ANNO ACCADEMICO 2025/2026

Modellazione Concettuale per il Web Semantico

Teoria

Altair's Notes

UNIVERSITÀ DI TORINO

DIPARTIMENTO DI INFORMATICA

Premessa

Licenza

 Sono basati sulle lezioni del corso "Modellazione Concettuale per il Web Semantico" (2020-2021[1]) dell'università di Torino: prof. Damiano. Il tutto è integrato con le slides della versione 2024-2025 del corso.

Formato utilizzato

Box di "Concetto sbagliato":

Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contente il concetto giusto.

Box di "Corollario":

Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

Definizione 0.0.1: Nome delle definizione

Testo della definizione.

Box di "Domanda":

Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

[1] Perché la prof pensa che non registrare sia una cosa tanto furba e io non ho ancora sviluppato il dono dell'ubiquità per seguire due lezioni contemporaneamente.

Box di "Esempio":

Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

Box di "Note":

Note:-

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

Box di "Osservazioni":

Osservazioni 0.0.1

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.

1 Web of Data: I Linguaggi

1.1 Introduzione

1.1.1 Perché Studiare il Semantic Web e Linked Data?

- Gli approcci quantitativi non sono sufficienti per domini complessi:
  - Non è possibile apprendere il comportamento giusto per ogni contesto.
  - Reattività e ragionamento.
- L'ambito della conoscenza è intrinsecamente basato su modelli:
  - Arte, media, tecnologie, etc.
- Interoperabilità dei dati:
  - Conoscenza esperta per stabilire i *mapping*.
  - Utilizzo di standard.
- Ruolo nel ragionamento in molte applicazioni:
  - Elaborazione del linguaggio naturale.
  - Question answering.
  - Chatbots.

1.1.2 Obiettivi del Corso

- Imparare a rappresentare un dominio di conoscenza con i linguaggi del Web Semantico (RDF e OWL), che permettono di implementare ontologie computazionali.
- Utilizzare strumenti di ragionamento automatico per realizzare inferenze sulla conoscenza formalizzata nelle ontologie computazionali.
- Interrogare basi di conoscenza in cui i dati sono rappresentati in un formato semantico utilizzando il linguaggio SPARQL.
- Rendere interoperabili rappresentazioni diverse (ontologie, basi di dati, fogli di calcolo) utilizzando strumenti di mapping.

1.2 Dalle Reti Semantiche alle Ontologie

1.2.1 I Limiti della Logica Classica

A partire dagli anni 60 si è sviluppata una branca dell'intelligenza artificiale specificamente orientata alla rappresentazione della conoscenza. Questo rappresenta un tentativo di superare la logica classica allora utilizzata. Il motivo è che la logica classica ha alcuni limiti importanti:

- È caratterizzata da *inadeguatezza espressiva*.
- È *monotòna*.
- Presenta svantaggi dal punto di vista *computazionale*.

Definizione 1.2.1: Inadeguatezza Espressiva

Alcuni aspetti dell'inadeguatezza della logica classica possono essere attribuite a differenze con i sistemi cognitivi.

Osservazioni 1.2.1

- La logica classica è un *formalismo piatto*:
  - Tutte le affermazioni si collocano sullo stesso piano.
  - Esprime conoscenza di carattere generale e immutabile.
- Le procedure di dimostrazione sono diverse dal ragionamento umano[a].
- I *valori di verità* non sono adatti a rappresentare gli aspetti quantitativi che caratterizzano il mondo reale

[a] Da premesse false discendono conseguenze vere.

Definizione 1.2.2: Monotònicità

La logica del primordine è monotòna: le conoscenze inserite nel sistema logico non possono essere cancellate.

Note:-

La conoscenza può solo aumentare.

Osservazioni 1.2.2

- Le nuove conoscenze non possono contraddire quelle già presenti nel sistema.
- Impossibilità di rappresentare il cambiamento, quindi gli aspetti temporali della conoscenza.

Definizione 1.2.3: Limitazioni Computazionali

Il calcolo dei predicati è *semi-decidibile*: è possibile dimostrare con certezza cosa discende dalla base di conoscenza, ma non si ha la certezza di dimostrare ciò che non discende dalla base di conoscenza.

Corollario 1.2.1 Inefficienza nelle Procedure di Dimostrazione

Per le sue caratteristiche intrinseche (è un formalismo orientato alla rappresentazione) le procedure di dimostrazione non sono abbastanza efficienti per le applicazioni reali.

Domanda 1.1

Ma cosa possiamo dire sulle logiche non classiche?

Le logiche non classiche rilasciano alcune caratteristiche della logica classica:

- Valori di verità ⇒ *logiche fuzzy*, hanno valori di verità in un intervallo, nascono per superare la rappresentazione binaria dell'informazione.

Figure 1.1: Rappresentazione accurata di una logica fuzzy.

- Conoscenza certa ⇒ *logiche bayesiane*, incorporano il ragionamento probabilistico.
- Monotonicità ⇒ *default logics*, belief revision.
- Indecidibilità ⇒ insiemi decidibili della logica del primordine.
- Inefficienza delle dimostrazioni ⇒ uso di *euristiche* nella dimostrazione automatica.

Un quadro storico:

- Anni 60s-70s: reti semantiche e frames.
- Anni 70s-80s: logiche non classiche.
- Anni 80s-90s: logiche descrittive.
- Anni 2000s: ontologie computazionali.
- Dal 2010: Linked Data.

1.2.2 Reti Semantiche

Definizione 1.2.4: Reti Semantiche

Le reti semantiche sono:

- Basate su una struttura a *grafo*.
- I nodi del grafo rappresentano *concetti*.
- Gli archi tra i concetti rappresentanole *relazioni tra concetti*.

Note:-

Nel "Mondo dei Blocchi" c'è un tavolo su cui è collocato un blocco verde chiamato "a" e sopra un blocco rosso chiamato "b". Una delle possibili rappresentazione è quella dell'immagine in cui sono presenti concetti, qualità

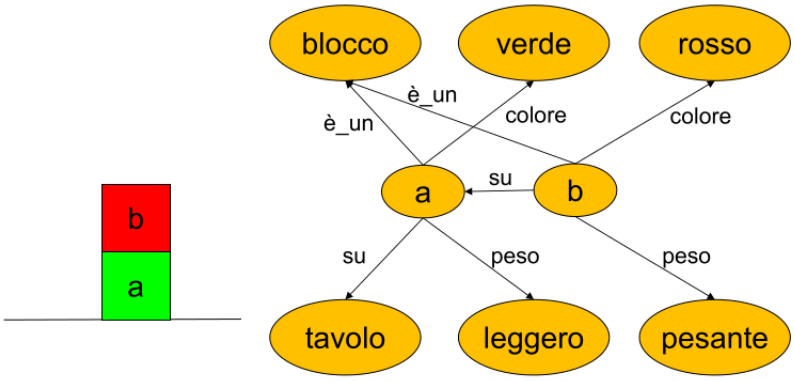


Figure 1.2: Grafo relazionale del "Mondo dei Blocchi", un dominio giocattolo utilizzato nel mondo dell'AI.

ed entità concrete. Gli archi sono varie relazioni.

Vantaggi di questa rappresentazione:

- Le informazioni relative a un nodo sono *immediatamente disponibili* dato che ogni blocco è direttamente collegato alle sue proprietà.
- Permette di rappresentare una nozione di *rilevanza* per cui dato un focus, alcune informazioni si trovano in prossimità.

Corrispondente rappresentazione logica:

- Blocco(a).
- Blocco(b).
- su(b,a).
- su(a,tavolo).
- rosso(b).
- verde(a).
- pesante(b).
- leggero(a).

Corollario 1.2.2 Ragionamento nelle Reti Semantiche

Nelle reti semantiche il ragionamento consiste nel seguire un percorso tra nodi.

Domanda 1.2

Su quale blocco si trova il blocco b?

Risposta: È sufficiente seguire l'arco etichettato come «su» nella rete, da b verso il nodo a cui punta. Seguendo ulteriormente gli archi si possono inferire relazioni indirette: il blocco b è "al di sopra" del tavolo.

1.2.3 Reti Semantiche e Logica

L'espressività delle reti semantiche corrisponde a un sottoinsieme della logica del primordine:

- Nodi = *termini*.
- Archi = *predicati*

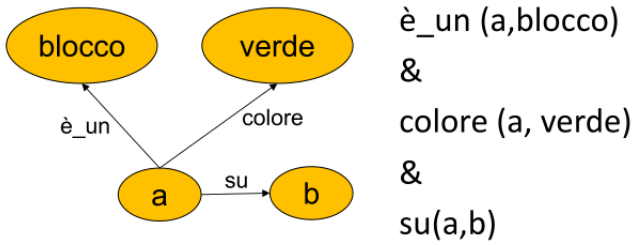


Figure 1.3: Grafo relazionale in relazionale alla logica del primordine.

Note:-

Alcune relazioni possono coinvolgere più di due entità.

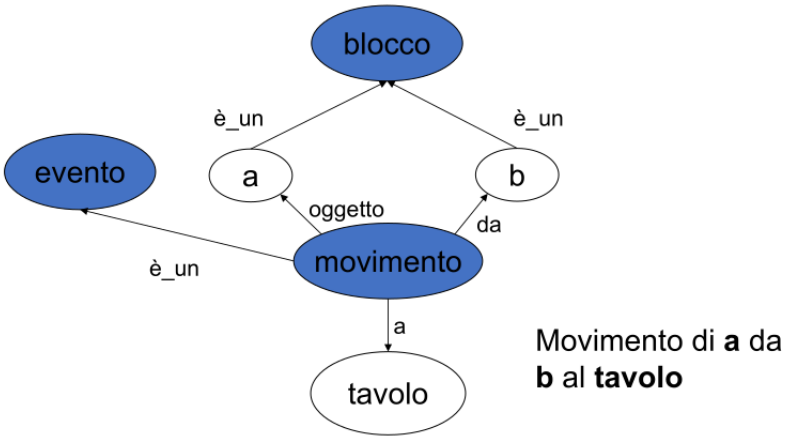


Figure 1.4: La relazione "movimento" che unisce a, b e tavolo diventa un termine cioè un nodo.

Definizione 1.2.5: Reti Semantiche Proposizionali

Le *reti semantiche proposizionali* includono nodi che rappresentano proposizioni. Usando nodi per rappresentare proposizioni è possibile introdurre una *dimensione epistemica*:

- Rappresentare credenze soggettive.
- Lo stesso sistema può rappresentare le credenze di più soggetti senza che insorgano contraddizioni.

Osservazioni 1.2.3

- Reti semantiche proposizionali posso avere l'espressività della logica del primordine una volta introdotti connettivi, variabili, quantificatori, ecc.
- Anche l'inferenza nelle reti proposizionali ha le stesse caratteristiche che nella logica del primordine.

- Soluzione: limitare l'espressività privilegiando tipi di ragionamento più comuni e computazionalmente trattabili.

Definizione 1.2.6: SNePs

Rete semantica proposizionale che incorpora alcuni elementi della teoria dei frame (Shapiro '79).

Note:-

Si tratta di una rete semantica con un *motore di ragionamento*. Permette vari tipi di ragionamento:

- Su formule.
- Su slot.
- Su percorsi.

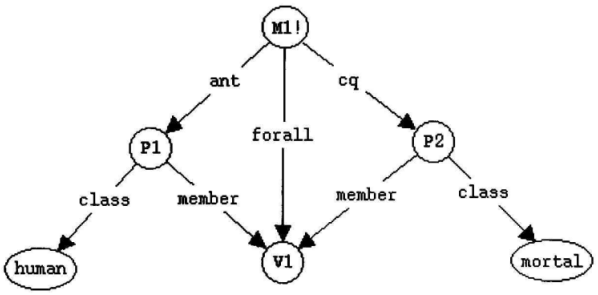


Figure 1.5: Tutti gli uomini sono mortali.

Domanda 1.3

Di cosa parla la rete?

Definizione 1.2.7: Eterogeneità

- Archi rappresentano relazioni di tipo diverso tra concetti (relazioni epistemiche vs fatti).
- Nodi rappresentano concetti di tipo diverso (blocco A, blocco)

Note:-

Alcuni tipi di nodi e di archi sono particolarmente importanti.

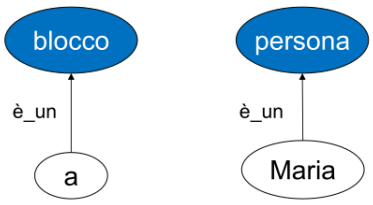


Figure 1.6: I nodi colorati rappresentano tipi di entità (classi), i nodi bianchi rappresentano entità singole (individui).

Domanda 1.4

"What ISA is and isn't?"

- Gli archi "è un" (IS-A o ISA) hanno un significato diverso se collegano due classi oppure un individuo a una classe.
- Brachman ('83) propone di distinguere i due tipi di relazioni: "What isa is and isn't":
  - *Archi ISA*: appartenenza di una classe a una sottoclasse (transitiva).
  - *Archi instance_of*: appartenenza di un individuo a una classe.

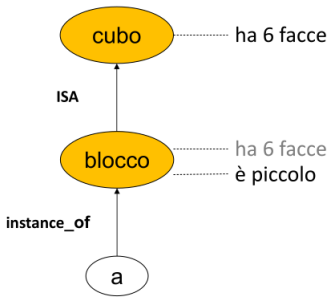


Figure 1.7: ISA e instance_of.

