

---

ANNO ACCADEMICO 2024/2025

---

# Modellazione Concettuale per il Web Semantico

---

Teoria

Altair's Notes



---

DIPARTIMENTO DI INFORMATICA

---



<b>CAPITOLO 1</b>	<b>WEB OF DATA: I LINGUAGGI</b>	<b>PAGINA 5</b>
1.1	Introduzione Perché Studiare il Semantic Web e Linked Data? — 5 • Obiettivi del Corso — 5	5
1.2	Dalle Reti Semantiche alle Ontologie I Limiti della Logica Classica — 6 • Reti Semantiche — 7 • Reti Semantiche e Logica — 9 • Sistemi a Regole — 14	6
1.3	Il Sistema SNePS Interrogare la Rete — 18 • Rappresentazione delle Credenze e Deduzioni — 19 • Inferenze — 19	15
1.4	RDF Introduzione — 22 • Vocabolari e Grafi — 24	21
1.5	Turtle Introduzione — 24 • Sintassi — 25	24

<b>CAPITOLO 2</b>	<b>ONTOLOGIE COMPUTAZIONALI</b>	<b>PAGINA 28</b>
-------------------	---------------------------------	------------------



# Premessa

## Licenza

Questi appunti sono rilasciati sotto licenza Creative Commons Attribuzione 4.0 Internazionale (per maggiori informazioni consultare il link: <https://creativecommons.org/version4/>). Sono basati sulle lezioni del corso "Model-  
lazione Concettuale per il Web Semantico" (2020-2021<sup>1</sup>) 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 contenente 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 definizioni

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.

---

<sup>1</sup>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 *monotona*.
- 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<sup>a</sup>.
- I *valori di verità* non sono adatti a rappresentare gli aspetti quantitativi che caratterizzano il mondo reale

<sup>a</sup>Da premesse false discendono conseguenze vere.

#### Definizione 1.2.2: Monotonicità

La logica del primordine è monotona: 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  $\Rightarrow$  *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  $\Rightarrow$  *logiche bayesiane*, incorporano il ragionamento probabilistico.
- Monotonicità  $\Rightarrow$  *default logics*, belief revision.
- Indecidibilità  $\Rightarrow$  insiemi decidibili della logica del primordine.
- Inefficienza delle dimostrazioni  $\Rightarrow$  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 rappresentano *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

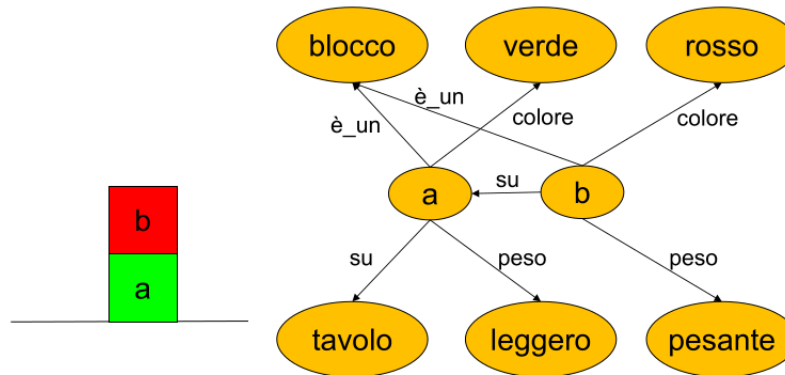


Figure 1.2: Grafo relazionale del "Mondo dei Blocchi", un dominio giocattolo utilizzato nel mondo dell'AI.

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*

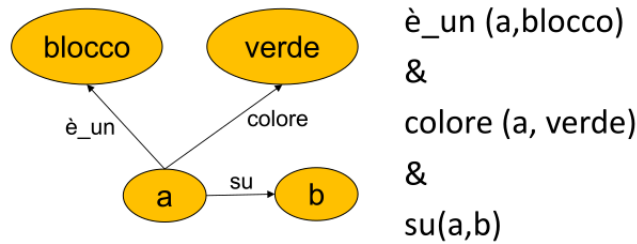


Figure 1.3: Grafo relazionale in relazione alla logica del primordine.

