi di ragionamento per le ***DL poco espressive***



Riduzione alla soddisfacibilità

I compiti di ragionamento fondamentali per le *TBox* possono anche essere ridotti alla sola soddisfacibilità

Sussunzione $\mathbf{T} \models C \sqsubseteq D$

$\mathbf{T} \models C \sqcap \neg D$ **è insoddisfacibile**

Equivalenza $\mathbf{T} \models C \equiv D$

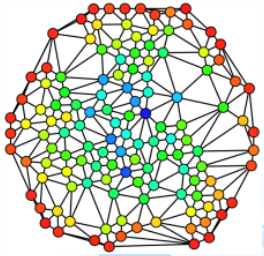
$\mathbf{T} \models C \sqcap \neg D$ **è insoddisfacibile** and

$\mathbf{T} \models \neg C \sqcap D$ **è insoddisfacibile**

Disgiunzione

$\mathbf{T} \models C \sqcap D$ **è insoddisfacibile**

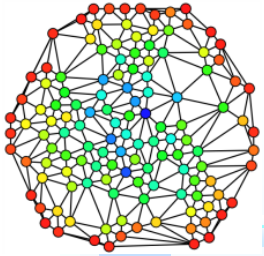
Questa è la strada che si segue per implementare i servizi di ragionamento per le ***DL molto espressive***, es. $\mathcal{SHOIN}(D_n)$



La procedura SAT

Per le ***DL decidibili*** – come $\mathcal{SHOIN}(D_n)$ – si può formulare una procedura che prende in ingresso una *TBox* arbitraria **T** e un termine arbitrario **C** e, in un numero finito di passi, stabilisce se **C** è o non è soddisfacibile *(tenendo conto ovviamente delle definizioni terminologiche di* **T***)*

Nelle sue versioni più diffuse questa procedura, che chiameremo **SAT**, è basata sul cosiddetto metodo dei tableaux da tempo studiato e applicato nell'ambito di ***FOL***



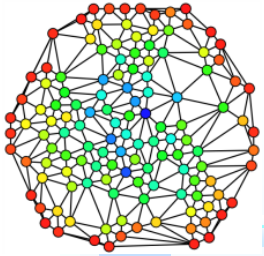
Compiti di ragionamento *(ABox)*

Ci occuperemo ora dei servizi di ragionamento che coinvolgono non soltanto assiomi terminologici della *TBox*, ma anche ***asserzioni*** dell'*ABox*.

Come abbiamo già notato le ***asserzioni*** contenute in un'*ABox* possono essere *basate su termini* o *basate su ruoli*; ovvero, le ***asserzioni*** possono assumere una delle due forme seguenti:

`C(a)` *(*`C` *termine arbitrario;* `a` *nominale)*

`R(a,b)` *(*`R` *ruolo;* `a`, `b` *nominali)*



Compiti di ragionamento *(ABox)*

Instance check

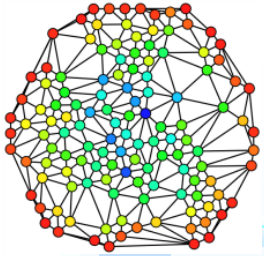
dati una *TBox* **T**, una *ABox* **A**, un termine arbitrario **C** e un nominale **a**, stabilire se si ha $\mathbf{T},\mathbf{A} \models \mathbf{C}(\mathbf{a})$

Retrieval

dati una *TBox* **T**, una *ABox* **A** e un termine arbitrario **C**, fra tutti i nominali presenti nella base di conoscenza trovare tutti i nominali $\mathbf{a}_1, \ldots, \mathbf{a}_n$ tali che $\mathbf{T},\mathbf{A} \models \mathbf{C}(\mathbf{a}_k)$

Realizzazione

dati una *TBox* **T**, una *ABox* **A**, un insieme di termini arbitrari $\{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ e un nominale **a**, determinare gli ***m*** termini $\{\mathbf{C}_{i1}, \ldots, \mathbf{C}_{im}\}$ ***più specifici*** *(sussunzione)* in $\{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ per cui si ha $\mathbf{T},\mathbf{A} \models \mathbf{C}_k(\mathbf{a})$



