

# IL MODELLO A OGGETTI IN IA

---

Premessa:

- Concetto di *oggetto* riconducibile a settori dell'informatica sviluppatisi storicamente in modo alquanto disgiunto e indipendente:
  - **linguaggi di programmazione**
  - **basi di dati**
  - **intelligenza artificiale**
- Necessità di una migliore **integrazione** tra sottosistemi diversi (basi di dati, pacchetti applicativi, reti di comunicazione, supervisor intelligenti)
- Modello ad oggetti come fattore di **aggregazione** sia a livello concettuale sia realizzativo
- Come perseguire questa integrazione è ancora un problema aperto
- Esiste un insieme di **concetti comuni** per dare una caratterizzazione largamente accettabile del paradigma ad oggetti?
- Importanza dei formalismi per Semantic Web

# CONCETTO DI OGGETTO

---

- **Oggetto** = "Knowledge grain"

per rappresentare **concetti** del mondo reale

*A knowledge representation framework should not merely prescribe how small bits of information ought be represented, but it should also detail how the totality of information ought to be **structured** and organized so that information may be **retrieved** and relevant **inferences** may be drawn **efficiently** (Shastri, 1989)*

# RAPPRESENTAZIONE A OGGETTI VS. RAPPRESENTAZIONE RELAZIONALE

---

Dipendenza dal modo con cui si rappresentano i dati

- **Rappresentazione relazionale: dati raggruppati concettualmente attraverso le loro proprietà**
  - `person (john)`
  - `job (john, employee)`
  - `project (john, cnr1234)`
- **Rappresentazione ad oggetti: i dati sono concettualmente tutti raggruppati all'interno dello stesso oggetto**
  - `object     john: is-a person`
  - `job -> employee`
  - `project -> cnr1234`

# RAPPRESENTAZIONE A OGGETTI VS. RAPPRESENTAZIONE RELAZIONALE

---

- L'utente può accedere a *tutte* le informazioni di un oggetto mediante l'identificatore unico (logico o fisico) dell'oggetto stesso
- Si possono individuare altre proprietà (opzionali).
- Anche se storicamente i linguaggi ad oggetti adottano uno stile di programmazione procedurale, i *paradigmi di rappresentazione* (dichiarativo, procedurale) e *programmazione* (funzionale, imperativo, dichiarativo) possono essere concepiti come *ortogonali* rispetto al concetto di sistema "*object-oriented*"

# APPROCCI

---

- **Approccio Logico o Dichiarativo:**
  - Usato nelle reti semantiche, nella logica terminologica o descrittiva (famiglia KL-ONE).
  - Un oggetto è una congiunzione di **proprietà** e può essere rappresentato come clausole della logica
- **Approccio Procedurale:**
  - Usato nei sistemi di rappresentazione della conoscenza basati su *frames* e nei linguaggi di programmazione ad oggetti
  - Un oggetto è una struttura dati con uno **stato** e un **comportamento** (behaviour)
  - Le proprietà di un oggetto possono essere attributi, relazioni e procedure (slots, subslots, tipi, cardinalità, vincoli)
  - Conoscenza procedurale espressa mediante demoni (attaccati agli attributi) e metodi (attaccati agli oggetti)

# RETI SEMANTICHE

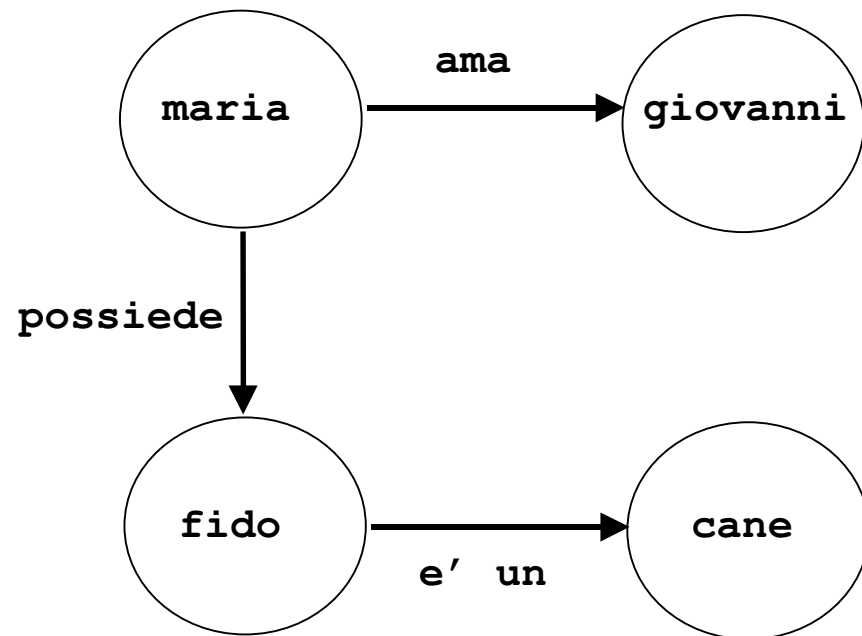
---

(Brachman, 1979)

- Non sono di chiara origine e definizione;
- Particolarmente utili per il linguaggio naturale, descrizione e comprensione di forme, elaborazione di modelli psico-cognitivi.
- Oggetto o **concetto** rappresentato da un nodo
- Relazioni gerarchiche (*is-a*, *instance-of*)
  
- Valori di proprietà rappresentati come nodi legati attraverso archi etichettati
  
- **CONOSCENZA: OGGETTI E RELAZIONI BINARIE**
  - OGGETTI = NODI di un grafo diretto etichettati;
  - RELAZIONI = ARCHI etichettati con una direzione per evitare ambiguità

# ESEMPIO

---



# CARATTERISTICHE

---

- La duplicazione di nodi che rappresentano lo stesso concetto viene sempre evitata.
- NON RIDONDANZA:
  - NODI: singola locazione di memoria
  - ARCHI: strade concettuali, ma anche fisiche (puntatori), per raggiungere in modo efficiente i concetti interessati.
- **Rappresentazione = Conoscenza + Metodi di Accesso** (Newell)
- Corrispondono a una LOGICA DEI PREDICATI CON RELAZIONI BINARIE (senza duplicazione)
  - Rappresentazione binaria:
  - modificabilità;
  - modularità.



# ESEMPIO

---

“giovanni dà un libro a maria”.

- **LOGICA DEI PREDICATI:**

- a1) `dà( giovanni, libro, maria )`;

- a1 può essere considerato come un evento **e1** con le seguenti caratteristiche:

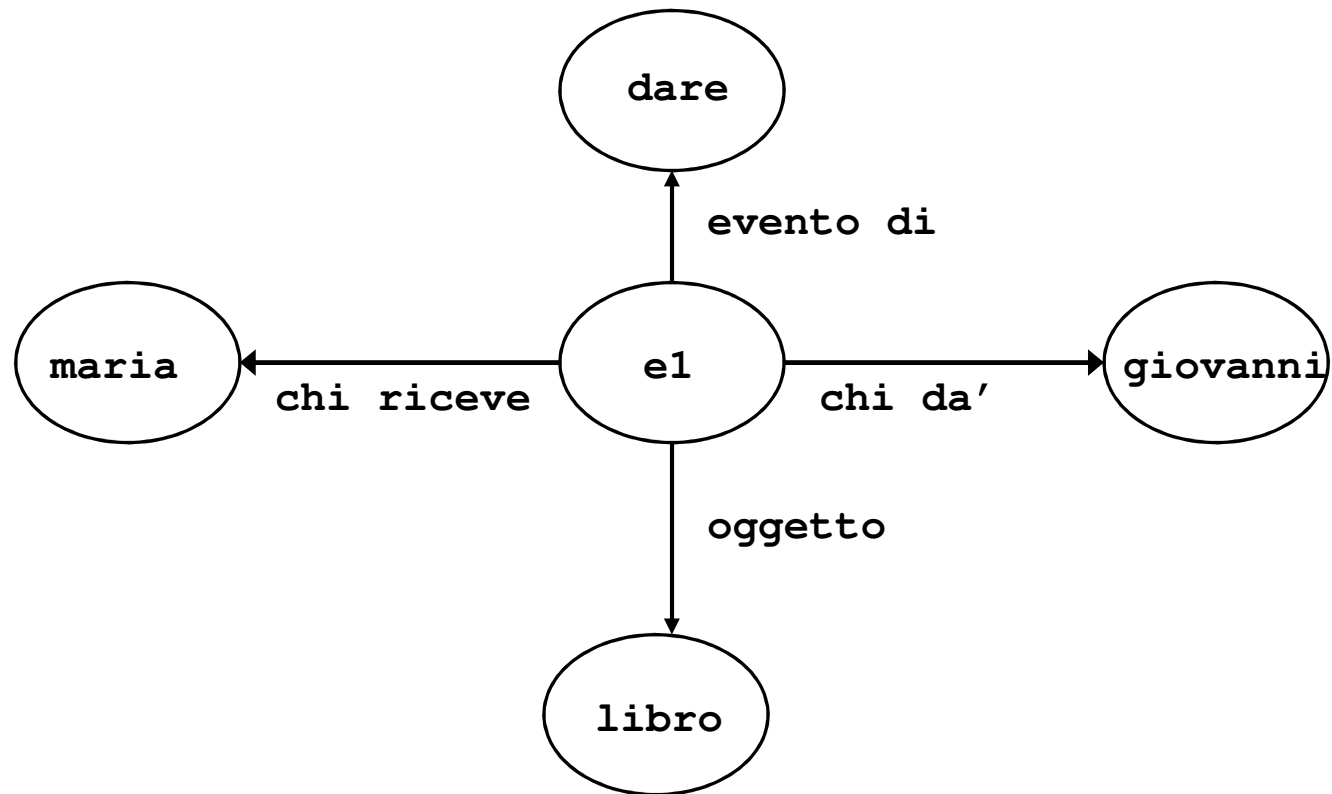
- a2) `evento( e1, dare, giovanni, libro, maria )`;

- oppure alternativamente:

- a3) `evento( e1, dare) and chi-da( e1, giovanni) and chi-riceve( e1, maria) and oggetto( e1, libro)`

# RETI SEMANTICHE

---



# RETI SEMANTICHE

---

- **ESPRIMONO IN MODO IMPLICITO REGOLE PER:**
- Trattare concetti di **ereditarietà** (semplice o multipla ) di proprietà;
- Permettere la sovrascrittura di proprietà ritenute erranee, aggiornabili, o più semplicemente sconosciute in base a nuova/vecchia conoscenza acquisita (**eccezioni** e **default** ).
- Semantica? Spesso riconducibile a frammenti di FOL

# NODI

---

- NODI: possono rappresentare diverse entità semantiche
  - **single entità individuali.**



- **Insiemi (categorie) di entità.**

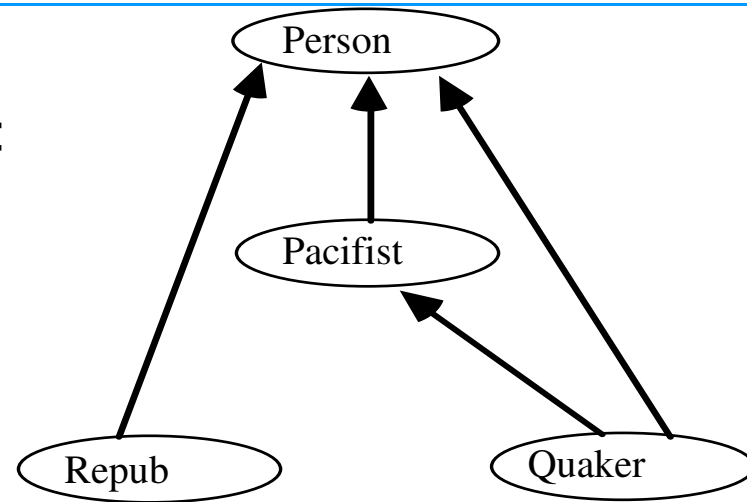


- Netta distinzione fra questi differenti tipi di nodi per ottenere maggiore chiarezza nella notazione.

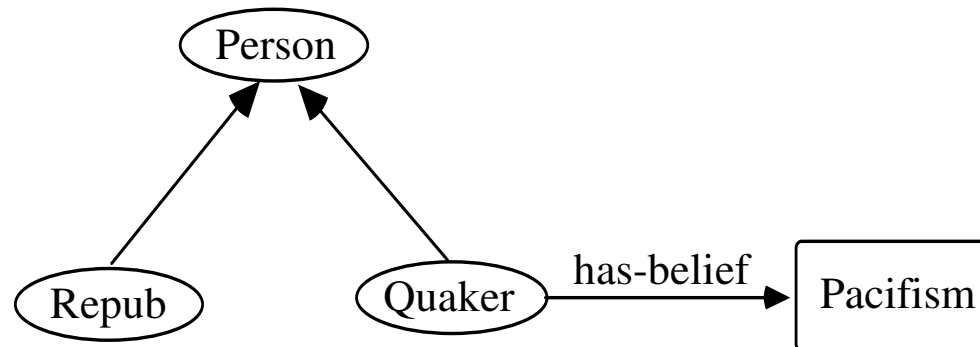
# NODI

Sistemi :

- “di sole categorie (classi)” (e istanze):



- “di categorie e proprietà”:

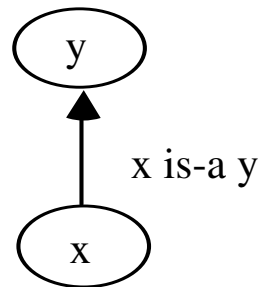


- Rappresentabili in Logica dei Predicati del I Ordine

# TRASLAZIONE IN LOGICA

---

- Se  $x$  è un individuo e  $y$  una classe il legame:



- Può essere interpretato come la formula logica:  
 $y(x)$                       Es: *cane(fred)*
- Se  $x$  e  $y$  sono classi, il legame tra loro può essere interpretato come la formula logica:  
 $\forall z \ x(z) \Rightarrow y(z)$   
Es:  $\forall z \ \textit{cane}(z) \Rightarrow \textit{mammifero}(z)$