Definizione 1.2.8: Tassonomia

Utilizzando la relazione ISA è possibile esprimere conoscenza di tipo tassonomico. Si utilizza un ragionamento classificatorio:

- Y è una sottoclasse di Z? ⇒ archi ISA.
- x appartiene alla classe Y? ⇒ archi instance_of.

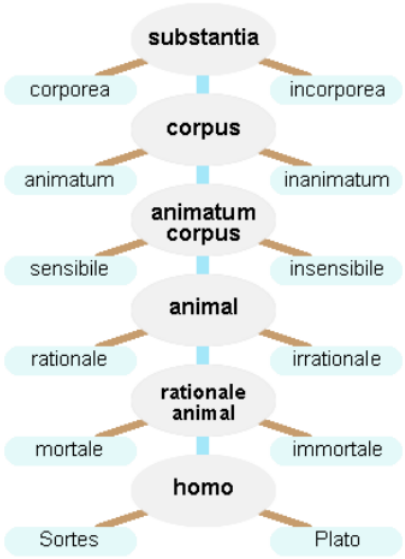


Figure 1.8: Rappresentazione che deriva dalla filosofia aristotelica, confluita nella scolastica, in cui si raffigurano i principali concetti del pensiero dalla sostanza all'uomo.

Note:-

Ogni livello *eredità* le caratteristiche del livello superiore e le caratteristiche non si sovrappongono.

Vantaggi dell'ereditarietà:

- E' possibile avere una rappresentazione meno ridondante facendo una semplice assunzione:
  - Una classe eredita le proprietà delle classi di cui è sottoclasse.
  - Se i cubi hanno sei facce e i dadi sono una sottoclasse dei cubi, allora anche i dadi hanno sei facce.
- Le sottoclassi hanno proprietà più specifiche delle classi.
- Seguendo il percorso degli archi ISA e instance_of e diventa possibile ragionare sulle proprietà di un individuo/classe.

Note:-

Ma esistono eccezioni, per esempio i pinguini non volano, ma non per questo smettono di essere uccelli. Allo stesso tempo può essere falso per un individuo all'interno di una sottoclasse.

Definizione 1.2.9: Default Rules

Per gestire le eccezioni è necessario rilasciare la proprietà della monotonicità (la conoscenza non diminuisce mai). Un esempio è la Default Logic: assunzioni che possono essere cancellate quando sopravviene nuova conoscenza.

Osservazioni 1.2.4

- Il trattamento delle eccezioni nelle reti semantiche si basa sul principio che le conoscenze specificate localmente a un certo nodo prevalgono su quelle ereditate.
- Un corollario è che le conoscenze che comportano meno passi di inferenza prevalgono su quelle che ne comportano di più.
- Questo principio però non permette di scongiurare tutti gli inconvenienti.

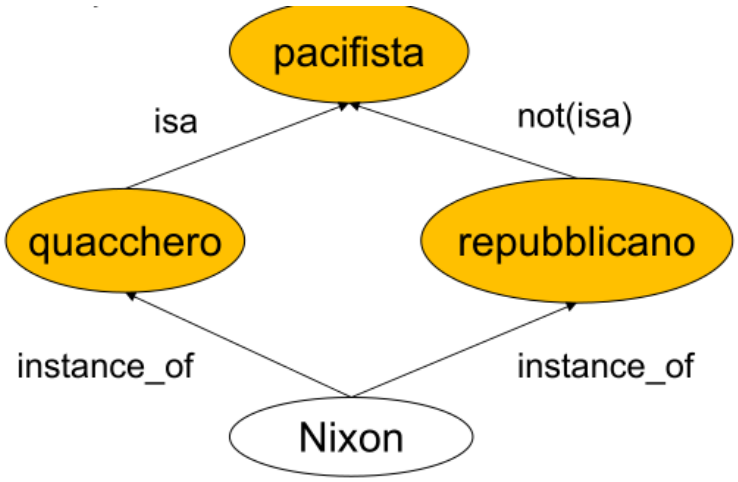


Figure 1.9: Nixon è pacifista o guerrafondaio? In quando quacchero, lo è; in quanto repubblicano, non lo è (Reiter and Criscuolo, On Interacting Defaults, 1981).

Note:-

Nemmeno la logica dei default risolve il problema, ciascun default blocca l'altro.

Definizione 1.2.10: Frame Theory

Evoluzione delle reti semantiche finalizzata a rappresentare la conoscenza di tipo stereotipato. Conoscenza di sfondo utile per alcune applicazioni come la *visione artificiale* o *l'elaborazione del linguaggio*.

Definizione 1.2.11: Frame

Un *frame* è una struttura dati per la rappresentazione situazioni stereotipate, come trovarsi in una certa tipo di soggiorno o andare al compleanno di un bambino partito. I livelli superiori del telaio sono fissi, e rappresentano cose che sono sempre vere riguardo le situazioni presunte. I livelli inferiori hanno molti terminali: slot che devono essere riempiti da istanze o dati specifici.

Corollario 1.2.3 Slot

Ogni terminale (slot) può specificare le condizioni che Gli incarichi devono soddisfarsi. (Gli incarichi sono solitamente telai ausiliari più piccoli). Le condizioni possono richiedere un terminale l'incarico di essere una persona, un oggetto ora puntatore a un subframe di un certo tipo. I terminali di un frame sono solitamente riempiti da assegnazioni predefinite. Collezioni di frame collegati sono uniti in frame systems.

Struttura di un frame:

- Identificativo.
- Slot: permettono di creare una tassonomia di frame.
- Slot generici: i valori degli slot sono vincolati a un certo tipo, però il contenuto di uno slot può puntare a un altro frame.
- Procedure per il calcolo automatico dei valori.
- Valori predefiniti.

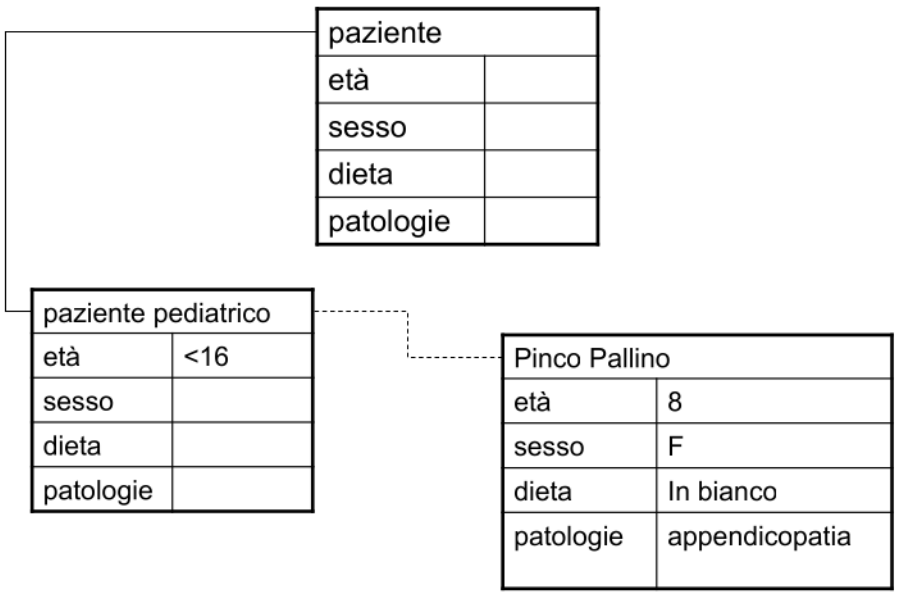


Figure 1.10: Esempio di tassonomia.

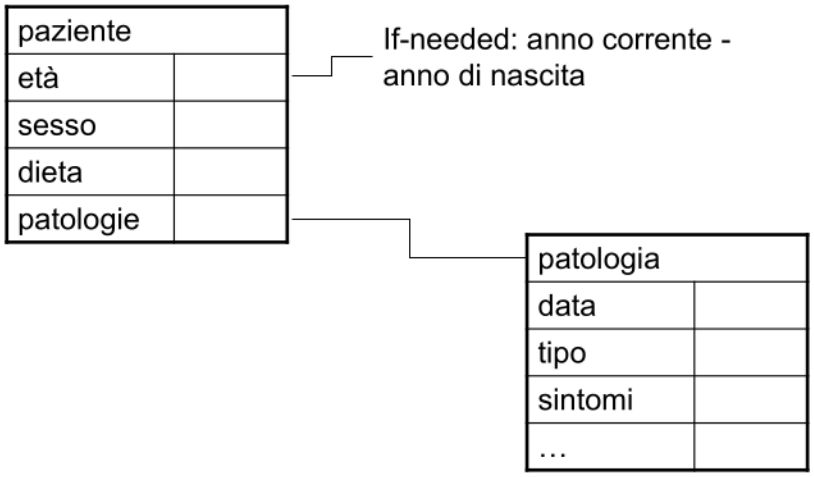


Figure 1.11: Esempio di collegamenti non tassonomici.

Definizione 1.2.12: Regole di Produzione

Conoscenza di tipo condizione-azione. Regole IF - THEN (anche dette produzioni):

- *La parte sinistra*: condizioni per l'applicazione delle regola.
- *La parte destra*: azioni che sono effettuate se la condizione è soddisfatta.

Osservazioni 1.2.5

- Conoscenza dichiarativa:
  - IF allatta(animale,prole) THEN isa(animale,mammifero).
  - IF febbre>38,5 and tosse THEN malattia_da_raffreddamento.
- Conoscenza procedurale:
  - IF intensità(sibilo)>t THEN chiudi(valvola,5mm).

1.2.4 Sistemi a Regole

Componenti:

- Base delle regole.
- Memoria di lavoro.
- Interprete delle regole: motore inferenziale.

L'interprete delle regole esegue questo ciclo:

1. Confronto (match) tra fatti (nella memoria di lavoro) e antecedente delle regole.
2. Risoluzione di conflitti (più regole applicabili).
3. Esecuzione (e aggiunta di nuovi fatti alla memoria di lavoro).

Conflitti di applicazione:

- *Refrattarietà*: la stessa regola non viene applicata più volte.
- *Recenza*: vengono privilegiate le regole che si applicano ai fatti più recenti.
- *Specificità*: vengono applicate le regole più specifiche.
- *Pesatura*: assegnamento di importanza (peso) a priori alle regole o ai fatti.

Note:-

Un passo delicato è il confronto tra la memoria di lavoro e l'antecedente di tutte le regole,

Definizione 1.2.13: Mycin

Sistema esperto sviluppato a Stanford nei primi anni 70s (Shortliffe, '75). I fatti sono associati alle probabilità.

```
PREMISE: (AND
  (SAME CNTXT GRAM GRAMNEG)
  (SAME CNTXT MORPH ROD)
  (SAME CNTXT AIR ANAEROBIC))
ACTION: (CONCLUDE CNTXT IDENTITY BACTEROIDES TALLY .6)
```

- IF: 1) The site of the culture is blood, and 2) The gram stain of the organism is gramneg, and 3) The morphology of the organism is rod, and 4) The portal of entry of the organism is udne, and 5) The patient has not had a genito-urinary manipulative procedure, and 6) Cystitis is not a problem for which the patient has been treated THEN: There is *suggestive evidence* (.6) that the identity of the organism is e.coli

Figure 1.12: Esempio di Mycin.

1.3 Il Sistema SNePS

SNePs è una rete semantica e sistema proposizionale che supporta vari tipi di inferenza.

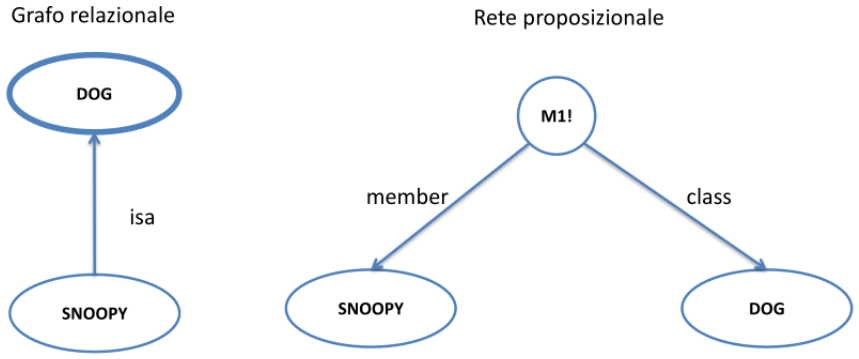


Figure 1.13: Nella rappresentazione di SNwPS (a destra) la proposizione "Snoopy è un cane" è rappresentata da un nodo (M1!). Nessun arco denota una proposizione x isa (è un) y, come invece avviene nel grafo relazionale (a sinistra).

Osservazioni 1.3.1

- SNePS fu progettato per la *robotica cognitiva*.
- Solo i nodi sono espressioni ben formate dotate di una semantica.
- Ogni nodo denota un'*entità mentale*.

Il sistema SNePS è un software dotato delle seguenti funzionalità:

- Rappresentare la rete: si utilizza un linguaggio utente fatto di comandi che permettono di *fare asserzioni* (Socrate è un uomo), *esprimere regole* (Gli uomini sono mortali) e *manipolare la rete* (creare proposizioni non asserite).
- *Cercare* nella rete nodi con certe caratteristiche (gli individui che sono uomini).
- *Inferire* nuove asserzioni a partire da quelle esistenti.

Input e Output:

- L'utente usa il *linguaggio utente* per asserire una proposizione.
- Il sistema crea una rete che rappresenta la proposizione:
  - Tipicamente, un nodo "proposizione" da cui emanano degli archi.
  - La rappresentazione viene creata internamente al sistema e comunicata all'utente per mezzo di una particolare notazione (*linguaggio del sistema*).
  - Alcune versioni del sistema erano dotate di un output grafico per rappresentare visivamente la rete.
- Un insieme di proposizioni può essere interrogata oppure può diventare la base per inferenze. Sia l'estrazione di proposizioni dalla rete che la deduzione di nuove proposizioni avvengono su richiesta dell'utente, cioè quando l'utente digita il comando corrispondente nel linguaggio utente.