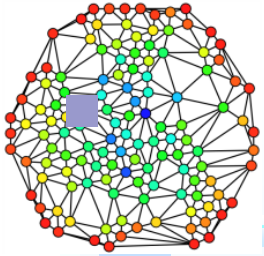
Riduzione alla soddisfacibilità

Un compito di ***instance check*** può essere ridotto a un problema di *Soddisfacibilità* (un termine arbitrario `C` è *soddisfacibile* se esiste almeno un modello di **T**,**A** in cui non è vuoto l'insieme degli individui che soddisfano `C`, ovvero se `∃ a` t.c. **T**,**A** ⊨ `C(a)`)

Un compito di ***retrieval*** può essere ridotto a un compito di *instance check* per ciascun nominale presente nella base di conoscenza

Un compito di ***realizzazione*** può essere ridotto a una serie di compiti di *instance check* e a una serie di compiti di *sussunzione*

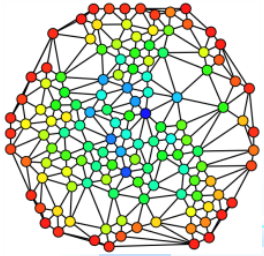
In linea di principio, tutti i compiti di ragionamento che abbiamo esaminato possono essere ridotti a problemi di *soddisfacibilità*



Invocazione dei servizi

In particolare esprimeremo le invocazioni dei servizi di ragionamento presentati nel modo seguente:

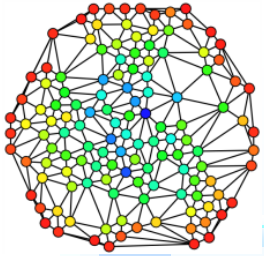
Sussunzione	`?- C ⊑ D`	→ `yes/no`
Equivalenza	`?- C ≡ D`	→ `yes/no`
Soddisfacibilità	`?- C`	→ `yes/no`
Disgiunzione	`?- C ⊓ D ⊑ ⊥`	→ `yes/no`
Instance check	`?- C(a)`	→ `yes/no`
Retrieval	`?- C(*)`	→ $\{a_1, \ldots, a_n\}$
Realizzazione	`?- a:`$C_1, \ldots, C_n$	→ $\{C_{i1}, \ldots, C_{im}\}$



Esempio (1)

Definiamo la *TBox* **T** seguente:

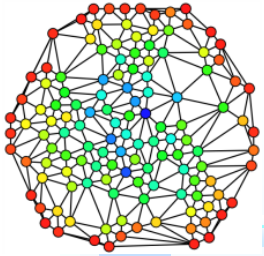
T1. $\mathtt{GENITORE} \equiv \mathtt{PERSONA} \sqcap \exists \mathtt{GenDi}$
T2. $\mathtt{GenDi}: \mathtt{PERSONA} \rightarrow \mathtt{PERSONA},$
T3. $\mathtt{DONNA} \equiv \mathtt{PERSONA} \sqcap \mathtt{FEMMINA}$
T4. $\mathtt{UOMO} \equiv \mathtt{PERSONA} \sqcap \neg\mathtt{FEMMINA}$
T5. $\mathtt{MADRE} \equiv \mathtt{GENITORE} \sqcap \mathtt{FEMMINA}$
T6. $\mathtt{PADRE} \equiv \mathtt{GENITORE} \sqcap \neg\mathtt{FEMMINA}$
T7. $\mathtt{STATO} \equiv \{\mathtt{au},\mathtt{ch},\mathtt{de},\mathtt{es},\mathtt{fr},\mathtt{it},\mathtt{uk}\},$
T8. $\mathtt{CittDi}: \mathtt{PERSONA} \rightarrow \mathtt{STATO},$
T9. $\mathtt{ITAL} \equiv \mathtt{PERSONA} \sqcap \exists \mathtt{CittDi}.\{\mathtt{it}\},$
T10. $\mathtt{BRIT} \equiv \mathtt{PERSONA} \sqcap \exists \mathtt{CittDi}.\{\mathtt{uk}\}.$



Esempio (2)

Definiamo l'*ABox* **A** seguente:

```
A1. DONNA(anna)
A2. DONNA(cecilia)
A3. UOMO(bob)
A4. GenDi(anna,cecilia)
A5. GenDi(bob,cecilia)
A6. CittDi(anna,it)
A7. CittDi(bob,uk)
A8. CittDi(cecilia,it)
A9. CittDi(cecilia,uk)
```