**Note:-**

Alcune relazioni possono coinvolgere più di due entità.

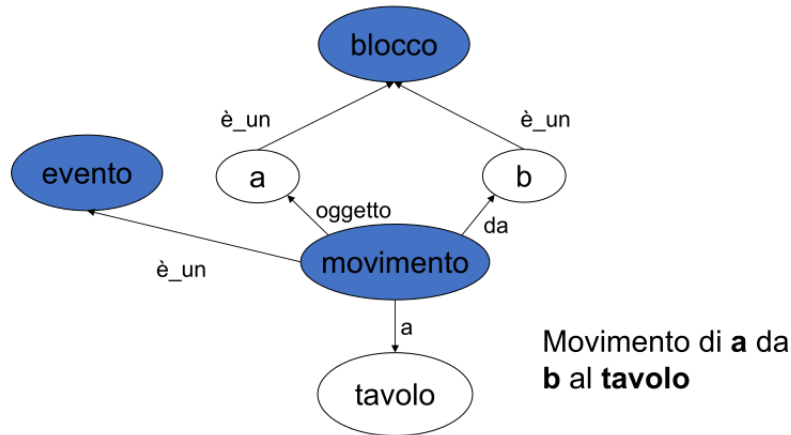


Figure 1.4: La relazione “movimento” che unisce a, b e tavolo diventa un termine cioè un nodo.

#### Definizione 1.2.5: Reti Semantiche Proporzionali

Le *reti semantiche proporzionali* 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 proporzionali posso avere l'espressività della logica del primordine una volta introdotti connettivi, variabili, quantificatori, ecc.
- Anche l'inferenza nelle reti proporzionali 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.

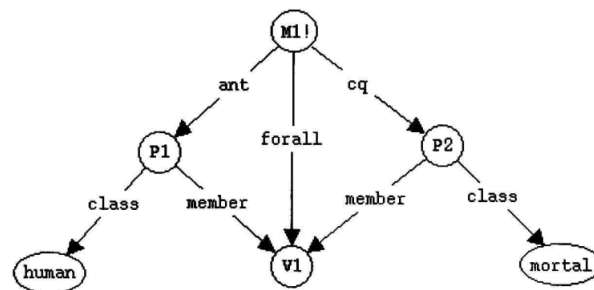


Figure 1.5: Tutti gli uomini sono mortali.

#### 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.

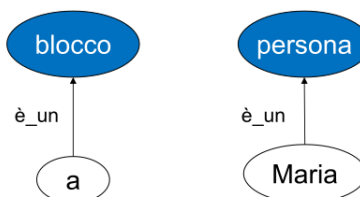


Figure 1.6: I nodi colorati rappresentano tipi di entità (classi), i nodi bianchi rappresentano entità singole (individui).

**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.

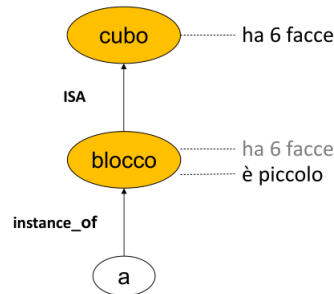


Figure 1.7: ISA e instance\_of.

**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?  $\Rightarrow$  archi ISA.
- x appartiene alla classe Y?  $\Rightarrow$  archi instance\_of.