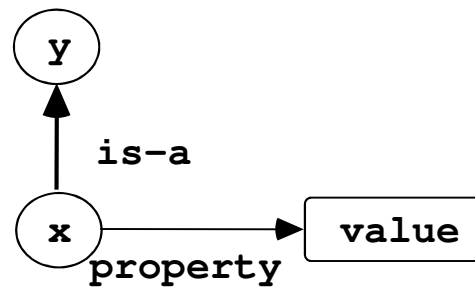
# TRASLAZIONE IN LOGICA

---

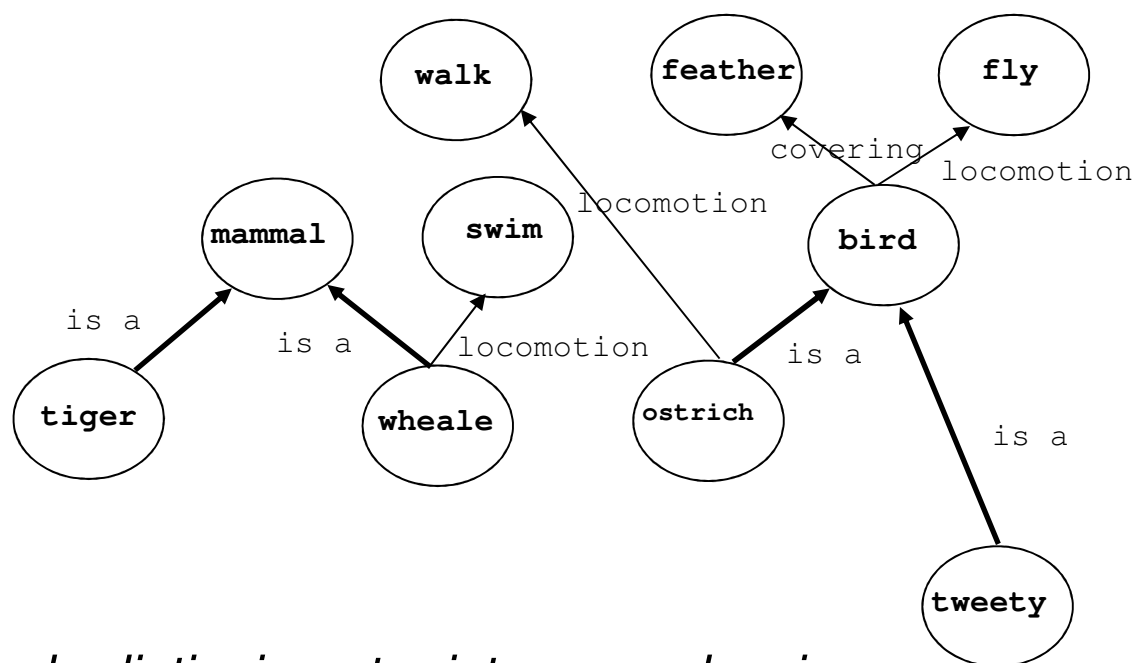
- Se una classe o un individuo ha delle proprietà, queste possono essere traslate in predicati binari

$\forall Z \ y(Z) \Rightarrow \text{property}(Z, \text{value})$   
 $\text{property}(x, \text{value})$

*classe*  
*individuo*



# PROBLEMI DELLE RETI SEMANTICHE



- *Manca la distinzione tra istanze e classi*
  - In realtà, l'unico link *"is-a"* è *"sovraccaricato"* (Woods, 1975)
  - Rappresenta sia la relazione di *appartenenza* ad una classe, sia quella di *sottoclasse*.
  - Richiedono la presenza di un secondo link *"instance-of"* (ma anche *part-of*, etc.)

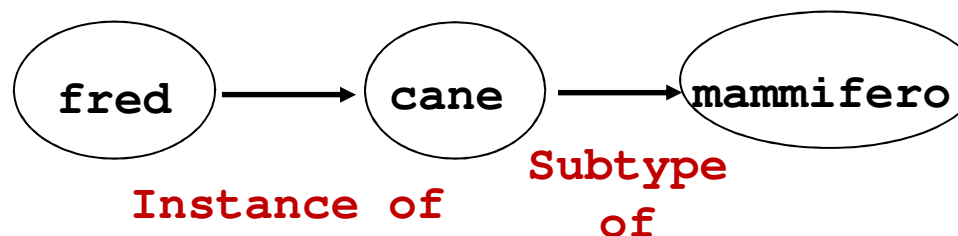


# PROBLEMI DELLE RETI SEMANTICHE

## Significato di "IS-A" ...



- "fred è un cane"
- "il cane è un mammifero".



- Due significati diversi:
  - 1) member-of; instance-of;
  - 2) Is-a; subset-of; subtype-of.
- Non esiste distinzione tra attributi associati a una classe e attributi (ereditati) dalle istanze della classe

# ATTRIBUTI DI CLASSI E ISTANZE

---

## ***Esempio:***

- *Gli elefanti sono una specie in via di estinzione*
- *Clyde è un elefante*
- *Clyde è in via di estinzione*
  
- *Gli elefanti sono numerosi*
- *Clyde è un elefante*
- *Clyde è numeroso*

## ***Semantica formale:***

- Manca una semantica formale (almeno nelle prime versioni)
- Aumento del potere espressivo e semantica:
  - ***grafi concettuali*** (Sowa, 1984)
  - famiglie dei ***linguaggi terminologici*** (alla KL-ONE)

# Logiche terminologiche o descrittive

---

Possono essere viste come:

1. *Evoluzioni “logiche”* di linguaggi di KR basati sulle nozioni di *frame* e *reti semantiche*
  2. *Contrazioni* della logica del prim'ordine (FOL) per ottenere migliori proprietà computazionali
- Verso gli anni '80 si ha una sterzata verso la logica delle reti semantiche
  - Il processo di formalizzazione consiste nel ...
    - riformulare i costrutti secondo i canoni della logica
    - eliminare i costrutti che non si prestano a tale riformulazione (*default* ed *eccezioni*)

# Esempio

---

La seguente è una formula di una delle LT:

(and paper  
    (atmost 2 author)  
    (atleast 2 author))[paper372]

equivalente a:

paper(paper372)  $\wedge$   
 $\exists x$  author(paper372, x)  $\wedge$   
 $\exists y$  author(paper372, y)  $\wedge$   $x \neq y$   $\wedge$   
author(paper372, z)  $\Rightarrow$  (z = x)  $\vee$  (z = y)

# Concetti e ruoli

---

- Ogni LT è caratterizzata da operatori per la costruzione di termini di due tipi:
  - *concetti* (corrispondenti a relazioni unarie)  
con operatori *and*, *or*, *not*, *all*, *some*, *atleast*, *atmost*, ... per la costruzione di concetti complessi
  - *ruoli* (corrispondenti a relazioni binarie)

# FRAMES

---

- Molto simili al formalismo delle reti semantiche
- Utilizzati assieme a regole in gran parte degli ambienti evoluti per sistemi esperti
- Introdotti per la prima volta da Minsky nel 1975 (Minsky, 1975)
  - *"Quando si incontra una nuova situazione (o si fa un sostanziale cambiamento nel modo di concepire un certo problema) si seleziona dalla memoria una struttura chiamata **frame** che è uno **schema concettuale** precedentemente memorizzato che viene adattato per rappresentare la realtà ..."*
  - *"Un **frame** è una struttura dati per rappresentare una **situazione stereotipata**, come essere in un certo tipo di camera o andare al compleanno di un bambino. **Attaccate a ogni frame** ci sono vari tipi di **informazioni** . Alcune sono relative a **come utilizzare il frame**, altre a cosa può richiamare in seguito ..."*

# FRAMES

---

**Concetto intuitivo:** Modulo di conoscenza che descrive qualcosa in termini delle sue *proprietà*

- Ha un *nome* che identifica ciò che è descritto
- Ognuna delle proprietà possedute è rappresentata mediante una coppia *slot/valore* (KRL, NETL, KEE, KAPPA, ART)

- *Esempio:*

Nome	Slot	Valore ( <i>default</i> )
Volpe	Is-a	Piccolo-Animale
	Colore	Fulvo
	Furbizia	Molto-sviluppata
Volpe_albina	Is-a	Volpe
	Colore	Bianco

# FRAMES

---

- Differentemente dai linguaggi terminologici, l'**appartenenza a una categoria** (classe o concetto) non viene caratterizzata mediante proprietà **necessarie e sufficienti**, ma nei termini di una maggiore o minore **somiglianza** rispetto a membri tipici della categoria detti **prototipi**
- In generale, tutti i sistemi a frame permettono di ragionare su classi di oggetti usando **rappresentazioni prototipali** che, valide in linea di massima, devono essere adattate e modificate (**eccezioni**)
- Le descrizioni dei prototipi non sono nemmeno condizioni necessarie
- I valori per **default** vengono usati come valori **provvisori e approssimativi**, in assenza dei valori reali
- Come nelle reti semantiche, le **connessioni** fra un frame e gli altri e i rispettivi ruoli che devono rappresentare nella conoscenza globale sono determinati da valori attribuiti a specifici slots.