Esempio (3)

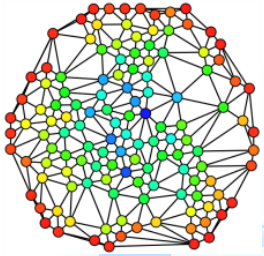
Instance check

Dati il termine **FEMMINA ⊓ GenDi** e il nominale **anna**

si ha:

?- (FEMMINA ⊓ ∃GenDi)(anna) → yes

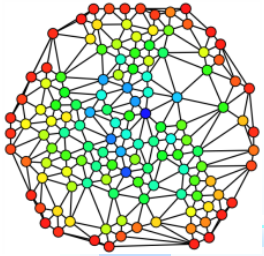
T,A ⊨ **FEMMINA ⊓ ∃GenDi(anna)**



Esempio (3)

ABox **A**:

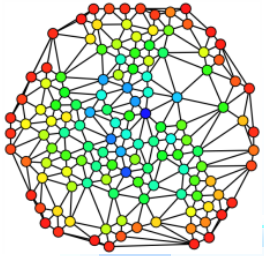
```
A1. DONNA(anna)
A2. DONNA(cecilia)
A3. UOMO(bob)
A4. GenDi(anna,cecilia)
A5. GenDi(bob,cecilia)
A6. CittDi(anna,it)
A7. CittDi(bob,uk)
A8. CittDi(cecilia,it)
A9. CittDi(cecilia,uk)
```



Esempio (3)

TBox **T**:

T1. GENITORE ≡ PERSONA ⊓ ∃GenDi
T2. GenDi: PERSONA → PERSONA,
T3. DONNA ≡ PERSONA ⊓ FEMMINA
T4. UOMO ≡ PERSONA ⊓ ¬FEMMINA
T5. MADRE ≡ GENITORE ⊓ FEMMINA
T6. PADRE ≡ GENITORE ⊓ ¬FEMMINA
T7. STATO ≡ {au,ch,de,es,fr,it,uk},
T8. CittDi: PERSONA → STATO,
T9. ITAL ≡ PERSONA ⊓ ∃CittDi.{it},
T10. BRIT ≡ PERSONA ⊓ ∃CittDi.{uk}.



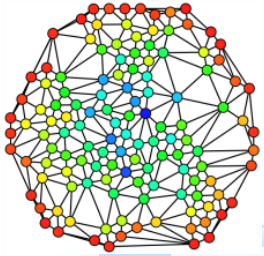
Esempio (4)

Retrieval

Dato il termine **GENITORE** si ha:

?- GENITORE → {anna, bob}

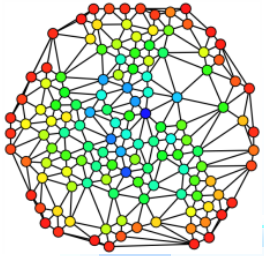
T,A ⊨ GENITORE(anna) **T,A ⊨ GENITORE(bob)**



Esempio (4)

TBox **T**:

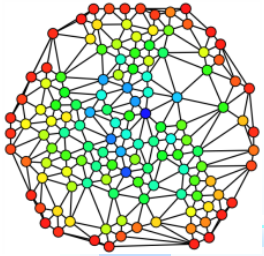
T1. GENITORE ≡ PERSONA ⊓ ∃GenDi
T2. GenDi: PERSONA ⟶ PERSONA,
T3. DONNA ≡ PERSONA ⊓ FEMMINA
T4. UOMO ≡ PERSONA ⊓ ¬FEMMINA
T5. MADRE ≡ GENITORE ⊓ FEMMINA
T6. PADRE ≡ GENITORE ⊓ ¬FEMMINA
T7. STATO ≡ {au,ch,de,es,fr,it,uk},
T8. CittDi: PERSONA ⟶ STATO,
T9. ITAL ≡ PERSONA ⊓ ∃CittDi.{it},
T10. BRIT ≡ PERSONA ⊓ ∃CittDi.{uk}.