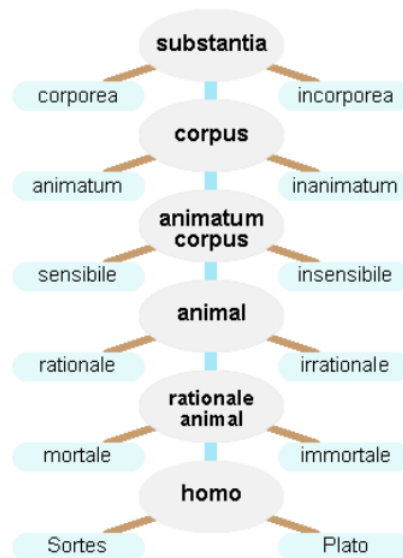


Figure 1.8: Rappresentazione che deriva dalla filosofia aristotelica, confluita nella scolastica, in cui si raffigurano i principali concetti del pensiero dalla sostanza all'uomo.

**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.

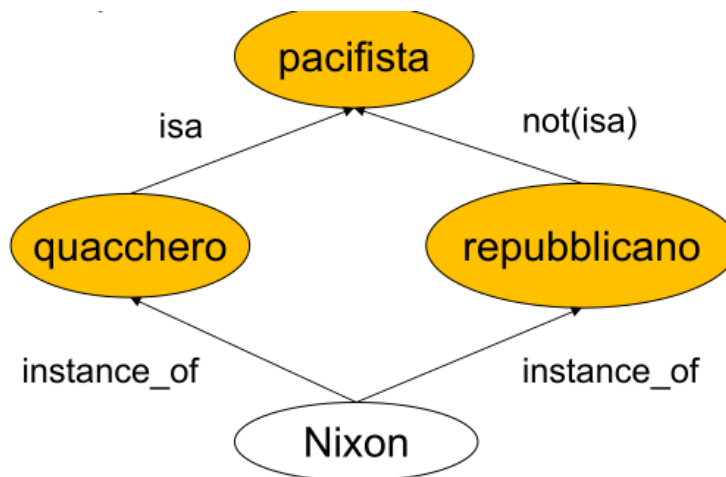


Figure 1.9: Nixon è pacifista o guerrafondaio? In quanto quacchero, lo è; in quanto repubblicano, non lo è (Reiter and Criscuolo, On Interacting Defaults, 1981).

**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.

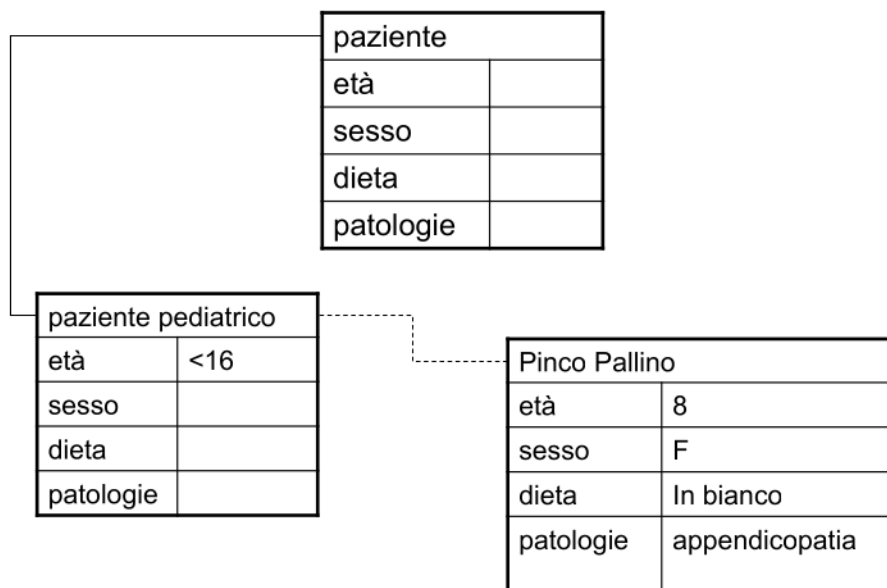


Figure 1.10: Esempio di tassonomia.