Tipi di nodo:

- I nodi molecolari (Mi) hanno archi uscenti. Rappresentano proposizioni, incluse le regole di ragionamento, oppure individui.
- I nodi base (Bi) non hanno archi uscenti. Rappresentano individui che hanno certe caratteristiche, ma a cui non vogliamo dare un nome.
- I nodi variabili non hanno archi uscenti ma rappresentano individui arbitrari o proposizioni, come le variabili logiche.

Definizione 1.3.1: Relazioni

Le relazioni tra i nodi sono date dagli archi che li uniscono. Se non specificato diversamente una relazione tra un nodo A e uno B genera un arco diretto da A verso B con il nome della relazione specificata.

Note:-

Il comando seguente definisce due relazioni, isa e ako (a kind of):

(define isa ako)

Definizione 1.3.2: Asserzione

Il comando *assert* permette di asserire una proposizione. Ha come argomento una lista di coppie, di cui:

- Il primo elemento è una relazione.
- Il secondo è un node o una lista di nodi.

- forall
- exists
- min
- max
- thresh
- ant
- &ant
- cq
- dcq
- arg
- default
- if
- action
- ...

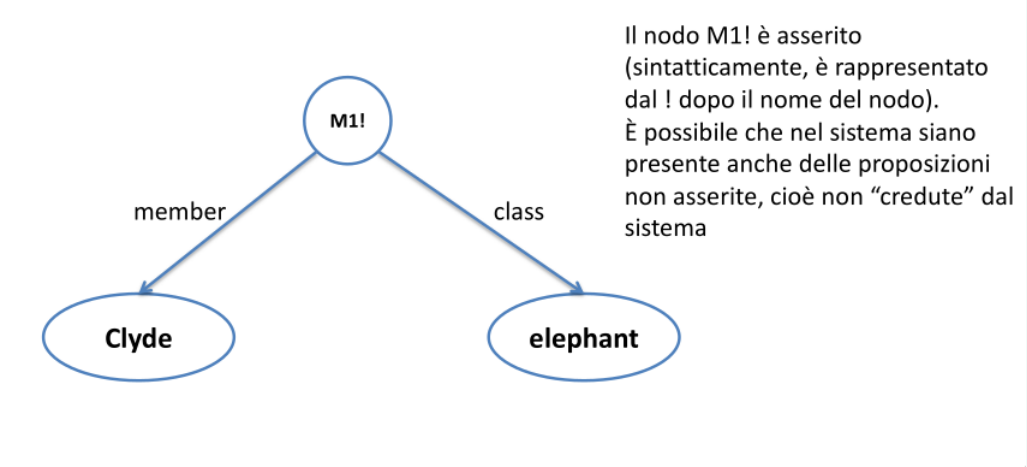
- Esistono relazioni (archi) predefiniti nel sistema
- Nella versione più recente i tipi di relazione sono organizzati in una tassonomia

Figure 1.14: Alcune relazioni predefinite.

Esempio 1.3.1 (Assert)

(assert member Clyde class elephant)

Il sistema costruirà un nodo M1 con un arco member che punta all'identificativo Clyde e un arco class che punta un nodo elefante. M1! Significa che il nodo M1 è asserito.



Definizione 1.3.3: Contesti

Un nodo può essere asserito oppure non asserito. Allo scopo di rappresentare le credenze di più soggetti diversi, il sistema può contenere più *contesti*:

- Un contesto è formato da ipotesi.
- Un'*ipotesi* in un contesto è un nodo asserito oppure derivato via inferenza.

Note:-

Un contesto si crea con il comando:

:context nodeset context-name

Uso dei nodi di base:

- I nodi base vengono usati per rappresentare individui che soddisfano determinate descrizioni senza dargli un nome.
- Non hanno archi uscenti; cioè non hanno informazioni strutturali. Tutto ciò che si sa di loro è ciò che viene asserito.
- Un nodo base potrebbe essere descritto tale l'espressione "un entità tale che . . . ".

Esempio 1.3.2 (Nodi di base)

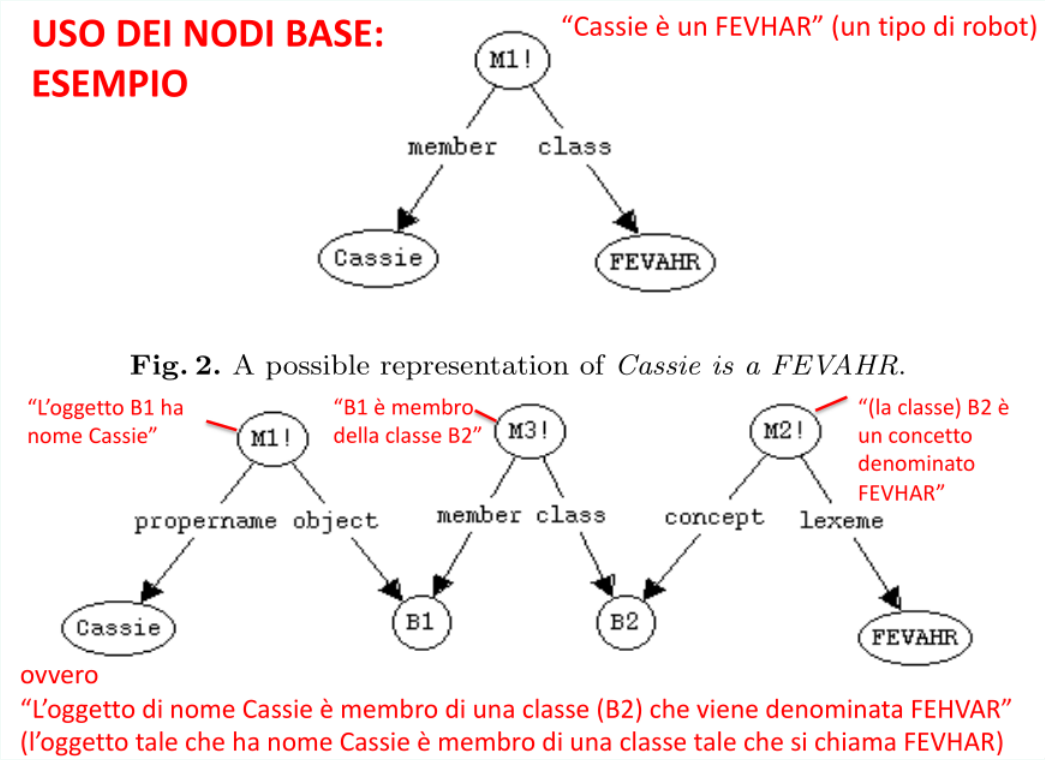


Fig. 2. A possible representation of *Cassie is a FEVAHR*.

1.3.1 Interrogare la Rete

Definizione 1.3.4: Find

Il comando utente *find* trova uno o più nodi nella rete.

Note:-

In pratica, la rete si comporta come un database di fatti che può essere interrogato descrivendo il tipo di fatti cercato.

Esempio 1.3.3 (Find)

(find class elephant)

Trova tutti i nodi che hanno un arco di tipo classe che esce dal nodo e punta al nodo elefante, ossia: *trova tutte le proposizioni che esprimo che qualcosa è un elefante.*

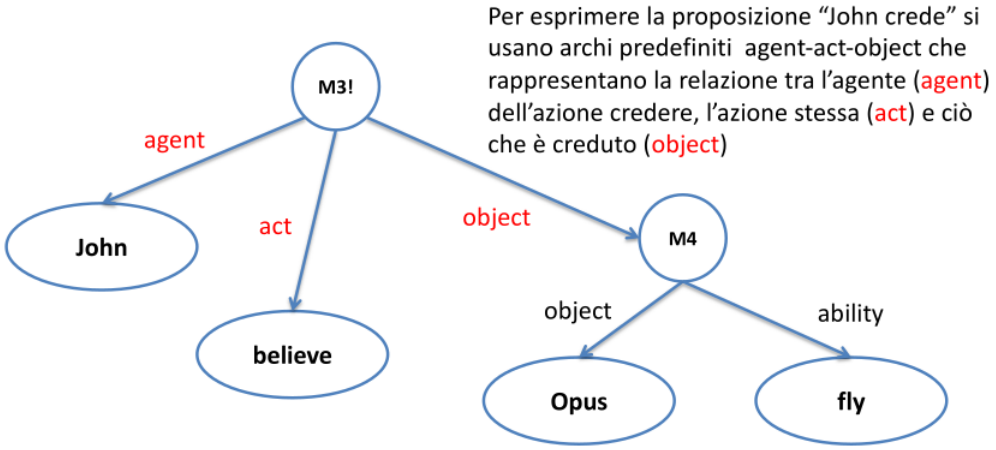
Osservazioni 1.3.2

- Il comando find permette di specificare uno o più percorsi per trovare i nodi cercati.
- (find (member- class) canary (object- ability) fly): Trova i canarini (membri della classe canarino) che volano (che hanno l'abilità di volare).
- Tutti i nodi che hanno un arco member entrante, collegato (tramite un nodo intermedio) a un arco class, e un arco object entrante, collegato (tramite un nodo intermedio) a un arco ability.

1.3.2 Rappresentazione delle Credenze e Deduzioni

(assert agent John act believe object (build object Opus ability fly))

- Attenzione: il sistema non crede che Opus possa volare ma crede che John creda che Opus possa volare.
- Con un find si può scoprire cosa il sistema crede che John creda.



M3 = "John crede M4" (John è *l'agente dell'azione* di credere *l'oggetto* M4)
M4 = "Opus può volare"
"John crede che Opus possa volare"
Importante: M4 non è asserito perché è una credenza di John non del sistema!

Definizione 1.3.5: Deduce

Il comando deduce permette di cercare proposizioni che non sono direttamente asserite dal sistema ma che il sistema è in grado di inferire.

Esempio 1.3.4 (Deduce)

(deduce member $x class canary)

Trova tutti gli x che appartengono alla classe dei canarini ($ indica una variabile).

1.3.3 Inferenze

Definizione 1.3.6: Riduzione

La riduzione consiste nel dedurre da un grafo una porzione contenuta in esso.

Esempio 1.3.5 (Riduzione)

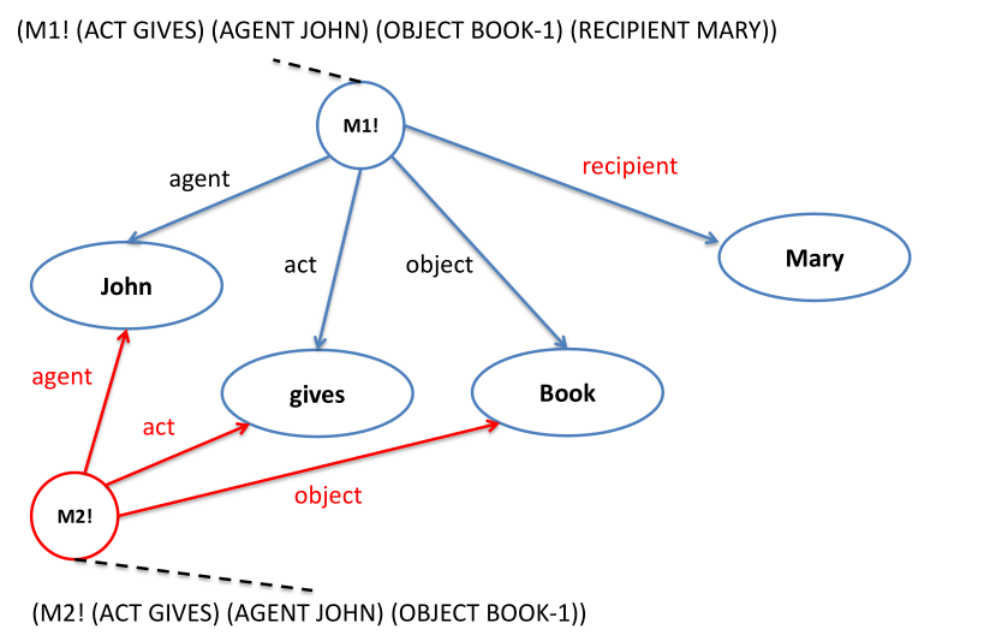
(assert agent john act gives object book-1 recipient mary)

John dà un libro a Mary.

Può essere ridotto a:

(deduce agent john act gives object book-1)

John dà un libro.



Definizione 1.3.7: Define-path

È possibile stabilire che un percorso fatto di certe relazioni è uguale a una relazione. Si usa il comando define-path.

Corollario 1.3.1 Compose

Per definire un percorso si usa il comando compose che si può pensare come comporre una nuova relazione (arco) basandosi su un percorso di relazioni.

Note:-

In pratica, si dice che un certo percorso (o meglio ogni percorso con certe caratteristiche) è uguale a un singolo arco.

Esempio 1.3.6 (Relazione isa transitiva)

(define-path isa (compose isa (kstar (compose object- isa))))

- kstar[a] significa "zero o più occorrenze".
- Si legge: isa è guale a isa composto con un percorso di zero o più percorsi ottenuti da un object-

composto con un isa.

[a]Kleene star per chi si ricorda di LFT.