# SUBSLOT O FACET

---

- Gli slot possono contenere *informazioni aggiuntive* sui valori associati (default, tipo, quantità, etc.)
- *Informazioni procedurali*:
  - Connesse anch'esse agli slot del frame (*procedural-attachment*)
  - Indicano come utilizzare tale frame, come trovare i valori per gli slots, etc.
- *Integrazione di conoscenza dichiarativa e procedurale*
  - Le procedure (*demoni*) sono spesso individuate da particolari “facet” che attivano la procedura associata durante:
    - la ricerca di valori di uno slot           if-needed
    - l'aggiunta di valori                       if-added
    - la cancellazione di valori               if-removed

# ESEMPIO

---

Nome	Attributo	Facet	Valore
• Resistenza	Is-a	Uguale	Componente
	Valore	Unit	Ohm
	Precisione	Dominio	(1 5 10)
		Default	5
	Potenza	If-needed	Request

Istanza di nome R1:

Nome	Attributo	Facet	Valore
• R1	Is-a	Uguale	Resistenza
	Valore	Uguale	5000
	Precisione	Uguale	1

# ORGANIZZAZIONE DI OGGETTI E INFERENZE

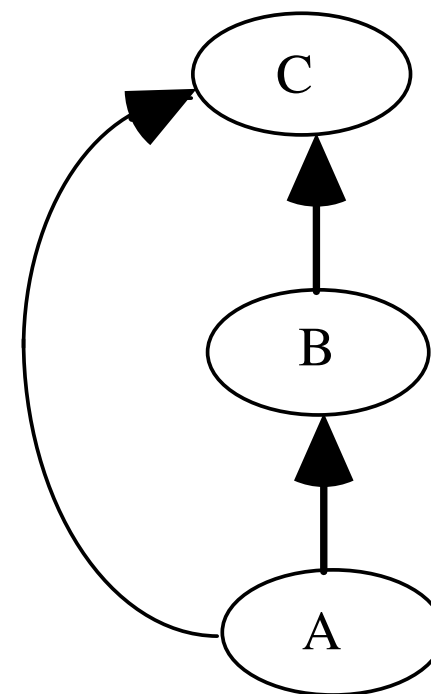
---

- Gli oggetti vengono organizzati in una *gerarchia* o *ordinamento parziale* detta *gerarchia di ereditarietà* o *tassonomia*
- *Oggetti* rappresentati da nodi
- *Relazioni gerarchiche* tra oggetti rappresentate connettendo nodi via archi “is-a” o “instance-of”
- Cattura il principio base di organizzazione sia delle reti semantiche, dei sistemi a frames e dei sistemi a oggetti
- La *tassonomia* viene utilizzata per memorizzare informazioni al livello più appropriato di generalità, rendendole automaticamente disponibili agli oggetti più specifici mediante il meccanismo di *ereditarietà*

# TASSONOMIA

---

- La *tassonomia* può essere *costruita*:
  - *direttamente* dal programmatore, asserendo i legami “is-a” tra coppie di concetti (reti semantiche, sistemi a frames e a oggetti)
  - *automaticamente* inferendo i legami “is-a” tra coppie di concetti (ad esempio a partire dalla loro struttura nella famiglia KL-ONE dei linguaggi terminologici)
- Costruita la tassonomia, interrogazioni sui legami gerarchici tra oggetti presenti nella tassonomia sono risolte attraverso l'applicazione della chiusura transitiva

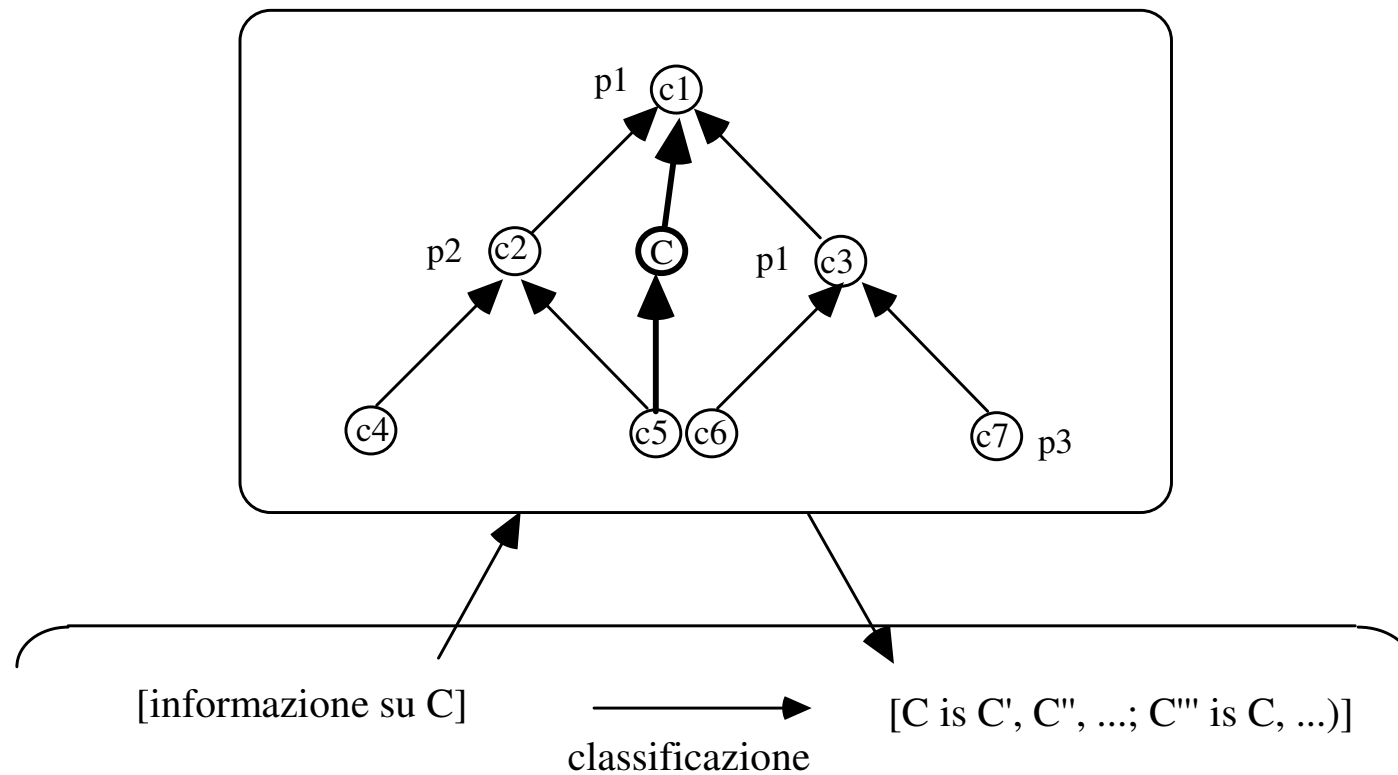


# FORME DI INFERENZA

---

- **Tre tipi di inferenze:**
  - *classificazione*
  - *riconoscimento*
  - *ereditarietà*