Esempio (4)

ABox **A**:

```
A1. DONNA(anna)
A2. DONNA(cecilia)
A3. UOMO(bob)
A4. GenDi(anna,cecilia)
A5. GenDi(bob,cecilia)
A6. CittDi(anna,it)
A7. CittDi(bob,uk)
A8. CittDi(cecilia,it)
A9. CittDi(cecilia,uk)
```



OWL: Ontology Web Language

Introduzione

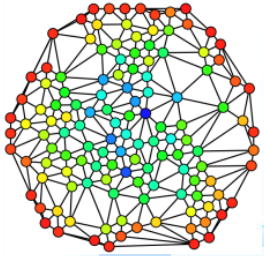
Descrizione di Classi e Assiomi

Le Proprietà

Individui e Fatti

Servizi di Ragionamento

Interrogare la Conoscenza: SPARQL



Interrogare la conoscenza (1)

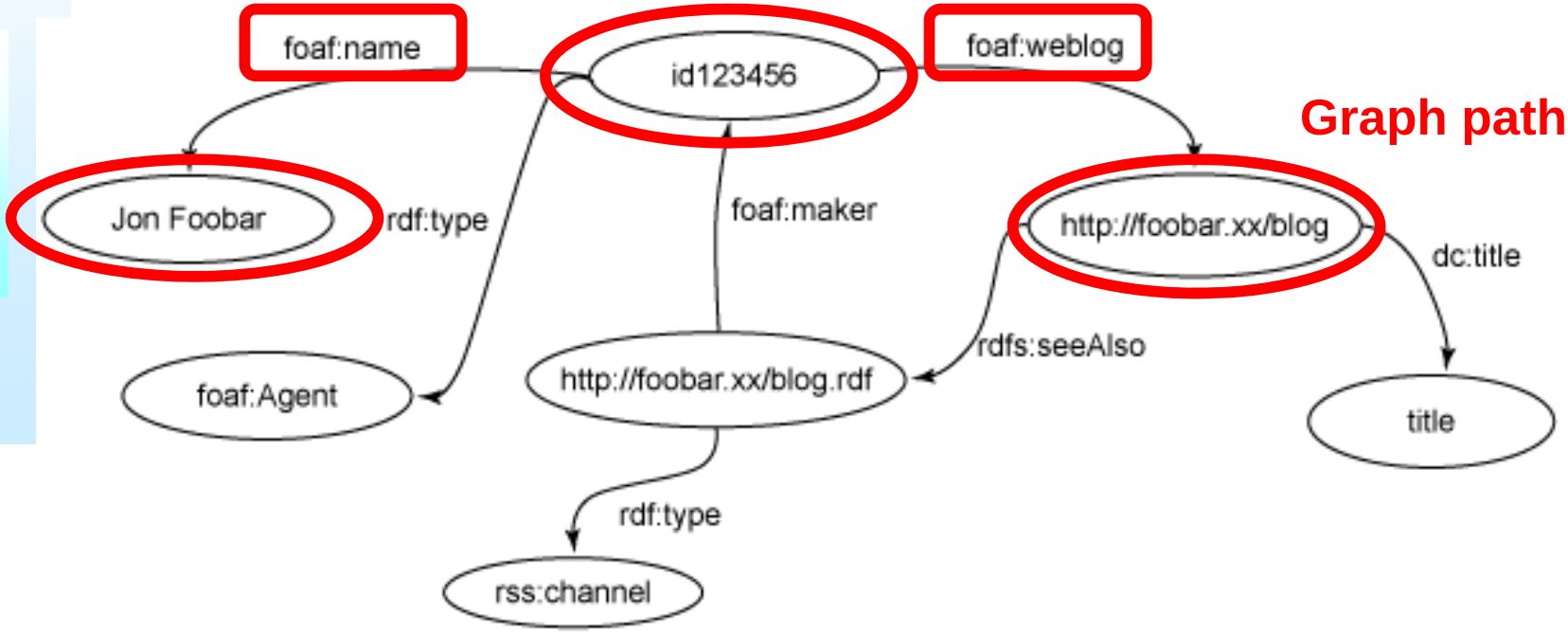
SPARQL - Protocol and RDF Query Language

A SPARQL query comprises, in order:

- *Prefix declarations*, for abbreviating URIs
- *Dataset definition*, stating what RDF graph(s) are being queried
- A *result clause*, identifying what information to return from the query
- The *query pattern*, specifying what to query for in the underlying dataset
- *Query modifiers*, slicing, ordering, and otherwise rearranging query results