Definizione 1.3.8: Regole di Ragionamento

Le regole sono definite dall'utente in modo arbitrario. Fanno parte della base di conoscenza. Permettono di aumentare drasticamente la conoscenza contenuta nella rete, perché aggiungono nuove proposizioni non implicite nella rete.

Esempio 1.3.7 (Regole)

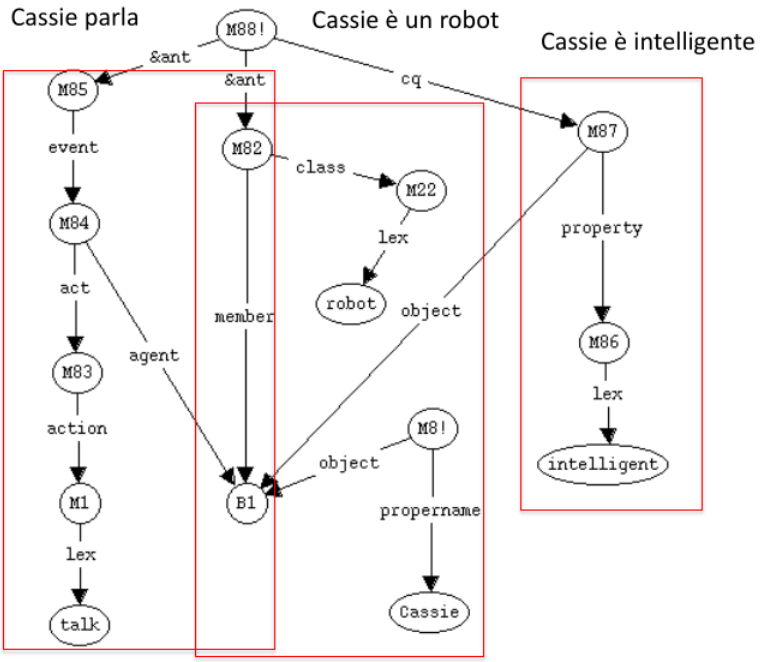


Fig. 5. `M88!` is a rule node denoting the proposition, *If Cassie is a robot that talks, then Cassie is intelligent.*

- Antecedente 1 (M85), Cassie parla: in un evento (M84) B1 è l'agente dell'esecuzione (act) dell'azione (M83) denominata parlare (lex talk).
- Antecedente 2 (M82), Cassie è un robot: B1 è membro della classe (M22) denominata (lex) robot e ha come nome proprio (propername) Cassie (quest'ultimo fatto è asserito).
- Conseguente (M87), Cassie è intelligente: B1 (Cassie) ha la proprietà (M86) di essere intelligente (denominate "intelligent").

1.4 RDF

Definizione 1.4.1: RDF

RDF è un formato che permette di descrivere risorse. Una risorsa può essere qualsiasi cosa: un documento, una persona, un oggetto fisico, un concetto astratto.

Definizione 1.4.2: RDFS

RDFS: permette di descrivere relazioni tra risorse. È un linguaggio "property-centric" ed è un'estensione di RDF.

Definizione 1.4.3: OWL

OWL (Ontology Web Language): permette di descrivere ontologie computazionali. Si colloca a un livello superiore di RDF. Permette di descrivere relazioni ricche e complesse tra entità ("complex knowledge about things, groups of things, and relations between things").

1.4.1 Introduzione

Carta d'Identità di RDF:

- Nato per *descrivere* risorse digitali.
- Il suo data model è basato sul concetto di *grafo*.
- I *nodi* rappresentano entità e gli *archi* relazioni tra di esse.
- Possiede diverse *serializzazioni*.

Elementi del data model di RDF:

- Triple (organizzate in grafi).
- IRI.
- Letterali.
- Blank nodes.
- Grafi.

Note:-

L'unità di base di RDF è una tripla, composta di soggetto, predicato e oggetto. Il soggetto e oggetto possono essere IRI o blank nodes, l'oggetto può essere anche un letterale, il predicato può essere solo un IRI.

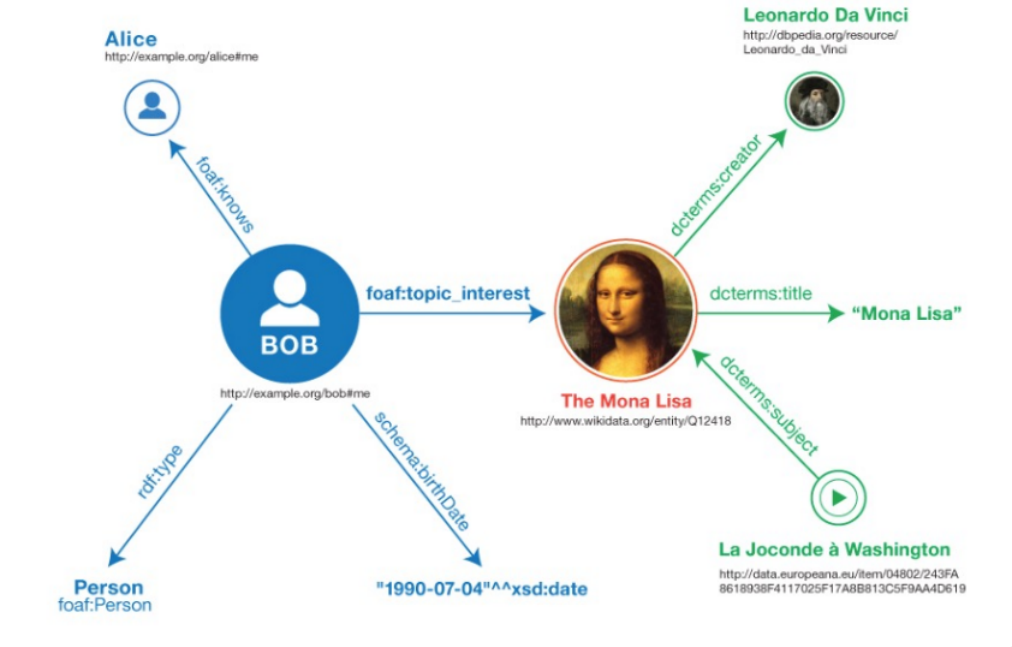
Definizione 1.4.4: Triple

RDF è formato di triple, che hanno la forma: Soggetto – Predicato – Oggetto

Esempio 1.4.1 (Triple)

<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
<the Mona Lisa> <is entitled> <Mona Lisa>.
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa>.

Esempio 1.4.2 (Dalla tripla al grafo)



Note:-

Gli elementi della tripla devono essere riconducibili a entità presenti nel web.

Definizione 1.4.5: URI e IRI

Lo strumento scelto da RDF per rappresentare le entità è dato dal sistema degli IRI (URI Internazionalizzato).

Note:-

Cosa rappresenti nella realtà un determinato IRI non è rilevante: l'importante è che sia de-referenziabile. Il predicato è sempre rappresentato da un IRI.

Definizione 1.4.6: Letterali

Qualsiasi tipo di dato che non sia un IRI (stringhe, numeri, booleani, etc.)

Note:-

Solo in posizione di oggetto nella tripla. Sostanzialmente coincidono con i data types di XML Schema, anche se il supporto può essere parziale in alcuni software.

Definizione 1.4.7: Blank Node

Permettono di denotare risorse senza utilizzare un IRI.

Note:-

Un blank node è come una variabile e può comparire in posizione di soggetto e oggetto nella tripla. Possono essere associati a un identificativo nella rappresentazione delle triple in un determinato store, ma non hanno significato al di fuori di esso.

1.4.2 Vocabolari e Grafi

Definizione 1.4.8: Vocabolari Condivisi

Per descrivere una certa entità, la tripla si basa sull'utilizzo di vocabolari condivisi. Invece di "inventare" ogni volta i termini per descrivere le entità di cui parla la tripla, meglio usare tutti gli stessi termini.

Esempio 1.4.3 (Utilizzo di vocabolari condivisi)
<Bob><http://www.my_vocab#like><The Mona Lisa>.
<Bob><http://www.likesanddislikes/loves/adores><The Mona Lisa>.
Questo causa confusione, meglio concordare in anticipo su un unico termine:
<Bob><http://xmlns.com/foaf/0.1/topic_interest ><The Mona Lisa>.

Osservazioni 1.4.1

- In molte serializzazioni di RDF, il vocabolario è identificato da un prefisso, che corrisponde a un namespace.
- RDF non permette di specificare il significato dei predicati che mettono in relazione soggetto e oggetto.
- Tuttavia, con RDF si può asserire che una risorsa appartiene a una certa tipologia (classe) utilizzando il predicato type.

Definizione 1.4.9: Grafi

Un grafo ha un identificativo dato da un IRI (o un blank node)

$$\text{Grafo} = \text{IRI} + \text{triple}$$

Corollario 1.4.1 Data Set

Un data set RDF è costituito da un insieme di grafi:

- Un grafo anonimo (*default graph*).
- Zero o più grafi con un nome (*named graphs*).

1.5 Turtle

Storicamente il formato di serializzazione è stato XML (RDF/XML), tuttavia XML si è imposto come standard per la condivisione di documenti .

1.5.1 Introduzione

Definizione 1.5.1: Turtle

Formato di serializzazione, preferito a RDF/XML per la maggiore coincisione e leggibilità.

Osservazioni 1.5.1

1. Definizione di prefissi: il "base" diventa il prefisso di default a cui vengono ricondotti tutti gli IRI.
2. Commenti: i "cancelletti".
3. Triple: espresse secondo certe convenzioni.

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<...#green-goblin>
   rel:enemyOf <...#spiderman> ;
   a foaf:Person ;   # in the context of the Marvel universe
   foaf:name "Green Goblin" .

<...#spiderman>
   rel:enemyOf <...#green-goblin> ;
   a foaf:Person ;
   foaf:name "Spiderman",
```

prefissi

commento

triple

Figure 1.15: Il formato Turtle.

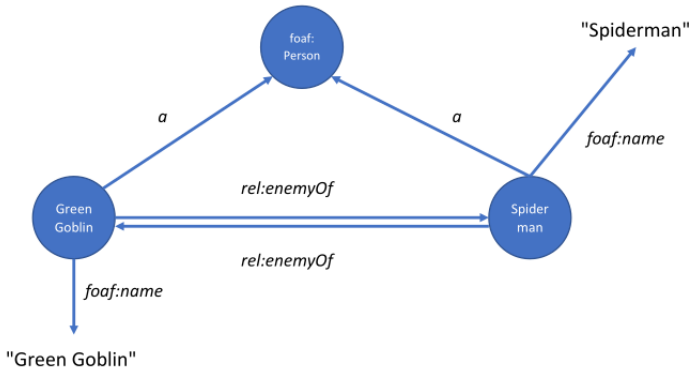


Figure 1.16: Rappresentazione a grafo.

1.5.2 Sintassi

Note:-

Gli IRI, se presenti per esteso, sono racchiusi tra < e >. La parte finale che segue # identifica un frammento (fragment identifier).

Definizione 1.5.2: Abbreviazione di Triple

";" indica che allo stesso soggetto si applicano più predicati (con diverso oggetto), equivale a ripetere il soggetto.

Osservazioni 1.5.2

- ":"è il prefisso vuoto.
- "," equivale a ripetere soggetto e predicato.

Note:-

La sintassi concreta dei blank node in Turtle è data dalle parentesi quadre [].
[] da solo indica un soggetto o oggetto non specificato.
È possibile raggruppare nelle parentesi un insieme di triple che hanno il blank node come soggetto.

Definizione 1.5.3: Abbreviazione di Valori Numerici

I valori numerici possono essere scritti in modo esteso (forma lessicale seguita da datatype) oppure in modo abbreviato.

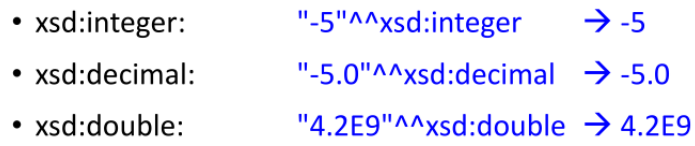


Figure 1.17: Abbreviazioni di numeri.

Note:-

Lo stesso vale per i booleani (true/false).

Definizione 1.5.4: N-Triples

Sottoinsieme di Turtle senza abbreviazioni e senza prefissi. Verboso e meno leggibile per la mancanza di abbreviazioni, ma è migliore per il trasferimento dei data set tra applicazioni.

Note:-

N-Quads è la sua estensione che aggiunge alla tripla un quarto elemento, il graph IRI.

1.6 Schemi RDF

RDF permette di creare un grafo per descrivere risorse, ma non fornisce nessuna informazione sulle entità descritte né sul vocabolario utilizzato per descriverle. La *descrizione del vocabolario* è affidata al livello successivo, Schemi RDF (RDFS). Attraverso il linguaggio degli *schemi RDF* è possibile descrivere un semplice vocabolario di tipi entità (classi) e relazioni tra di esse (proprietà).

Domanda 1.5

Perché il focus sulle proprietà?

Con RDF si può:

- Definire classi e proprietà.
- Creare gerarchie di classi e proprietà.
- Fare asserzioni sulle singole risorse è di pertinenza di RDF.

Si definisce property-centric perché il focus è sulle proprietà:

- Le classi compaiono nella definizione delle proprietà, ma non viceversa.
- Questo rende possibile aggiungere proprietà senza modificare le classi (estensibilità).

Class	C (a resource) is an RDF class
Property	P (a resource) is an RDF property
Type	I (a resource) is an instance of C (a class)
Subclass	C1 (a class) is a subclass of C2 (a class)
Subproperty	P1 (a property) is a sub-property of P2 (a property)
Domain	domain of P (a property) is C (a class)
Range	range of P (a property) is C (a class)

Figure 1.18: Principali costrutti di RDF Schema.

Osservazioni 1.6.1

- Gli elementi dei linguaggi RDF e RDFS sono contraddistinti dai prefissi rdf: e rdfs: che li identificano come tali.
- Ogni serializzazione (tranne N-Triples) rappresenta i namespace e gli IRI secondo un formato specifico.

1.6.1 Definizioni in RDF

```
#vocabolario Animals
@prefix animals: <http://www.funny_animals.org#> .