# CLASSIFICAZIONE

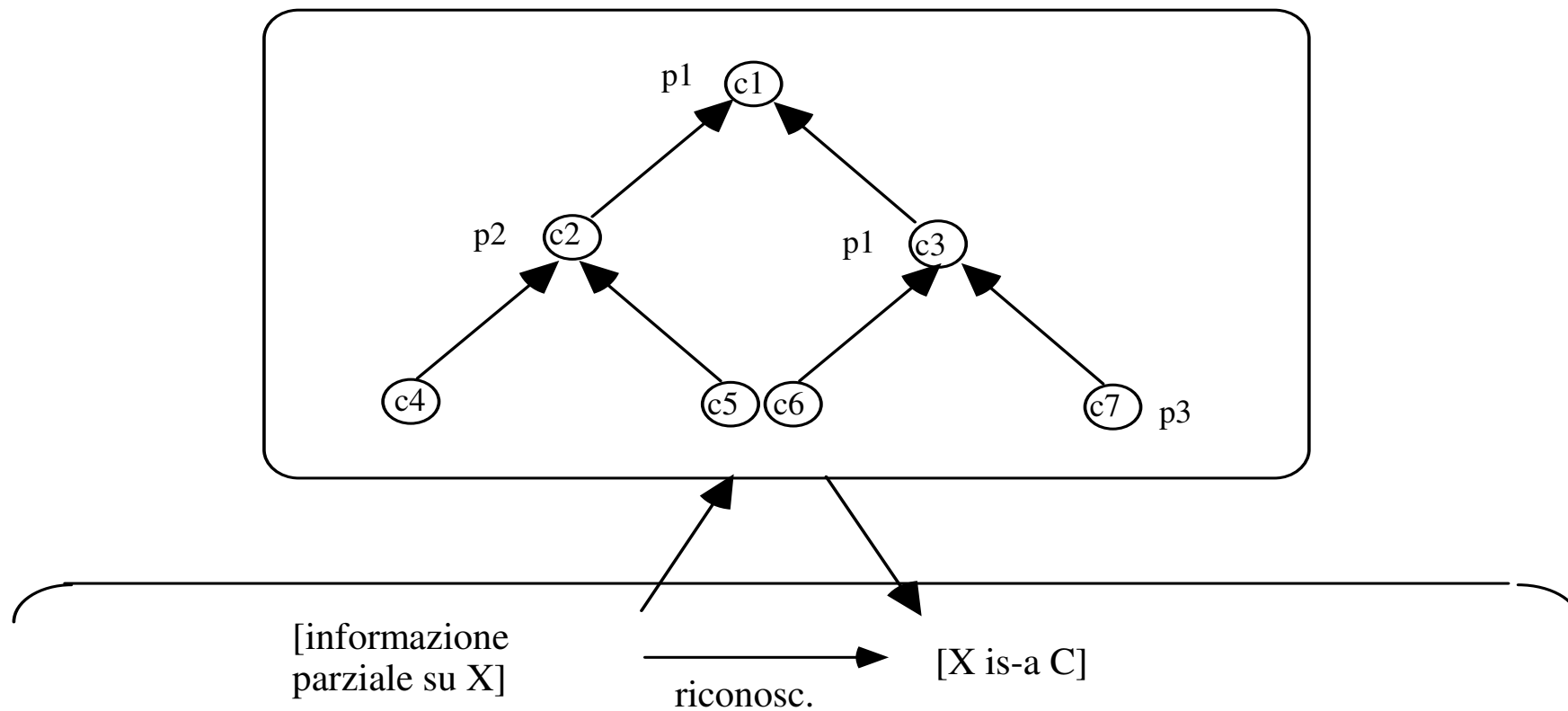


# CLASSIFICAZIONE

---

- Inserisce un concetto nella tassonomia al livello più appropriato (linguaggi terminologici)
- Utilizza il test di sussunzione
- L'algoritmo di classificazione di un nuovo oggetto (concetto) C consiste di due parti:
  - *most specific subsumers*, per individuare i concetti più specifici che sussumono C
  - *most general subsumees*, per individuare i concetti più generali sussunti da C
- Richiede che le classi siano descritte da proprietà necessarie e sufficienti

# RICONOSCIMENTO





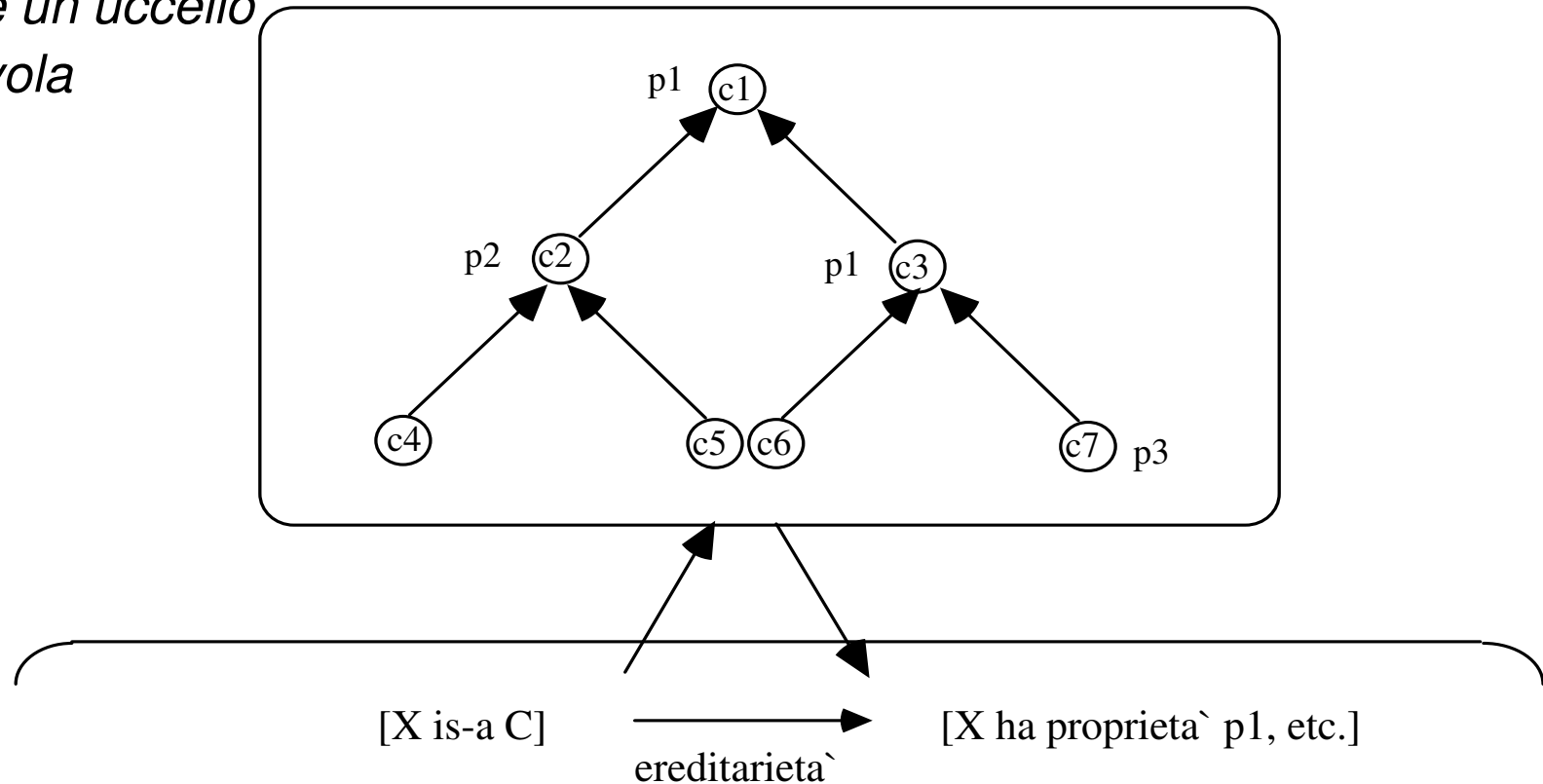
# RICONOSCIMENTO

---

- Data una tassonomia, è l'operazione con cui si "classificano" individui cioè istanze (linguaggi terminologici)
- Data una descrizione che consiste di un insieme di proprietà, trovare un concetto che *meglio* corrisponde alla descrizione data
- Inserimento di un individuo sotto i suoi *most specific subsumers* (prima parte dell'algoritmo di classificazione)
- Inferenza corretta nel caso di descrizioni complete (*necessarie e sufficienti*)