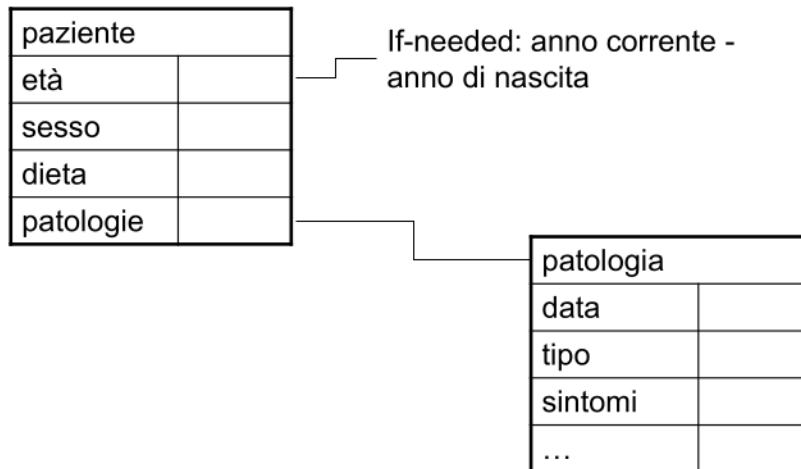


Figure 1.11: Esempio di collegamenti non tassonomici.

#### 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 regole.
- *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.

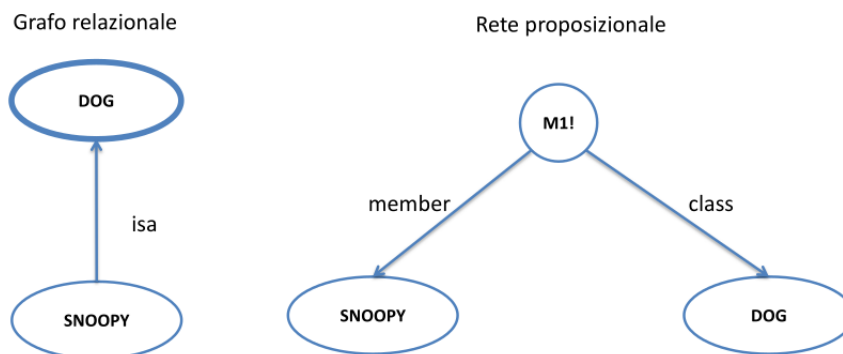


Figure 1.13: Nella rappresentazione di SNwPS (a destra) la proposizione “Snoopy è un cane” è rappresentata da un nodo (M1!). Nessun arco denota una proposizione  $x$  isa (è un)  $y$ , come invece avviene nel grafo relazionale (a sinistra).

### 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 ( $M_i$ ) hanno archi uscenti. Rappresentano proposizioni, incluse le regole di ragionamento, oppure individui.
- I nodi base ( $B_i$ ) 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.

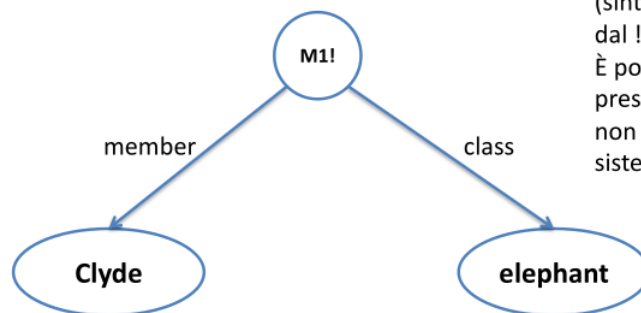
- forall
  - exists
  - min
  - max
  - thresh
  - ant
  - &ant
  - cq
  - dcq
  - arg
  - default
  - if
  - action
  - ...
- Esistono relazioni (archi) predefiniti nel sistema
  - Nella versione più recente i tipi di relazione sono organizzati in una tassonomia

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.