# definire la classe Film
animals:Film rdf:type rdfs:Class .

#definire la classe Animale
animals:Animale rdf:type rdfs:Class .

#definire la classe Quadrupede, sottoclasse di Animale
animals:Quadrupede rdf:type rdfs:Class ;
                   rdfs:subClass animals:Animale .
```

Figure 1.19: Definizione di classi e sottoclassi in RDF.

```
<!– classe Animal -->
<rdfs:Class rdf:about="http://www.funny_animals.org#Animal"/>

<!– classe Quadruped, sottoclasse di Animal -->
<rdfs:Class rdf:about="http://www.funny_animals.org#Quadruped">
    <rdfs:subClassOf rdf:resource="http://www.funny_animals.org#Animal"/>
</rdfs:Class>
```

Figure 1.20: Definizione di classi e sottoclassi in RDF/XML.

```
#proprietà hasLegs
animals:hasLegs rdf:type rdf:Property .

#proprietà hasFourLegs
animals:hasFourLegs  rdf:type rdf:Property ;
                     rdfs:subPropertyOf animals:hasLegs .
```

Figure 1.21: Definizione di proprietà in RDF.

```
<rdf:Property rdf:about="http://www.funny_animals.org#hasLegs" />

<rdf:Property rdf:about="http://www.funny_animals.org#hasFourLegs">
    <rdfs:subPropertyOf rdf:resource="http://www.funny_animals.org#hasLegs"/>
</rdf:Property>
```

Figure 1.22: Definizione di proprietà in RDF/XML.

Note:-

La proprietà definita si chiama hasFourLegs (rdf:about), è una sottoproprietà di hasLegs (subPropertyOf).

In RDFS, è possibile definire domain e range delle proprietà.

```
# La proprietà abita ha come dominio la classe Animal e come
range la classe Habitat
animals:abita rdfs:domain animals:Animale ;
              rdfs:range animals:Habitat .

# La proprietà hasLegs ha come range la classe Leg
animals:hasLegs rdfs:range animals:Leg ;
```

Figure 1.23: Collegare le classi tramite le proprietà.

```
<rdf:Property rdf:about="http://www.funny_animals.org#hasLegs">
    <rdfs:domain rdf:resource="http://www.funny_animals.org#Animal"/>
    <rdfs:range rdf:resource="http://www.funny_animals.org#Leg"/>
</rdf:Property>
```

Figure 1.24: Domain e range delle proprietà (RDFS).

1.6.2 Vocabolari RDF

- *FOAF (Friend Of A Friend)*: vocabolario per descrivere reti sociali.
- *Dublin Core*: un vocabolario di elementi (titolo, autore, ecc.) per descrivere risorse in termini editoriali.
- *Schema.org*: vocabolario nato su iniziativa di un consorzio di industrie per descrivere gli argomenti delle pagine web secondo uno schema semantico.
- *SKOS*: W3C recommendation dal 2009, permette di descrivere e allineare terminologie diverse.

Definizione 1.6.1: Vocabolario RDF

La definizione di un insieme di classi e proprietà costituisce un vocabolario. Il vocabolario viene pubblicato a un certo IRI perché possa essere riferito da tutti.

Note:-

Uno o più vocabolari possono essere usati per descrivere i dati.

Osservazioni 1.6.2

- La descrizione di risorse in riferimento a uno o più vocabolari (es. Monna Lisa), con proprietà che collegano tra di loro le risorse è denominata Knowledge Graph.
- Classi e proprietà $\Rightarrow$ Vocabolario.
- Triple $\Rightarrow$ Knowledge graph.

Domanda 1.6

Cosa non si può dire?

- Non si può dire che Zebre e Leoni sono classi disgiunte. Solo che sono tipi specifici di animali.
- Non si può dire che la relazione 'conoscente di' è transitiva.
- Non si può dire che esistono individui. Esistono solo risorse, cioè IRI creati esternamente al linguaggio.

1.6.3 Dublin Core

Definizione 1.6.2: Dublin Core

- In un archivio, i documenti sono descritti in base alle loro proprietà: i metadati.
- I documenti/opere (risorse per RDF) sono descritti in un catalogo.
- Il catalogo è un insieme di schede che descrivono le risorse.

Note:-
Dublin non è Dublino, ma una città dell'Ohio.

Figure 1.25: Only in Ohio.

Domanda 1.7

Ma quindi cos'è Dublin Core?

- È uno schema di *metadati*: annotano risorse testuali e multimediali.
- Nasce per il web.
- Ha lo scopo esplicito di permettere il *reperimento di risorse*.

Evoluzioni successive:

- Metadati in formato XML con mapping tra schemi diversi.
- Formato RDF con apporti da altri vocabolari.
- Linked Dara: diffusione dei termini di Dublin Core, supporto per l'estrazione della conoscenza.

Domanda 1.8

Cosa permette di descrivere?

- Risorse che hanno un'identità.
- Collezioni, dataset, eventi, immagini, risorse interattive, servizi, software, suoni, testi, oggetti fisici.

Definizione 1.6.3: Dublin Core Metadata Element Set (DCMES)

L'insieme base di 15 descrittori e il loro significato.

Corollario 1.6.1 DCTerms

I termini del vocabolario Dublin Core. Classi e proprietà on namespace dc.

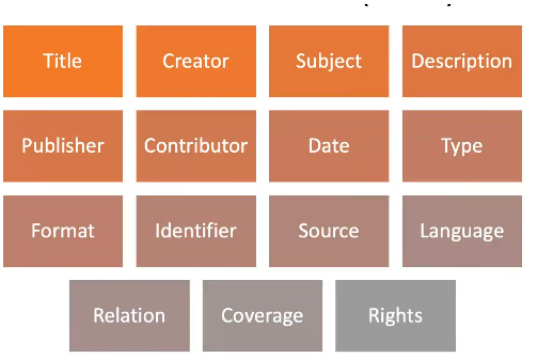


Figure 1.26: I 15 descrittori DCMES.

Osservazioni 1.6.3

- Domains e ranges specificano che tipo di risorse sono associate a una determinata proprietà.
- Domains e ranges specificano il significato implicito nel linguaggio naturale che è utilizzabile per la processazione automatica di inferenze logiche.
- Quando si trova una determinata proprietà, l'applicazione di un'inferenza può usare informazioni sui domains e i ranges assegnati alla proprietà in modo da derivare la risorsa descritta.

Name:	A token appended to the URI of a DCMI namespace to create the URI of the term.
Label:	The human-readable label assigned to the term.
URI:	The Uniform Resource Identifier used to uniquely identify a term.
Definition:	A statement that represents the concept and essential nature of the term.
Type of Term:	The type of term as described in the DCMI Abstract Model [DCAM].

Comment:	Additional information about the term or its application.
See:	Authoritative documentation related to the term.
References:	A resource referenced in the Definition or Comment.
Refines:	A Property of which the described term is a Sub-Property.
Broader Than:	A Class of which the described term is a Super-Class.
Narrower Than:	A Class of which the described term is a Sub-Class.
Has Domain:	A Class of which a resource described by the term is an Instance.
Has Range:	A Class of which a value described by the term is an Instance.
Member Of:	An enumerated set of resources (Vocabulary Encoding Scheme) of which the term is a Member.
Instance Of:	A Class of which the described term is an instance.
Version:	A specific historical description of a term.
Equivalent Property:	A Property to which the described term is equivalent.

Figure 1.27: Specifica dei termini.

Definizione 1.6.4: Europeana

Archivio europeo dei beni culturali. I dati sono descritti in un formato basato su Dublin Core. La semantica con cui sono descritti i dati permette di creare collezioni in modo semi-automatico.

Note:-

Europeana è basata su OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting): raccoglie le schede e crea un catalogo.

1.6.4 FOAF

Definizione 1.6.5: FOAF

FOAD è diviso in due sezioni:

- Core: dati classici.
- Social Web: riguardano gli aspetti dei social.

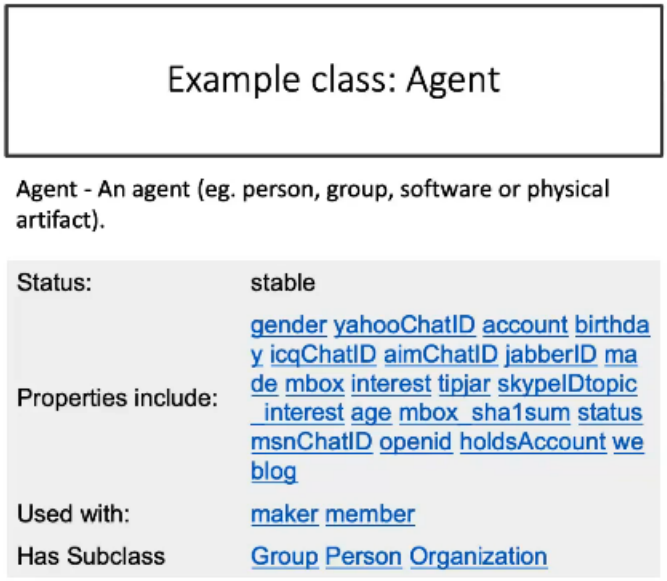


Figure 1.28: Esempio di classe.

1.6.5 Schema.org

Definizione 1.6.6: Schema.org

Lo schema.org è un prodotto di un'attività collaborativa che ha lo scopo di creare, mantenere e promuovere schemi per strutture dati su internet.

Note:-

Poiché è fondato da corporation come Google, Microsoft, Yahoo e Yandex fa schifo :P.

Osservazioni 1.6.4

- Il vocabolario è suddiviso in un insieme di schemi (o types) organizzati in una tassonomia.
- Vale l'ereditarietà.
- Il problema è che queste risorse riflettono gli interessi delle aziende e dei contributori.

2
Ontologie Computazionali

2.1 Introduzione

Definizione 2.1.1: Ontologia

Un *artefatto di ingenieria* costituito da uno specifico *vocabolario* usato per descrivere una certa realtà, aggiungendo un insieme di assunzioni esplicite a proposito del *significato inteso* dal vocabolario stesso.

Osservazioni 2.1.1

- Rappresentazione astratta di concetti e loro relazioni.
- Ontologie formali: rappresentate secondo un formalismo di rappresentazione.
- Finalità: condividere una concettualizzazione comune tra individui, organizzazioni, macchine.

Elementi costituitivi delle ontologie:

- Classi.
- Proprietà.
- Assiomi.
- Individui.

Corollario 2.1.1 Ontologie Formali

Le ontologie formali si basano su linguaggi che permettono di descrivere in maniera esplicita:

- Le caratteristiche delle classi.
- Le caratteristiche delle relazioni tra classi.

Note:-

Questi linguaggi permettono alle macchine di fare inferenze sui concetti e a noi di avere certezza della validità di queste inferenze.

Tipi di ontologie:

- *Ontologie top-level*: concetti fondazionali comuni a tutti i domini (spazio, tempo, ecc.).
- *Ontologie mid-level*: utilizzano il livello fondazionale per definire concetti generali ma non fondazionali: organizzazioni, comunicazione, stati fisici, sistemi di misurazione, ecc.
- *Domain ontologies*: rappresentano i concetti e le relazioni proprie di un dominio specifico.

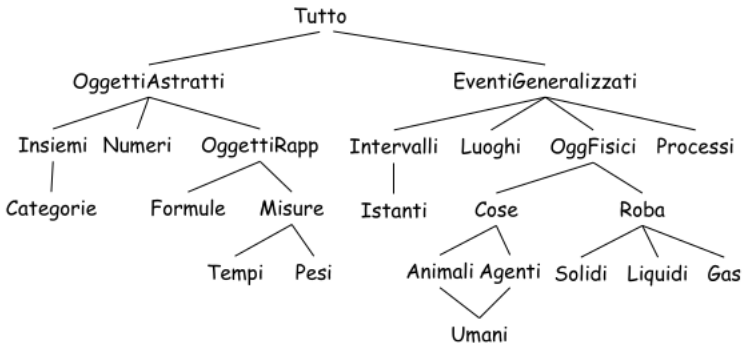


Figure 2.1: Esempio di ontologia top-level.

2.1.1 Conoscenza

Definizione 2.1.2: Conoscenza di Senso Comune

La conoscenza di senso comune (commonsense knowledge) è molto importante per i task che comportano l'interazione con le persone.

Note:-

Per esempio per chatbot e robot.

Definizione 2.1.3: CYC

CYC: enCYClopedic Knowledge, conoscenza enciclopedica. Base di conoscenza finalizzata a rappresentare tutto. Comprende oltre 200.000 concetti.

Note:-

OpenCyc è la versione open source (quindi superiore), ma non è più disponibile direttamente.

Osservazioni 2.1.2

- La base di conoscenza (Knowledge Base, KB) di CYC consiste di:
  - *Termini*, il vocabolario di CYC.
  - *Asserzioni* che mettono in relazione questi termini.
- Queste asserzioni includono:
  - Asserzioni semplici.
  - *Regole*.

Struttura di CYC:

- La KB di Cyc è divisa in molte *microteorie* (microtheories), ciascuna delle quali è costituita da un insieme di asserzioni che condividono le stesse assunzioni.
- Le microteorie si focalizzano su un particolare dominio di conoscenza.
- Il ragionamento avviene all'interno della singola microteoria.
- Questa suddivisione permette al sistema di fare asserzioni che sarebbero apparentemente contraddittorie.