# EREDITARIETA'

- È una forma di inferenza che consente di determinare le proprietà di un oggetto basandosi sulle proprietà dei suoi *antenati*
- *Gli uccelli volano*
- *Titti è un uccello*
- *Titti vola*



# EREDITARIETA'

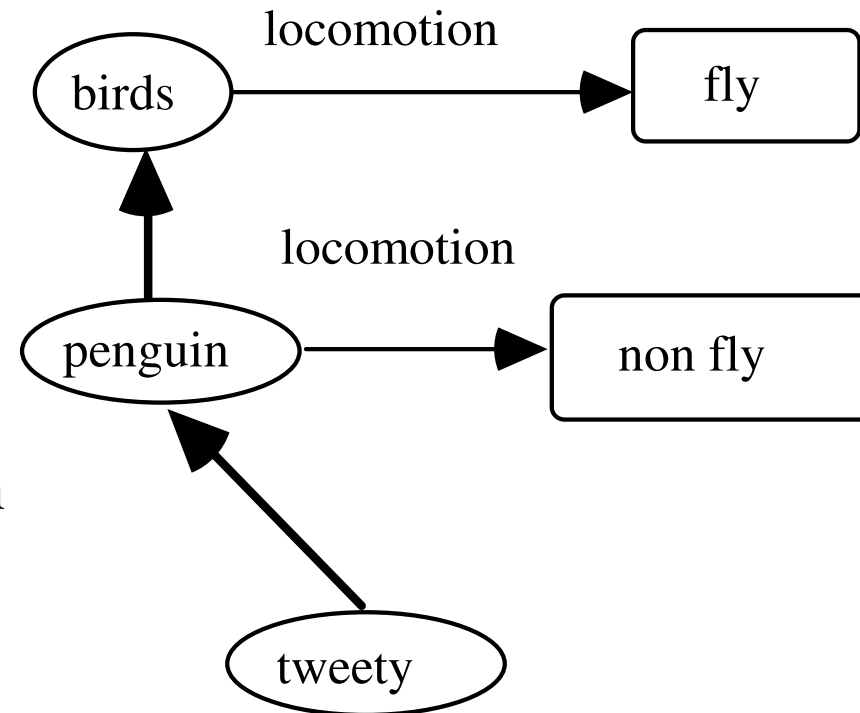
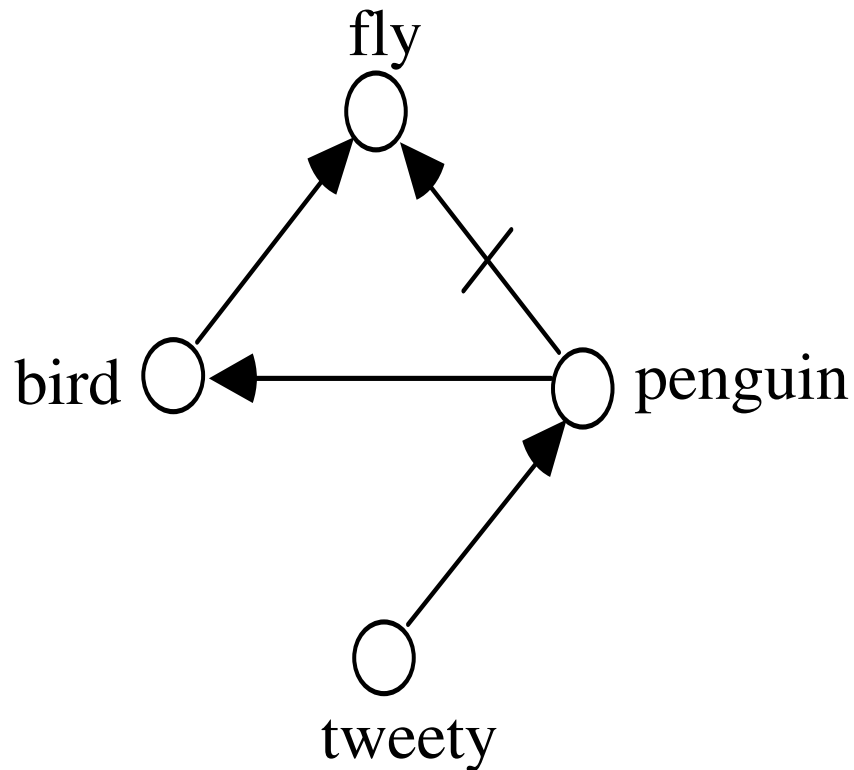
---

Molto spesso ereditarietà, classificazione, riconoscimento sono combinati tra loro

- Problema: possibile presenza di proprietà in conflitto (eccezioni, ereditarietà multipla)
- Molti sistemi sono *non-monotoni*
- *Non-monotonicità:*
  - Un sistema di ereditarietà è *non monotono* se ammette archi “annullabili” (defeasible)

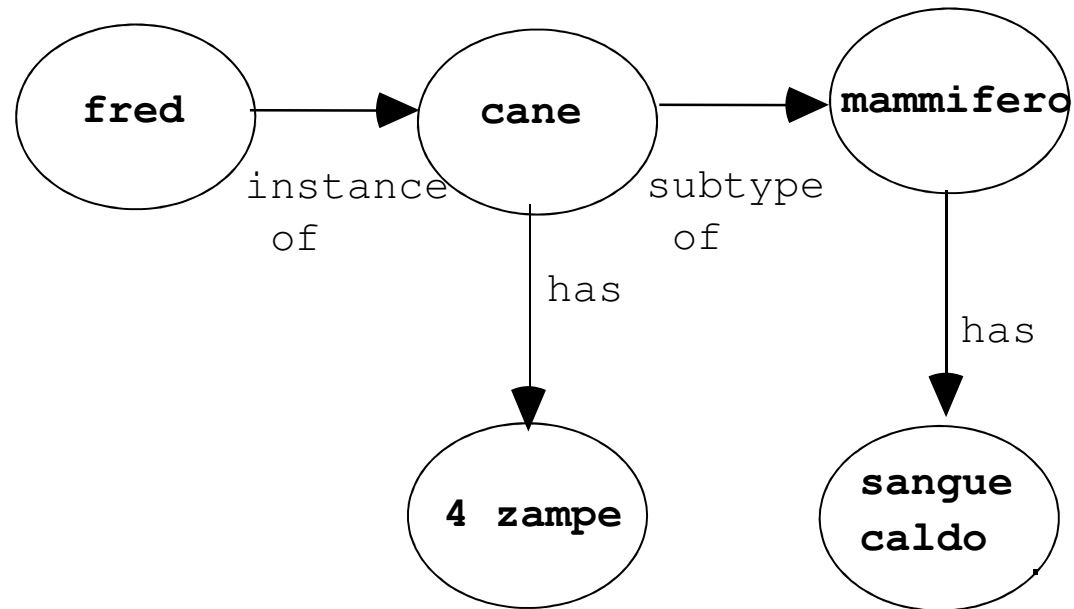
# EREDITARIETA'