Il nodo M1! è asserito (sintatticamente, è rappresentato dal ! dopo il nome del nodo). È possibile che nel sistema siano presente anche delle proposizioni non asserite, cioè non “credute” dal sistema

**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:

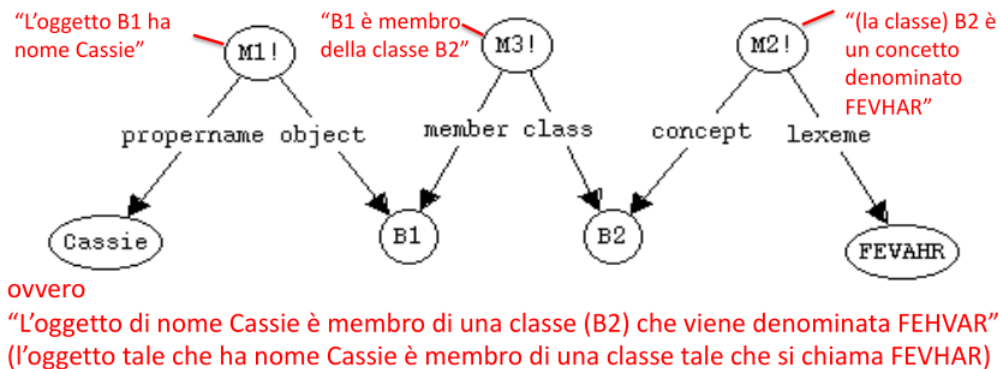
- I nodi base vengono usati per rappresentare individui che soddisfano determinate descrizioni senza dargli un nome.
- Non hanno archi uscenti; cioè non hanno informazioni strutturali. Tutto ciò che si sa di loro è ciò che viene asserito.
- Un nodo base potrebbe essere descritto tale l'espressione "un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

### Esempio 1.3.2 (Nodi di base)

#### USO DEI NODI BASE: ESEMPIO



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*.

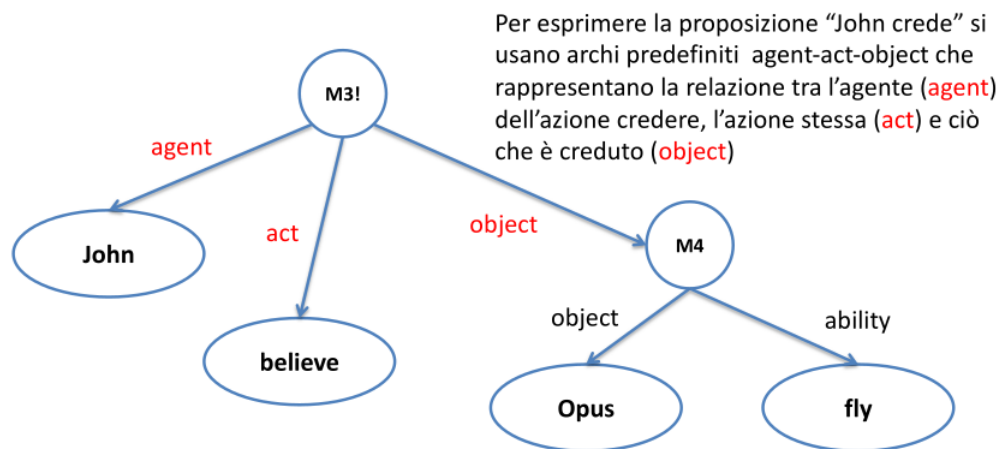
**Osservazioni 1.3.2**

- 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 assertito 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)