```
"Se Fred possiede un oggetto, possiede anche gli oggetti vicini a quello"
(#$forAll ?X
  (#$forAll ?Y
    (#$implies
      (#$and
        (#$owns #$Fred ?X)
        (#$owns #$Fred ?Y))
      (#$near ?X ?Y))))

"Se Fred possiede una bicicletta, è una bicicletta rossa"
(#$implies
  (#$owns #$Fred #$Bike001)
  (#$colorOfObject #$Bike001 #$RedColor))
```

Figure 2.2: Esempi di regole.

Domanda 2.1

Come ragiona CYC?

- Il *motore inferenziale* di CYC è in grado di effettuare la deduzione logica (modus ponens, modus tollens, quantificazione universale e esistenziale) e i meccanismi di inferenza tipici dell'IA (ereditarietà, classificazione automatica).
- La dimensione della base di conoscenza (200.000 termini e dozzine di asserzioni riguardanti ogni termine) hanno richiesto la messa a punto di tecniche speciali per affrontare la complessità.

Applicazioni di CYC:

- Modellazione della conoscenza.
- Apprendimento e pattern recognition.
- Assistenti intelligenti.
- Sicurezza delle reti.
- Basi di dati.

Definizione 2.1.4: SUMO

Ontologia non più mantenuta perché incorporata in altri progetti.
SUMO (Suggested Upper Merged Ontology) è un'ontologia di alto livello che incorpora un insieme di modelli. È scritta in KIF (Knowledge Interchange Format).

SUMO contiene:

- *Termini*: indivisui e classi, con relazioni gerarchiche (instance e subclass).
- *Connettivi AND e OR*.
- *Quantificazione e implicazione*.

Osservazioni 2.1.3

- Di ogni classe SUMO descrive le caratteristiche attraverso un insieme di assiomi.
- Tali assiomi sono espressi utilizzando le relazioni contenute nell'ontologia.

Posizione della classe nella gerarchia: *PermantentResidence* è una sottoclasse di *Residence*

```
(subclass PermanentResidence Residence)
```

Documentazione: *PermanentResidence* è una *Residence* in cui una persona abita

```
(documentation PermanentResidence EnglishLanguage "A &%Residence where
  people live, i.e.  where people have a &%home.")
```

Assiomi (regole) che valgono per il concetto: Data una certa PermanentResidence (?RESIDENCE), esiste una persona (?PERSON) per cui quella ?RESIDENCE è una casa

```
(=>
   (instance ?RESIDENCE PermanentResidence)
   (exists (?PERSON)
      (home ?PERSON ?RESIDENCE)))
```

Figure 2.3: Esempio di classe.

Note:-

Si possono descrivere relazioni di parti.

```
(instance properPart AsymmetricRelation)
(instance properPart TransitiveRelation)
(subrelation properPart part)

(=>
   (instance ?ROOM Room)
   (exists (?BUILD)
      (and
         (instance ?BUILD Building)
         (properPart ?ROOM ?BUILD))))

(documentation Room EnglishLanguage "A &%properPart of a &%Building
  which is separated from  the exterior of the &%Building and/or other
  &%Rooms of the &%Building by walls.")
```

Figure 2.4: Esempio di relazione part-of.

Note:-

Il browser per consultare SUMO si chiama *Sigma*. No, non metterò un meme brainrot su Sigma.

2.1.2 Altre Ontologie

Definizione 2.1.5: Ontologie Lightweight

Le ontologie leggere sono, normalmente, semplici tassonomie, senza assioni e con poche relazioni. Facilmente standardizzabili.

Corollario 2.1.2 WordNet

Rete di concetti rappresentati da insiemi di termini (detti synset). Relazione di sussunzione tra concetti (synset).

Definizione 2.1.6: Ontologie Large-Scale

Sono ontologie di grandi dimensioni:

- Possono essere ottenute tramite l'estrazione automatica di concetti da testi (Dbpedia e YAGO).
- Dall'integrazione di risorse (YAGO e YAGO2) diverse, incluse risorse lessicali.
- Dal lavoro di una comunità di utenti (CYC), via crowd sourcing.

Note:-

Date le dimensioni, gli strumenti di indicizzazione e di accesso acquisiscono grande importanza: spesso vengono pubblicate online e come tali integrate in altre applicazioni.

Osservazioni 2.1.4

- Per l'accesso ai concetti dell'ontologia, è importante l'integrazione tra di essi e il linguaggio naturale.
- L'integrazione avviene:
  - Attraverso la documentazione, cioè associando a ogni concetto una sua descrizione informale in linguaggio naturale.
  - Associando ai concetti una entry lessicale in una risorsa linguistica esterna.

Definizione 2.1.7: DBPedia

Iniziativa di ricerca iniziata nel 2007. DBPedia punta a estrarre contenuti strutturali dal progetto wikipedia. DBPedia consente agli utenti di estrarre relazioni e proprietà associate a risorse di wikipedia.

Definizione 2.1.8: Linked Open Data

Dati pubblicamente disponibili (Open), pubblicati secondo il paradigma dei Linked Data. I dataset risiedono nella rete, formando il Linked Open Data (LOD) Cloud.

```
(instance properPart AsymmetricRelation)
(instance properPart TransitiveRelation)
(subrelation properPart part)

(=>
   (instance ?ROOM Room)
   (exists (?BUILD)
      (and
         (instance ?BUILD Building)
         (properPart ?ROOM ?BUILD))))