- Attraverso archi *is-not-a* (reti di ereditarietà) oppure *overriding* di proprietà (frame, oggetti)



# GERARCHIA ED EREDITARIETÀ DI PROPRIETÀ

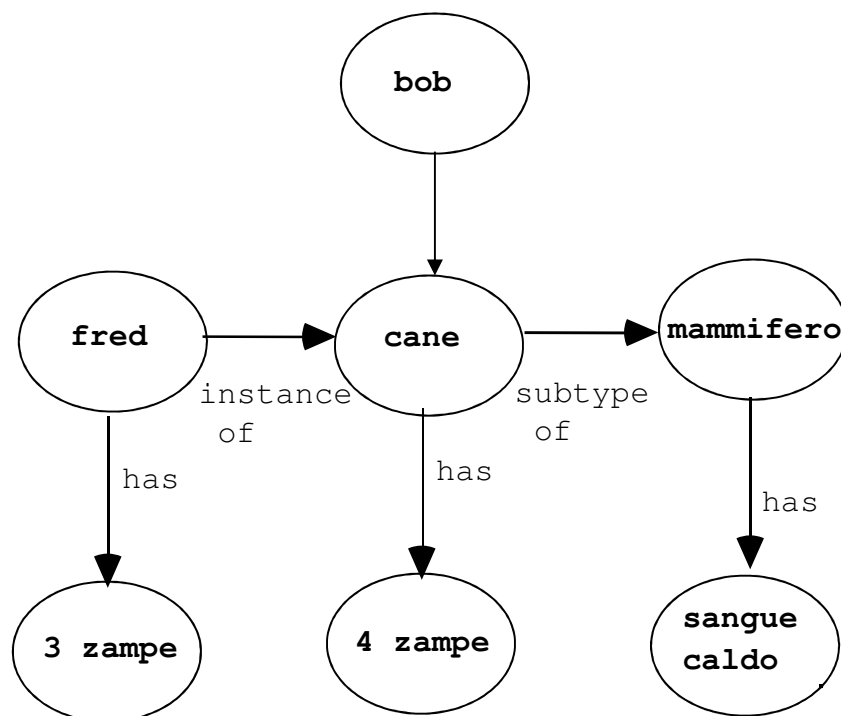
---



- fred è un cane allora:
- ha 4 zampe e ha il sangue caldo.

# DEFAULT

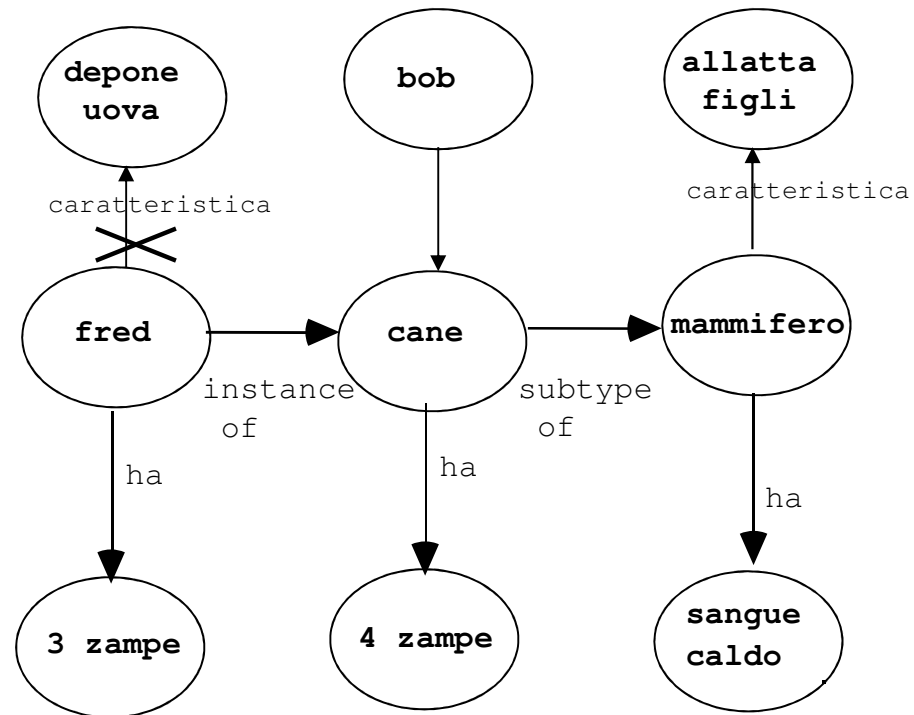
---



- mediante ricopertura di proprietà ai livelli più bassi della gerarchia
- fred ha tre gambe (eccezione).
- **ATTENZIONE!!!** ci sono proprietà che non possono essere ricoperte: (alcune sono quantificazioni universali)

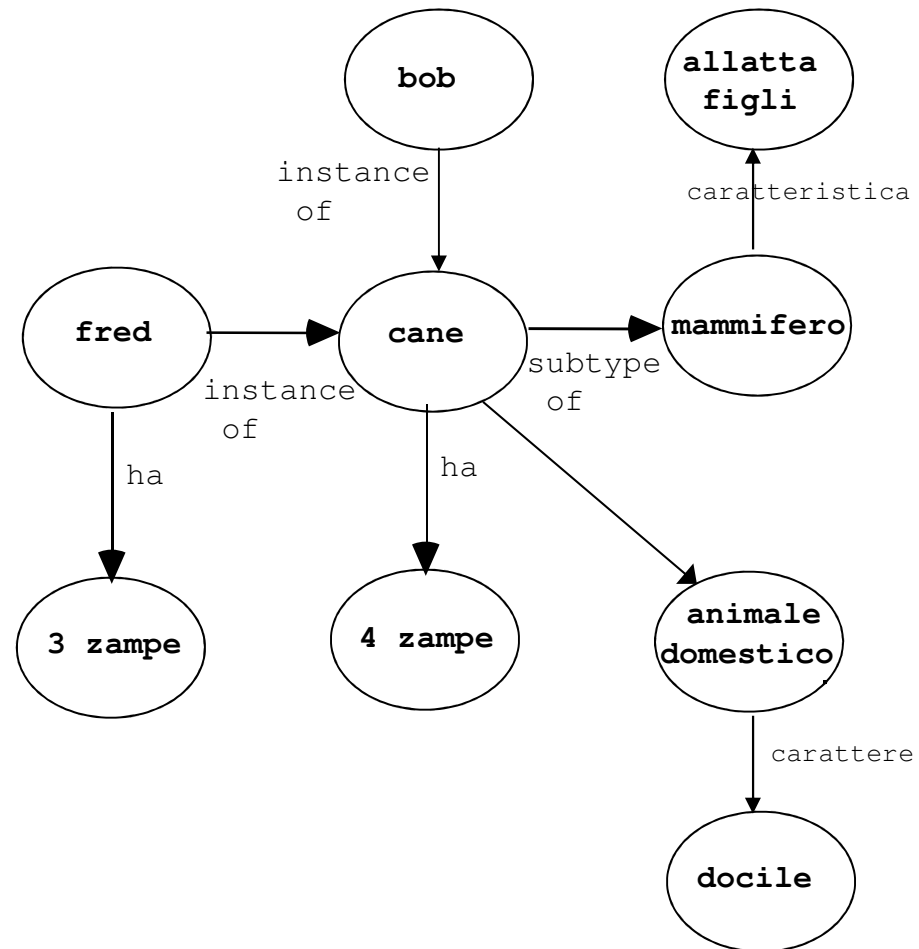
# RICOPRIRE VALORI DI PROPRIETA'