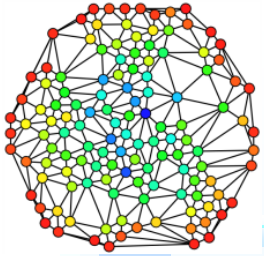
```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...
# result clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

Interrogare la conoscenza (2)



Trova l'url del blog creato dalla persona "Jon Foobar"

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?url
FROM <bloggers.rdf>
WHERE { ?contributor foaf:name "Jon Foobar" . ?contributor foaf:weblog ?url . }
```

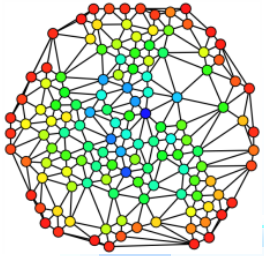


Interrogare la conoscenza (3)

Built-in SPARQL

- *Logical:* `!`, `&&`, `||`
- *Math:* `+`, `-`, `*`, `/`
- *Comparison:* `=`, `!=`, `>`, `<`, ...
- *SPARQL tests:* `isURI`, `isBlank`, `isLiteral`, `bound`
- *SPARQL accessors:* `str`, `lang`, `datatype`
- *Other:* `sameTerm`, `langMatches`, `regex`

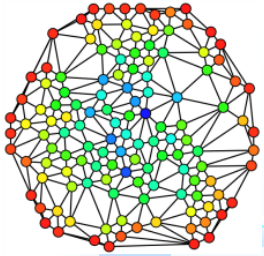
E' possibile effettuare l'unione di più graph paths tramite la clausola UNION



Esempio di query SPARQL (1)

Find me all landlocked countries with a population greater than 15 million.

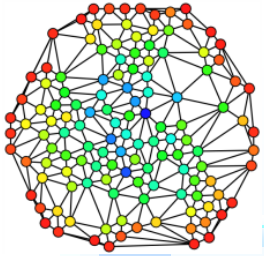
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .

FILTER (?population > 15000000) .
}
```



Esempio di query SPARQL (2)

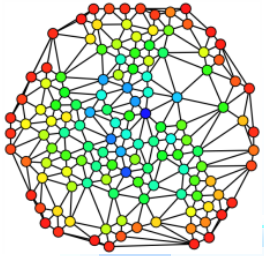
Retrieve the second page of names and emails of people in Tim Berners-Lee's FOAF file, given that each page has 10 people.

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    ?person foaf:name ?name .
    OPTIONAL { ?person foaf:mbox ?email }
} ORDER BY ?name LIMIT 10 OFFSET 10
```



Storing e Indexing dei dati (1)

- **OpenRDF Sesame** (http://www.openrdf.org/) è un framework standard per processare dati RDF (operazioni di parsing, storing, inferenza e possibilità di interrogazione tramite query). Le sue API si possono interfecciare con tutte le principali soluzioni commerciali RDF.
- Può essere utilizzato su una grande varietà di datastore (database relazionali, db in memoria o su filesystem, vocabolari, tassonomie, ontologie, etc.), e offre un ampio set di strumenti per lo sviluppo.
- Supporta il linguaggio SPARQL e offre un accesso trasparente a repository RDF remoti.
- **OWLIM** è una famiglia di repository semantici (o RDF Database Management Systems), con le seguenti caratteristiche:
  - Motore RDF nativo (implementato in Java)
  - Compatibile con OpenRDF Sesame
  - Supporto alla semantica di RDFS, OWL ecc…
  - Ottima scalabilità e performance, anche nella gestione di big data.
  - OWLIM è utilizzato in numerosi progetti di ricerca e strumenti software.