(documentation Room EnglishLanguage "A &%properPart of a &%Building
  which is separated from  the exterior of the &%Building and/or other
  &%Rooms of the &%Building by walls.")
```

Figure 2.5: Linguaggi per descrivere ontologie.

2.1.3 Relazioni tra Classi

Definizione 2.1.9: Sottoclasse

Tutti gli elementi della sottoclasse sono elementi della (sovra)classe, ma non viceversa.

Corollario 2.1.3 Classi Disgiunte

Due classi sono disgiunte se non hanno elementi in comune.

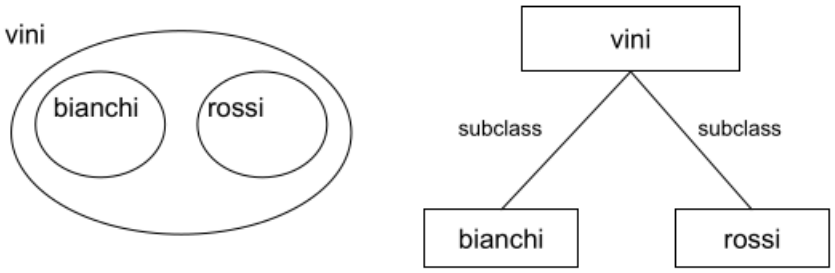


Figure 2.6: Esempio di classi disgiunte.

Corollario 2.1.4 Scomposizione Esaustiva

Due o più classi sono scomposizione esaustiva di una classe se tutti gli elementi della classe appartengono a una di esse

Note:-

{Statunitensi, Canadesi, Messicani} sono la scomposizione esaustiva di Nordamericani[a].

[a] Trump non approva questo elemento, è palese che il Canada sia il 51°esimo stato

Corollario 2.1.5 Partizione

Scomposizione esaustiva con classi disgiunte.

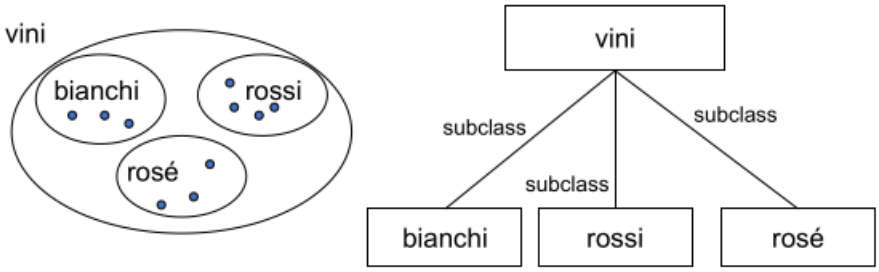


Figure 2.7: Esempio di partizione.

Definizione 2.1.10: Logiche Descrittive

Le logiche descrittive sono:

- Orientate alla classificazione.
- Basate sulla relazione di sottoclasse (sussunzione).
- Completezza e trattabilità computazionale.
- Sono la base del Progetto Web Semantico.

Osservazioni 2.1.5

- Nelle logiche descrittive si distinguono le definizioni di concetti dalle asserzioni sugli individui fatte utilizzando quei concetti:
  - *T-Box*: Terminologia, cioè definizione di concetti generali.
  - *A-Box*: Asserzioni su singoli individui.

Definizione 2.1.11: Ruoli

- I ruoli costituiscono il mezzo per mettere in relazione i concetti.
- In CLASSIC, il predicato che permette di esprimere il concetto di ruolo è FILLS.
- Tipicamente, si possono porre restrizioni numeriche sui ruoli (Atleast, Atmost).
- Un motore inferenziale (reasoner) usa queste informazioni per effettuare ragionamenti.

Terminologia vs. Asserzioni:

- La terminologia è un insieme di assiomi logici su classi e proprietà:
  - Sussunzione tra concetti (subclass).
  - Relazioni generiche in cui gli elementi di determinate classi rivestono un ruolo.
- Data la terminologia, si fanno asserzioni su un insieme di individui.
  - Le asserzioni devono essere coerenti con la terminologia (non si può dire che uno scapolo è sposato).

Domanda 2.2

Cosa si chiede a un sistema basato su DL?

- *Instance checking*: verificare se un certo individuo (nella A-Box) appartiene a una classe.
- *Relation checking*: verificare se vale una certa relazione tra classi.
- *Subsumption*: verificare se una classe è un sottoinsieme di un'altra classe.
- *Concept consistency*: verificare che le definizioni e le loro conseguenze non siano contraddittorie.

2.2 Il Linguaggio OWL

Definizione 2.2.1: OWL2

OWL 2 (Web Ontology Language) è un linguaggio per creare ontologie per il Web Semantico con un significato definito formalmente.

Osservazioni 2.2.1

- Le ontologie scritte in OWL 2 contengono classi, proprietà, individui, e letterali; sono scritte in documenti il cui formato è definito dal Web Semantico.
- Le ontologie OWL 2 possono essere utilizzate in associazione con documenti RDF, anzi sono essere stesse normalmente codificate come documenti RDF.

Ragionamento automatico:

- OWL si basa su logiche computazionali tali che la conoscenza espressa nell'ontologia può essere oggetto di ragionamento automatico da parte di *software specifici (reasoner)* che ne verificano la non contradditorietà e sono in grado di rendere esplicita la conoscenza implicita contenuta nell'ontologia.
- La sintassi RDF/XML per OWL è normata da una specifica: "OWL 2 RDF Mapping". Ci sono molte altre sintassi ma questa è l'unica che qualsiasi strumento software per OWL deve essere in grado di gestire.

Struttura di un documento OWL:

- OWL 2 non fornisce strumenti che descrivano in maniera prescrittiva la struttura di un documento OWL.
- In particolare, non c'è modo per specificare che una determinata informazione (per esempio, il codice fiscale di una persona) deve essere necessariamente presente.

Elementi dell'ontologia:

- *Entità*: gli elementi che si riferiscono al mondo reale.
- *Assiomi*: le asserzioni generali contenute nell'ontologia (per esempio, il concetto di persona è un concetto più specifico di quello di essere vivente).
- *Espressioni*: combinazioni di entità che vanno a formare entità più complesse.

Tipi di entità:

- Gli oggetti sono individui.
- Le categorie sono le classi.
- Le relazioni sono le proprietà.

Le proprietà sono suddivise in:

- Object properties: che collegano individui ad individui (come una persona al suo coniuge).
- Datatype properties: che assegnano un dato a un individuo (per esempio l'età a una persona).
- Annotation properties: che contengono commenti e descrizioni sulle entità.

Definizione 2.2.2: Assiomi

Gli assiomi, in OWL2, possono essere dichiarazioni, assiomi sulle classe, sugli oggetti o sulle proprietà dei dati, definizioni di tipi di dati, chiavi, asserzioni (anche chiamate fatti) e assiomi su annotazioni.

Note:-

In OWL2 ogni IRI deve essere dichiarato nell'ontologia.

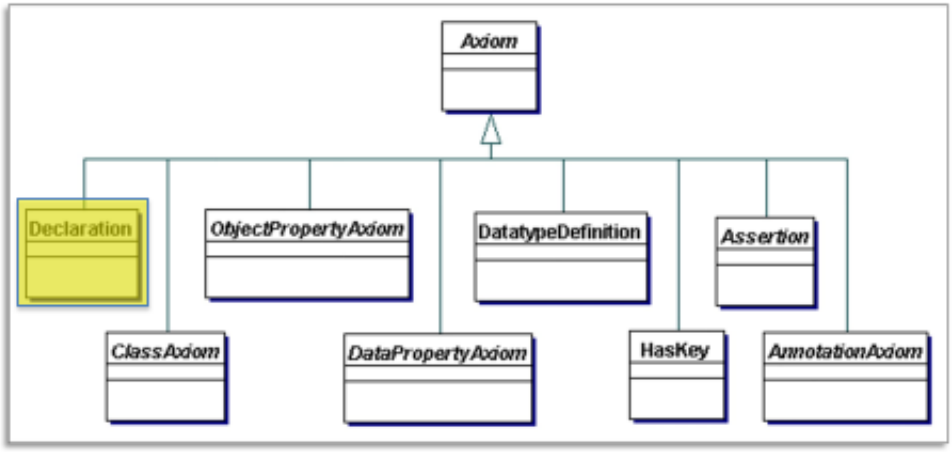


Figure 2.8: Assiomi.

Ci sono 2 tipi di dichiarazioni:

- Che un certo IRI fa parte dell'ontologia.
- A che tipo di entità appartiene l'IRI:
  - Class.
  - Datatype.
  - Object property.
  - Data property.
  - Annotation property.
  - An individual.

Tipi di relazioni:

- *SubClassOf*: ogni istanza di una classe è anche un'istanza di un'altra classe[1]
- *EquivalentClasses*: diverse classi sono equivalenti tra di loro.
- *DisjointClasses*: classi che non hanno nessuna istanza in comune.
- *DisjointUnion*: una scomposizione esaustiva.

Esempio 2.2.1

- Sottoclasse:
  - SubClassOf(a:Boy a:Child).
  - Boy è sottoclasse di Child.
- Classi equivalenti:
  - EquivalentClasses(a:CatOwner a:PadroneDiGatti).
  - Le due classi sono entrambe sottoclasse l'una dell'altra.
- Disgiunzione:

[1] Costruisce una gerarchia di classi.

  - DisjointClasses(a:Cat a:Dog).
  - Cat e Dog sono classi disgiunte.
- Disjoint Union:
  - DisjointUnion(a: Person a:Child a:Adult).
  - Person ha come sottoclassi le due classi disgiunte Child e Adult.

Note:-

In questi esempi è stata usata la *functional style syntax*.

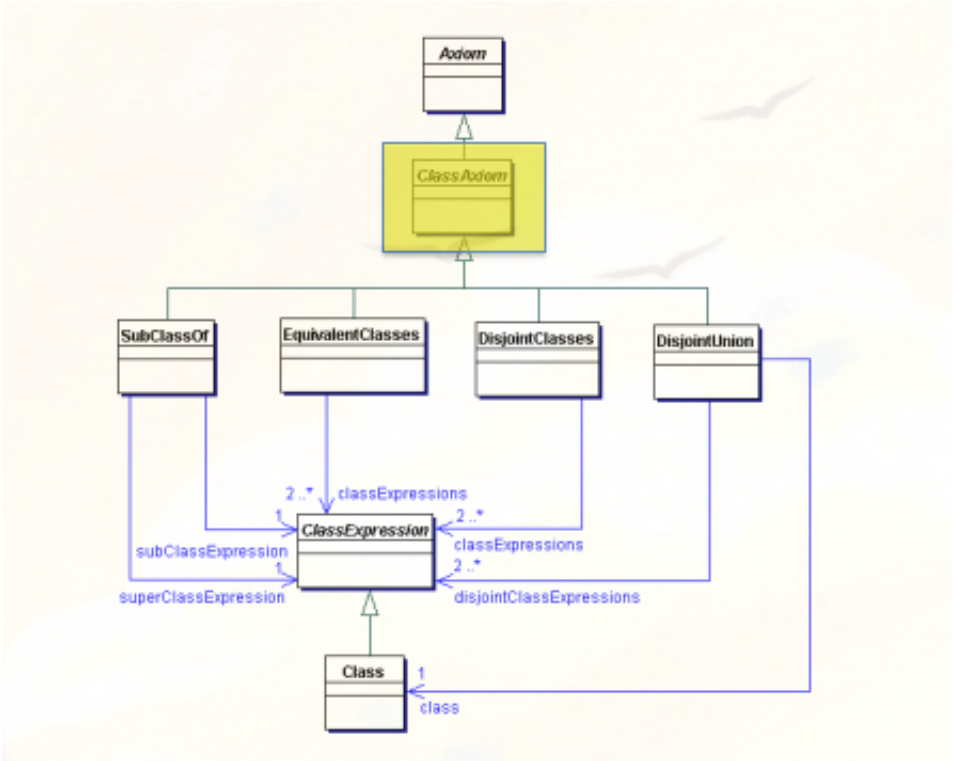


Figure 2.9: ClassAxiom

Definizione 2.2.3: Espressioni

In OWL2, classi e proprietà sono usate per costruire *espressioni di classe* (o descrizioni).

Note:-

Per esempio "Professoressa" può essere visto come l'unione di "Docente" e "Donna".

OWL2 fornisce un insieme di operatori per definire le classi:

- *Connettivi booleani*.
- *Quantificazione universale ed esistenziale*.
- *Restrizioni numeriche*.
- *Enumerazione di individui*.

2.2.1 Definizioni e Sintassi

- Si definisce una classe dichiarando che essa appartiene al tipo Classe di OWL.

:Person rdf:type owl:Class;

- Mary appartiene alla classe Person.

:Mary rdf:type :Person .

- John appartiene alla classe Father.

:John rdf:type :Father .

- Assiomi di sottoclasse.

:Woman rdfs:subClassOf :Person .

- Due classi possono essere dichiarate equivalenti.

:Person owl:equivalentClass :Human .

- Un insieme di classi possono essere dichiarate disgiunte.
- Si definisce una Object Property asserendo che essa appartiene al tipo ObjectProperty di OWL.

:hasSpouse rdf:type owl:ObjectProperty ;

- Le proprietà di tipo Object Property rappresentano relazioni tra classi (e quindi, si asseriscono degli individui).

:John :hasWife :Mary

- Le data property hanno come domain una classe e come range un tipo di dato.

:hasAge rdf:type owl:DatatypeProperty ;

Definizione 2.2.4: Restrizioni

Le restrizioni sono uno dei meccanismi principali per definire nuove classi a partire da quelle esistenti.

Esistono due tipi di restrizioni:

- Restrizioni su *classi* mediante operatori insiemistici (intersezione-and, unione-or, complemento-not).
- Restrizioni poste su *proprietà* (esistenziale, universale, sulla cardinalità).

Definizione 2.2.5: Intersezione

L'intersezione permette di definire una classe come intersezione di due classi.

Definizione 2.2.6: Complemento

Una classe può essere definita come complemento di un'altra.

Necessario e Sufficiente:

- Le classi definite come equivalenti a un certo insieme di restrizioni sono denominate classi definite.
- Le restrizioni individuano le condizioni necessarie e sufficienti per l'appartenenza alla classe.
- Senza l'utilizzo del costrutto EquivalentTo si possono associare a una classe solo proprietà necessarie ma non sufficienti.

Note:-

Solo le classi definite permettono determinate forme di ragionamento.

Definizione 2.2.7: Classi Enumerate

È possibile definire una classe come enumerazione di individui.

Descrivere le proprietà:

- Simmetriche.
- Funzionali.
- Inverse.
- Riflessive.
- Transitive.

Osservazioni 2.2.2 Modellazione

- Non confondere la relazione di sottoclasse con la relazione mereologica part-of (un motore non è un'automobile).
- L'ontologià deve essere leggibile:
  - Gerarchica.
  - Attribuire alle entità nomi che hanno senso secondo il senso comune o per l'esperto.
  - Associare domain e range alle proprietà.

2.3 Ragionamento Automatico

Ragionamento su classi:

2.3.1 Introduzione

- La gerarchia delle classi asserita mediante gli assiomi di sottoclasse può essere diversa da quella inferita.
- Il reasoner può inferire delle relazioni di sussunzione a partire dalla descrizione delle classi.

Note:-

Il reasoner ha sempre ragione.

Inserimento di individui:

- Gli individui vengono inseriti direttamente in una classe.
- Dopo averli inseriti è possibile predicarne le caratteristiche, cioè specificarne le proprietà che li mettono in relazione con altri individui o con un dato (class e property assertions).
- Il ragionamento può eventualmente ricollocare un certo individuo in un'altra classe.
- Inserendo un individuo con le caratteristiche di una certa classe, l'individuo viene classificato come appartenente alla classe dal reasoner.
- Anche se è stato collocato manualmente in una classe più generale.

Definizione 2.3.1: Assunzione di Mondo Aperto

Le inferenze effettuate dal reasoner possono sembrare poco intuitive in alcuni casi. Il ragionamento sulle ontologie OWL segue l'assunzione di mondo aperto:

- l fatto che un'informazione non sia rappresentata nel sistema non determina che essa sia assunta falsa.

Corollario 2.3.1 Quantificatore Universale (Only)

Se un individuo viene collocato in due o più classi che hanno un Quantificatore only il reasoner rileva un'inconsistenza.

Corollario 2.3.2 Classe Primitiva

La classe primitiva può avere condizioni necessarie associate a essa tramite lo slot SubClassOf.

Corollario 2.3.3 Classe Definita

La classe definita è una classe equivalente a un insieme di condizioni necessarie e sufficienti, specificate mediante EquivalentTo.

Osservazioni 2.3.1

- Only non significa almeno uno.
- Il quantificatore universale si applica a tutti.

Ragionamento su data properties:

- I valori assegnati a un individuo per una certa data property devono appartenere al data type specificato come range.
- Se il datatype è diverso da quello previsto, il reasoner rileva un'inconsistenza.

Restrizioni su data properties:

- È possibile porre una restrizione su una data property.
- Usando quantificatori e cardinalità tuttavia si esprime un vincolo sul numero delle relazioni, non sul valore che assume la proprietà.
- Per creare classi definite in molti casi si usano i pattern che individuano un range specifico di valori rispetto al tipo di dato.
- Per esempio le stringhe che cominciano con una certa lettera, gli interi superiori a un certo valore, ecc...

Classi primitive e ragionamento:

- Le classi primitive non permettono ai reasoner di collocare gli individui nelle classi in modo automatico.
- Esse permettono ai reasoner di individuare inconsistenze con le asserzioni sugli individui.

2.3.2 Esportare le Inferenze su Protege

- Protege contiene un tool per esportare le inferenze effettuate dal reasoner attivo in una nuova ontologia.
- È utile se si vogliono utilizzare le inferenze in un ambiente in cui non è disponibile un reasoner.
- Vengono esportate sia le triple che costituiscono l'ontologia sia le triple che rappresentano le inferenze.

Passi per l'esportazione:

1. Selezione del tipo di inferenze che si intendono esportare.
2. Selezione di elementi aggiuntivi:
   - Annotazioni.
   - Asserzioni contenute nell'ontologia.
3. Inserimento dell'IRI della nuova ontologia.
4. Percorso e file in cui salvare la nuova ontologia.
5. Selezione della codifica.

Inferenze esportate:

- Il set di inferenze di base non esporta tutte le inferenze che sono visualizzate nell'editor.
- Le inferenze rappresentate come triple nella nuova ontologia si dicono materializzate.
- Nei casi reali, il trade-off tra tempo di esportazione (e lo spazio occupato) e livello di dettaglio delle inferenze dipende dall'utilizzo che si intende fare dell'ontologia esportata.

Corollario 2.3.4 Object Property Characteristics

Il flag *Object* (o data) Property Characteristics esporta le inferenze fatte sulle caratteristiche delle object properties.