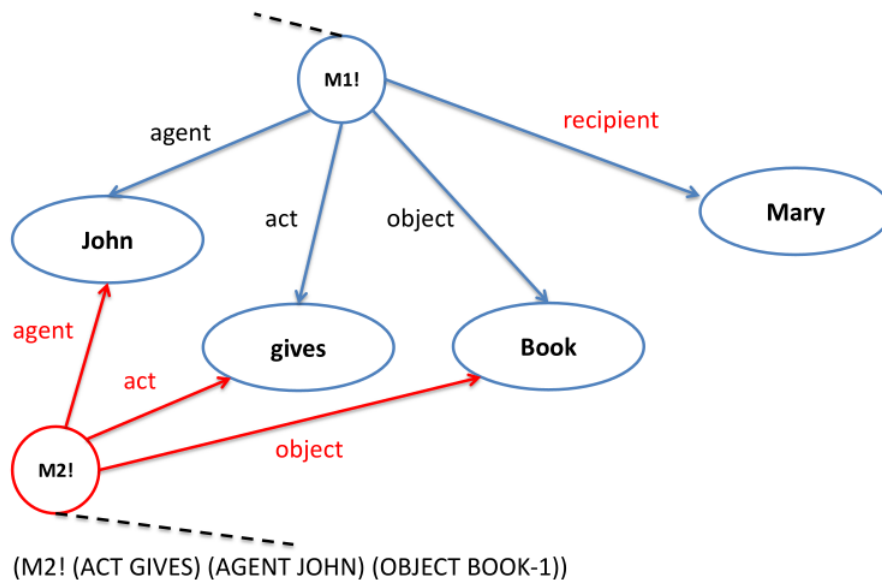
John dà un libro a Mary.

Può essere ridotto a:

(deduce agent john act gives object book-1)

John dà un libro.

(M1! (ACT GIVES) (AGENT JOHN) (OBJECT BOOK-1) (RECIPIENT MARY))



### 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<sup>a</sup> significa "zero o più occorrenze".
- Si legge: isa è uguale a isa composto con un percorso di zero o più percorsi ottenuti da un object-

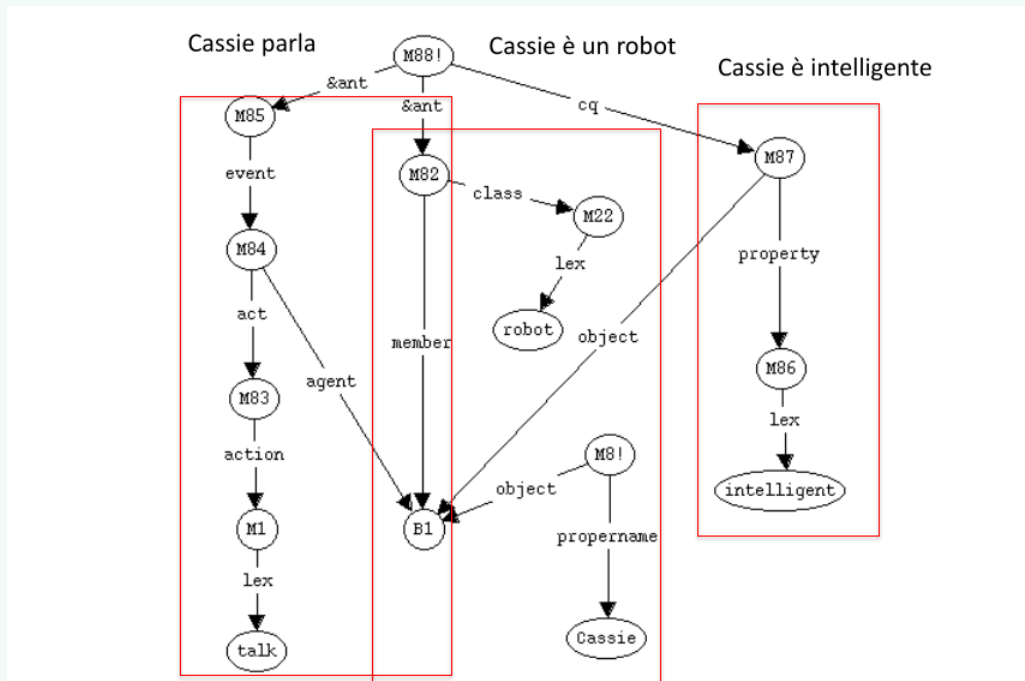
composto con un isa.

<sup>a</sup>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 (proprername) Cassie (quest'ultimo fatto è assertito).
- 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 *nod*i 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.