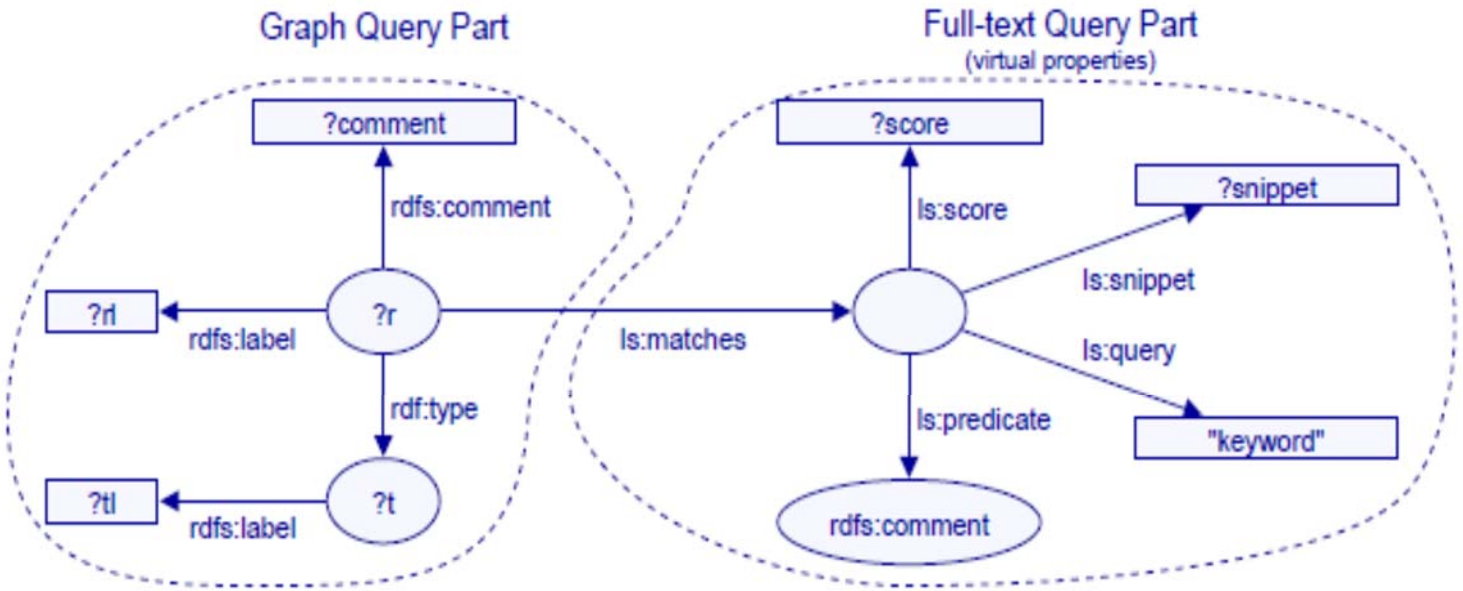


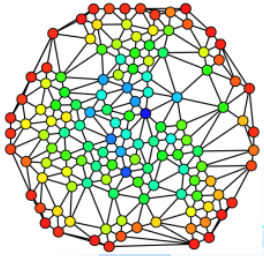
Storing e Indexing dei dati (2)

- ***LuenceSail*** è un sistema proposto dall'università di Hannover, che combina i servizi per l'indicizzazione sintattica offerti da Lucene, con il framework per la memorizzazione di documenti RDF Sesame

- Il sistema è indipendente dal particolare tipo di storage RDF ed estende lo standard SPARQL, con le caratteristiche tipiche delle ricerche full text e delle metodologie fuzzy

- Al fine di estendere lo standard SPARQL vengono utilizzate le cosiddette *virtual properties,* che aggiungono ai normali risultati di una query semantica alcune informazioni tipiche di una ricerca full-text o fuzzy



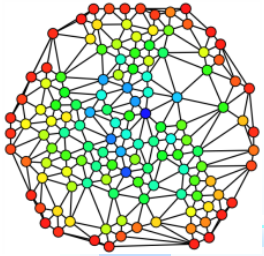


Storing e Indexing dei dati (3)

Esempio: query per sapere tutti i corsi che hanno una data competenza **skillName**

```
SELECT DISTINCT ?uri ?name ?type WHERE {
?uri a uni:Course.
?uri rdfs:label ?name.
?c search:matches ?match.
?uri skos:subject ?c.
?match search:query \"" + skillName + "~" + fuzzyDist + "\";
search:property rdfs:label;
search:score ?score;
FILTER(?score > 0.9)}
ORDER BY ?score ?name
```

Fuzzy con score 0.9



Link utili e Riferimenti

- https://www.mat.unical.it/informatica/Gestione%20della%20Conoscenza?action=AttachFile&do=view&target=GC060704101.pdf
- http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
- http://www.w3.org/TR/rdf-sparql-query/