Corollario 2.3.5 Class Assertion (Individual Types)

Il flag *Individual types* esporta le inferenze sugli individui basate sulla gerarchia delle classi.

Corollario 2.3.6 Property Assertions

Il flag *Property assertions* esporta le inferenze sulle proprietà degli individui.

2.3.3 Serializzazione di OWL

Dall'ontologia astratta alla sua rappresentazion:

- L'ovale al centro rappresenta la nozione astratta di ontologia, che può essere pensata sia come struttura astratta sia come grafo RDF.
- La sintassi di OWL nelle specifiche è descritta come sintassi *astratta*.
- L'ontologia può essere serializzata in varie sintassi.
- La semantica dell'ontologia (parte in basso) può essere specificata direttamente (Direct Mapping, semantica model-teoretica) oppure a seguito della conversione in un grafo RDF (RDF-based semantics).

Definizione 2.3.2: Mapping verso RDF

Il mapping di OWL verso RDF consiste nel rappresentare l'ontologia come triple RDF. È specificata come un insieme di mapping, dove l'operatore T mappa un'ontologia OWL O in un grafo RDF T(O).

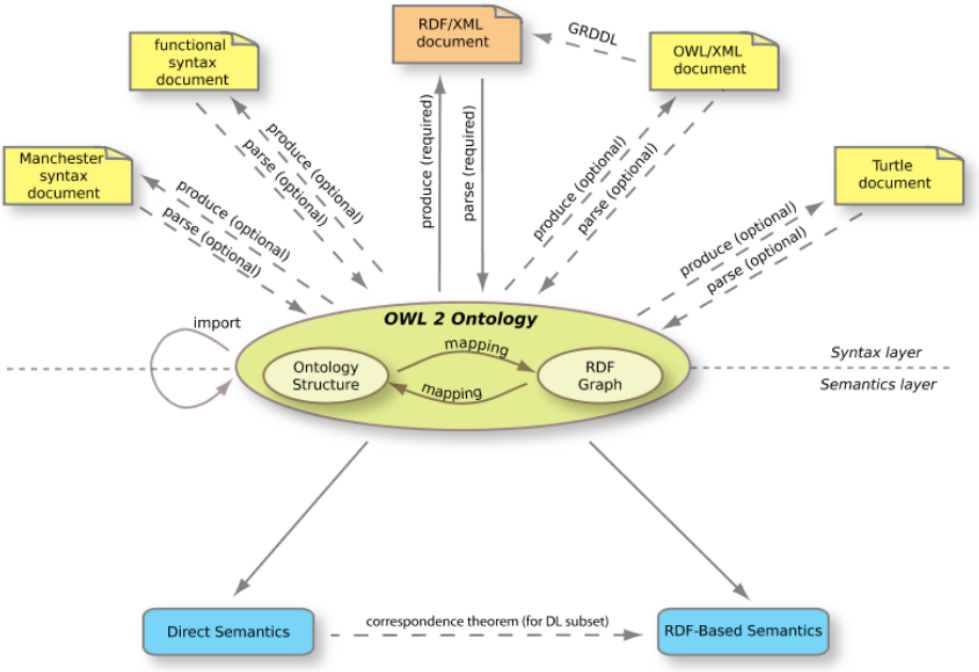


Figure 2.10: Specifiche di OWL.

OWL2 vs. OWL1:

- Property chains.
- Datatypes più ricchi e data ranges.
- Restrizioni sulla cardinalità.
- Proprietà asimmetriche, riflessive e disgiuntive.
- Più capacità di annotazione.
- Chiavi.

Profili di OWL2:

- OWL2 EL: progettato per grandi ontologies. Limitazioni:
  - No negazione e disgiunzione, no quantificazione universale sulle proprietà.
  - No proprietà inverse.
- OWL QL (Query Language): concepito per l'integrazione con le basi di dati relazionali. Limitazioni:
  - No quantificazione esistenziale in una class expression.
  - No property chains.
- OWL RL (Rule Language): concepito per applicazioni che richiedono l'utilizzo del ragionamento automatico senza sacrificare troppa espressività.

3 Dalle Ontologie ai Grafi

3.1 SPARQL

3.1.1 Introduzione

Definizione 3.1.1: SPARQL

SPARQL è un linguaggio di interrogazione per RDF. Permette di creare nuovi grafi con le triple esatte.

Le query possono essere diverse:

- Selezione di triple secondo un semplice pattern.
- Query complesse costruite con filtri, aggregatori, path expressions, ecc.
- Query booleane (ASK).

Risultati di SPARQL:

- Il risultato di una query può essere un insieme di triple o uno o più grafi RDF.
- SPARQL supporta i formati più comuni: XML, CSV, TSV, JSON.

3.1.2 Query

Corollario 3.1.1 SPARQL Endpoint

Le query vengono indirizzate a un indirizzo HTTP che ospita un endpoint SPARQL. L'endpoint esegue le query su uno o più dataset contenuti in uno store di triple.

Note:-

Molte Linked Data Platforms (LDP) hanno un'interfaccia in cui è possibile inserire manualmente le query.

WHERE:

- Nella clausola WHERE si trovano una o più triple che contengono variabili ?v
- Le triple formano un pattern che corrisponde a un insieme di grafi.

- Il pattern viene unificato con le tripl presenti nel repository rdf.
- Il risultato nella query è dato da tutte le occorrenze delle variabili contenute nella clausola WHERE (proiezione) nei grafi trovati.

> **Corollario 3.1.2** Filtri
>
> I filtri sono elementi opzionali che si possono includere nella clausola WHERE per selezionare un sottoinsieme nell'insieme dei risultati.

Operazioni sui risultati:

- OrderBy: ordina i risultati delle query sulla base di una delle variabili.
- Distinct.
- Group: estrae e conta gli individui che hanno determinati requisiti.
- Blank node: seleziona tutte le entità situate da qualche parte.
- Count.

4 Ontology Engineering

4.1 Introduzione

Definizione 4.1.1: Ontology Engineering

Criteri e linee guida per progettare ontologie interoperabili e ben fondate sul piano concettuale. Criteri e linee guida per progettare ontologie interoperabili e ben fondate sul piano concettuale. Approcci collaborativi allo sviluppo di ontologie.

4.1.1 OntoClean

Definizione 4.1.2: OntoClean

Metodo ontologico basato sull'analisi delle tassonomie delle classi.

Note:-

Non è prescrittiva rispetto alle entità del mondo reale, ma fornisce proprietà con cui caratterizzare le classi.

Meta-Proprietà in OntoClean:

- *Identity:* proprietà intrinseca che identifica un tipo di oggetti.
  - Sortal: classe di oggetti che possono essere identificati nello stesso modo.
  - Viene ereditata dalle sottoclassi.
- *Unity:* proprietà di un tipo di oggetto di essere unitario.
  - Whole: classe di oggetti che sono tali solo in quanto costituiti da parti collegate tra loro da relazioni strutturali.
  - Viene ereditata dalle sottoclassi.
- *Rigidity:* proprietà di un tipo di oggetto che non è soggetta a cambiamenti.
  - Solo l'antirigidità viene ereditata.
- *Dependence:* proprietà di un tipo di oggetti di dipendere da un altro per la propria definizione.

Definizione 4.1.3: Phase Sortals

Proprietà che attraversa una serie di fasi. Sono:

- Indipendenti.
- Antirigidi.
- Hanno un'identità.

4.1.2 Neon Methodology

Definizione 4.1.4: Neon Methodology

Metodologia orientata agli aspetti collaborativi nello sviluppo e nel mantenimento di network di ontologie. Si articola in un insieme di 9 scenari, associati a specifiche attività e documenti.

Scenari:

1. Dalle specifiche all'implementazione.
2. Riusare e reingegnerizzare risorse non ontologiche.
3. Riusare risorse ontologiche.
4. Riusare e reingegnerizzare risorse ontologiche.
5. Riusare e unire risorse ontologiche.
6. Riusare, unire e reingegnerizzare risorse ontologiche.
7. Riusare design pattern ontologici (ODPs).
8. Ristrutturare risorse ontologiche.
9. Localizzare risorse ontologiche.

Definizione 4.1.5: Ontology Design Patterns (ODPs)

Sorta di mattoncini per la creazione di ontologie secondo schemi (pattern) convidisi.

4.1.3 Altre Ontologie

Definizione 4.1.6: DOLCE

Ontologia orientata alla cognizione e al linguaggio.

Osservazioni 4.1.1 Concetti Importanti

- *Perdurante* → Evento, ha natura temporale.
- *Endurant* → Partecipanti, non ha natura temporale.

Definizione 4.1.7: Prov

Concepita per l'interscambio sul Web di informazioni riguardanti la provenienza delle entità. Descrive l'origine delle entità intesa soprattutto come processi che hanno determinato la creazione di quelle entità:

- La provenienza degli oggetti è rilevante per determinarne il possesso (inteso come diritti) e l'affidabilità.

I metadati propriamente detti delle entità e la descrizione degli agenti sono affidati a altre ontologie:

- FOAF (agenti).
- Dublin Core (entità).

In Prov:

- Un *agente* prende il ruolo in un'attività in modo che gli si possa assegnare un certo grado di responsabilità.
- *Attività:* sono il come le entità esistono e come i loro attributi cambiano per farle diventare nuove entità.
- Un *ruolo* è una descrizione della funzione di un'entità in un'attività. Specificano la loro relazione.

Problematiche:

- Come rappresentare i ruoli?
  - Classi dell'ontologia (sottoclassi di Ruolo).
  - Classe generica con etichette linguistiche.
- Classi dell'ontologia:
  - Vantaggi: ragionamento.
  - Svantaggi: non aperto, non modulare.
- Classe generica:
  - Vantaggi: modulare.
  - Svantaggi: più difficile condurre inferenze.

Definizione 4.1.8: LODE

L'ontologia LODE (Linking Open Descriptions of Events) rappresenta gli eventi (per esempio eventi storici) con un vocabolario molto semplice.

Osservazioni 4.1.2 LODE

- Non rappresenta i ruoli ma solo i partecipanti e la collocazione nel tempo e nello spazio degli eventi.
- Orientata alla costruzione di grandi data set (descrizione di eventi).
- Per ogni classe, corrispondenza con DOLCE.

4.2 Ontologie e Risorse Linguistiche

Note:-

Approfondita nella seconda parte del corso "Tecnologie del Linguaggio Naturale".

4.2.1 FrameNet

Definizione 4.2.1: FrameNet

Si tratta di un catalogo di frame linguistici su varie parti del discorso che rappresentano un frame con una serie di ruoli (complementi) associati.

Note:-

La struttura a frame di FrameNet si presta a essere utilizzata insieme allo schema a ruoli (utilizzato per esempio negli Ontology Design Pattern) per rappresentare la partecipazione all'azione. Ogni Frame Element corrisponde a un ruolo nello schema di azione.

Descrizione di Eventi e FrameNet:

- Repository di frame che descrivono una situazione in termini di partecipanti con un certo ruolo.
- Ogni frame corrisponde a un insieme di entità lessicali (tipicamente verbi ma anche preposizioni).
- Non esiste un insieme predefinito di ruoli (come per esempio in VerbNet.
- Conversione diretta in un pattern basato su ruoli.

4.2.2 WordNet

Definizione 4.2.2: WordNet

WordNet è una risorsa linguistica che consiste in un lessico organizzato secondo relazioni di significato. Gli elementi del lessico (sostantivi, verbi, aggettivi, ecc.) sono raggruppati in insiemi denominati synset (synonym set):

- Ogni synset corrisponde a un significato.
- L'idea è che i termini che fanno parte di uno stesso synset siano sinonimi.

Corollario 4.2.1 Synset

Nello stesso synset si trovano più termini. Lo stesso sostantivo può essere collocato in posizioni del tutto diverse della rete.

Altre relazioni:

- Meronimia: per i sostantivi, oggetto che è parte di un altro.
- Implicazione: per i verbi.
- Troponimia: per i verbi.

WordNet e ontologie:

- Nell'ontologia SUMO le classi sono associate ai synset di WordNet.
- Sigma Browser è uno strumento per cercare concetti nell'ontologia SUMO usando come chiavi di ricerca i termini della lingua inglese.
- Il browser funziona perché a ogni concetto di SUMO corrispondono uno o più termini di WordNet.
- Cercando in WordNet, si trovano tutti i significati della parola cercata e, per ogni significato, il concetto di SUMO corrispondente.

Ci sono due modi in cui un synset di WordNet può corrispondere a un concetto SUMO:

- Il concetto SUMO corrisponde esattamente al significato del synset (equivalent mapping).
- Il concetto SUMO include al suo interno ('sussume') anche il significato del synset (subsumption mapping).

4.3 SKOS

4.3.1 Contesto

Con l'avvento dei Linked Data, fanno il loro ingresso nel Web Semantico molte risorse di tipo tassonomico.

Esempi di tesauri e tassonomie:

- Iconclass.
- VIAF.
- ULAN.
- ACM Computing Classification System.

Definizione 4.3.1: VIAF

VIAF (Virtual International Authority File) è un Authority file internazionale:

- Ottenuto a partire dagli Authority file nazionali.
- Permette di accedere a una determinata entità dal suo nome (in una lingua nazionale) per ottenerne la denominazione in tutti gli authority in cui è presente.

Corollario 4.3.1 Authority File

Un authority file è una risorsa che contiene un elenco strutturato di valori da utilizzare per riferire un insieme di entità.

4.3.2 Simple Knowledge Organization System (SKOS)

Definizione 4.3.2: SKOS

L'utilizzo di vocabolari di diverse provenienze, tipico dei Linked Data, annulla le relazioni tra i concetti proprio delle ontologie e si apre alle differenze. Simple Knowledge Organization Systems (SKOS): recommendation del W3C per la creazione di vocabolari di vario tipo: lessici, thesauri, ecc.

Note:-

Il suo scopo è quello di facilitare e uniformare la pubblicazione dei vocabolari RDF come linked data.

Domanda 4.1

Cosa si può fare con SKOS?

- I concetti possono essere identificati mediante URI.
- Etichettati con stringhe in una o più lingue naturali.
- Assegnare notazioni.
- Documentare con vario tipo di note.
- Collegare concetti ad altri concetti e organizzarli in gerarchie informali.
- Aggregare i concetti in schemi, ordinarli, etichettarli o mapparli.

Corollario 4.3.2 SKOS Primer

Documento che riporta esempi di utilizzo di SKOS.

Concetti ed etichette:

- I concetti sono identificati con gli URI.
- Ai concetti sono associate delle etichette lessicali:
  - prefLabel.
  - altLabel.
- Indicizzazione.

Relazioni semantiche tra concetti:

- Relazione generica: related.
- Relazione specifica/generale: narrowed/broader.
- Anche transitive.

Documentazione:

- Associare concetti alla loro documentazione.
- Definizione: definition.
- Esempio: example.
- Cambiamenti: historyNote e changeNote.

Schemi di concetti:

- Uno schema di concetti e una risorsa strutturata.
- Schema di concetti: ConceptScheme.
- Appartenenza di un concetto a uno schema: inScheme.
- (Ri)utilizzare concetti in uno schema diverso richiamando quello di origine.

4.4 SWRL

4.4.1 Specifiche

Regole:

- Antecedente e conseguente: l'antecedente viene valutato su un determinato dataset.
- Dichiarative: risultano in una nuova asserzione, possono solo aggiungere nuove proprietà.
- Produzione: hanno come conseguente un'azione.

Definizione 4.4.1: Specifica di SWRL

Ufficiosa, limitata a causa di problematiche di tipo computazionale (OWL diventa non più decidibile) ed è supportata solo da alcuni reasoner.

SWRL Tab in Protege:

- Si basa su un engine esterno (Drools).
- Implementazione migliorata dell'algoritmo RETE per il pattern matching.
- Le regole vengono impostate in Drools ed eseguite. I risultati possono essere importati in modo permanente nell'ontologia.

Definizione 4.4.2: Rule Interchange Format

Standard del W3C, per facilitare la sintesi e l'integrazione di sistemi di refole.

Note:-

RIF può anche essere usato nei linked data.