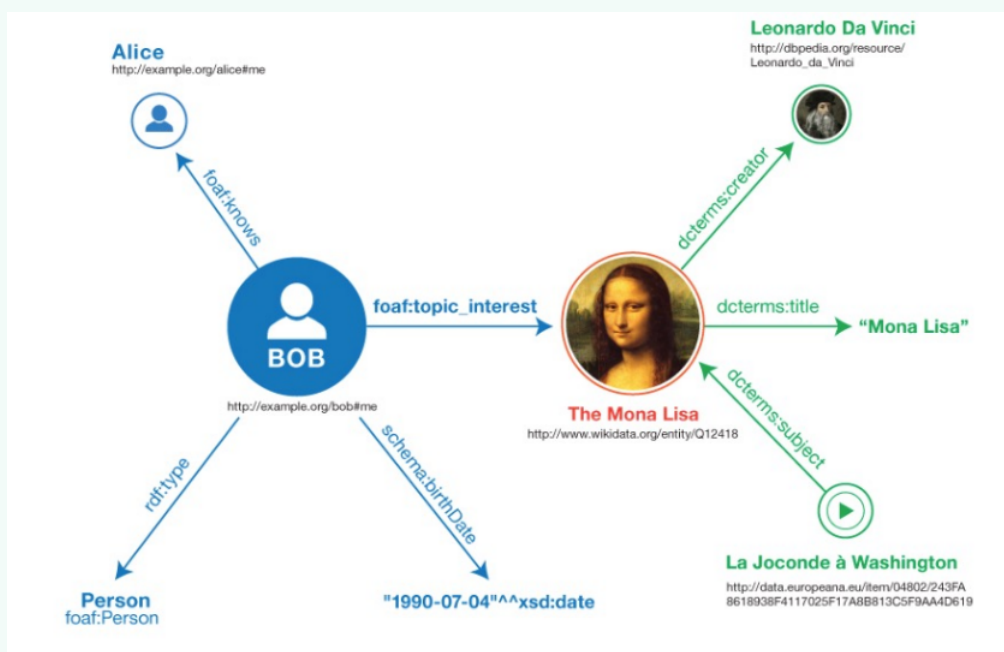
**Definizione 1.4.4: Triple**

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)

Grafo = IRI + 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 concisione 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",

```

Diagram illustrating the Turtle syntax structure:

- prefissi** (prefixes): A group of lines starting with `@prefix` or `@base`.
- triple** (triple): A group of lines representing a single statement, enclosed in angle brackets and separated by semicolons.
- commento** (comment): A line starting with `#` used for annotations.

Figure 1.15: Il formato Turtle.

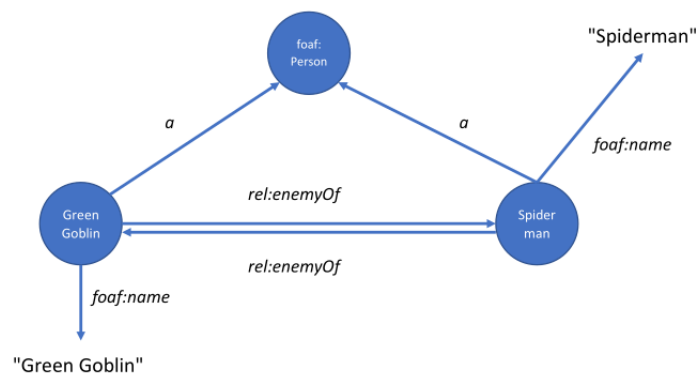


Figure 1.16: Rappresentazione a grafo.

### 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.

- xsd:integer:       "-5"^^xsd:integer     → -5
- xsd:decimal:       "-5.0"^^xsd:decimal   → -5.0
- xsd:double:        "4.2E9"^^xsd:double   → 4.2E9

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.



# 2

## Ontologie Computazionali