---



# EREDITARIETÀ MULTIPLA:

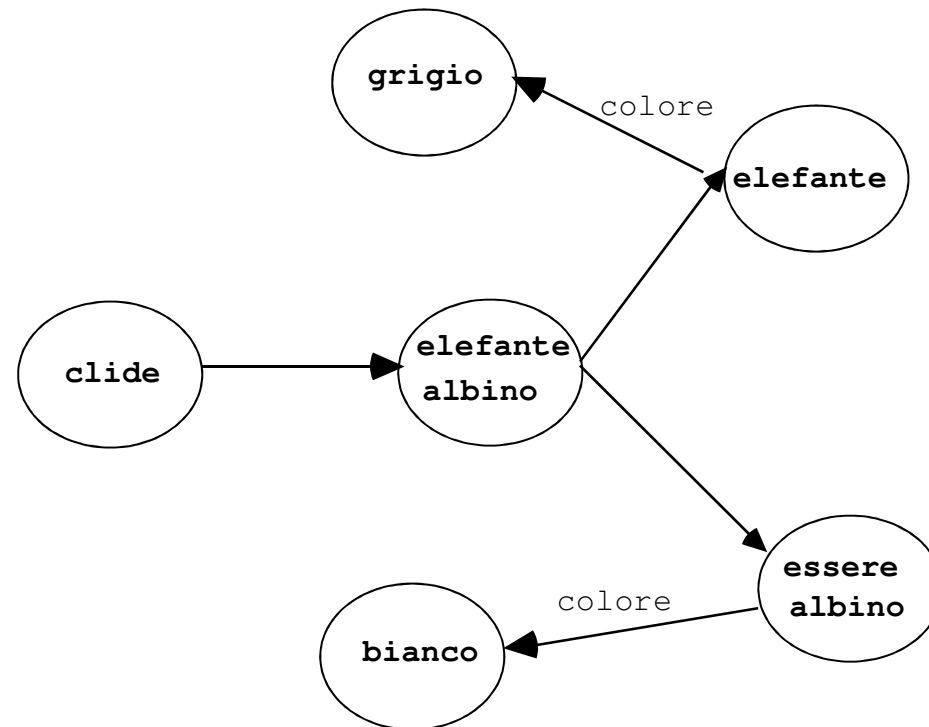
---





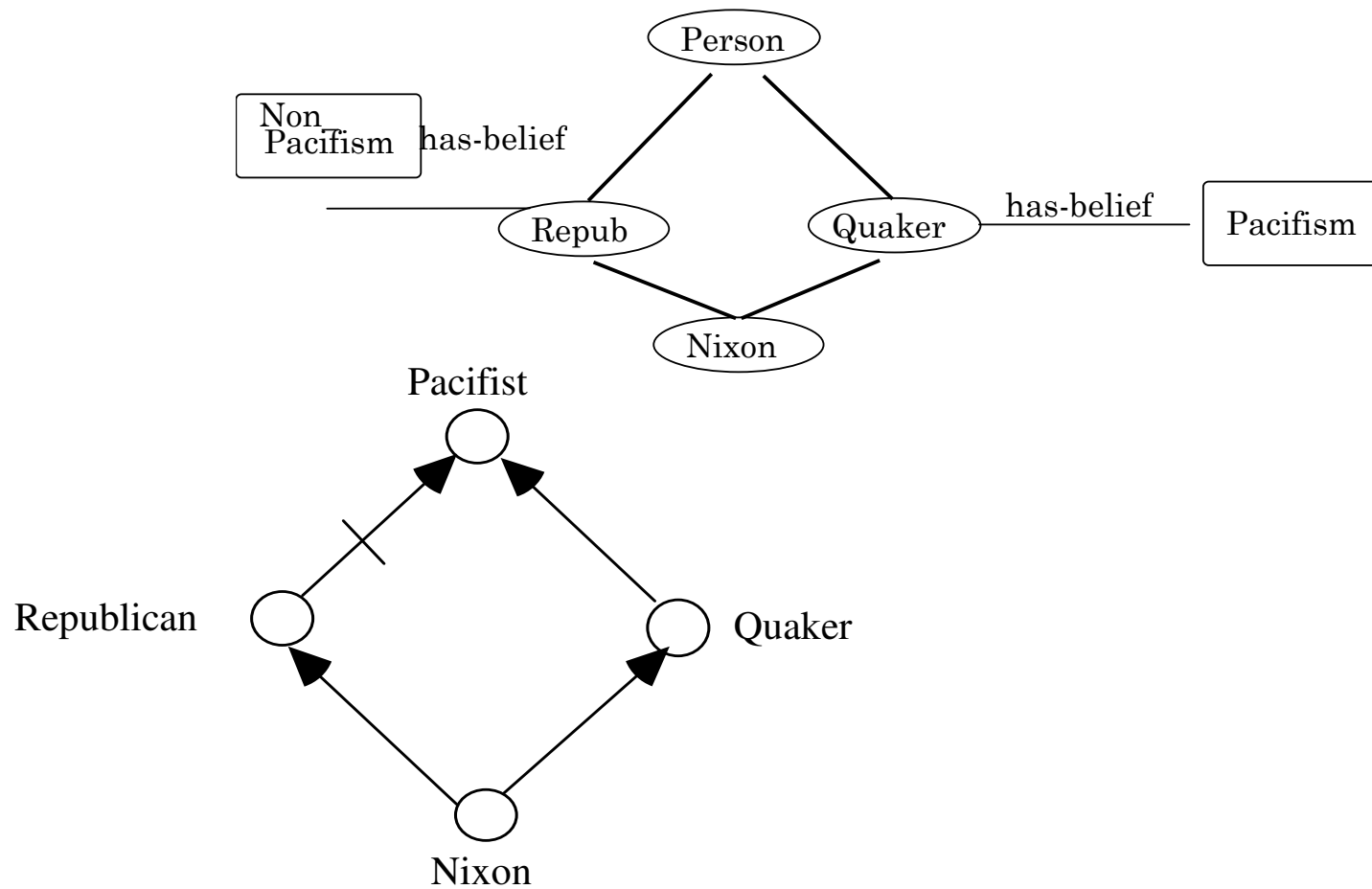
# PROBLEMI: EREDITARIETÀ DI PROPRIETÀ IN CONTRADDIZIONE

---



- Si deve indicare un VERSO DI PERCORRENZA
- Normalmente depth-first con backtracking da sinistra verso destra.

# EREDITARIETA' MULTIPLA



# EREDITARIETA'

---

Due approcci:

- nell'ambito della rappresentazione della conoscenza *reti di ereditarietà*
  - derivano dalle reti semantiche
  - si ereditano proprietà
  - esistono archi defeasibile (“is-not-a”)
  - ereditarietà multipla
- nell'ambito dei linguaggi di programmazione, *linguaggi a oggetti (Java, C++, C#)*
  - si ereditano attributi (variabili) e metodi (operazioni)
  - sia le variabili sia le operazioni ereditabili possono essere ridefinite (overriding)
  - in alcuni (ad es. C++) c'è ereditarietà multipla, in altri si fa una scelta più tranquilla (Java, C#)

# PATH-BASED REASONING

---

- Sistemi di ragionamento basato su cammini
- Ampiamente accettato perché intuitivo e di facile realizzazione
- *Eccezioni* (vedi esempio di “fred è un cane a tre zampe”)
- I concetti che hanno valori di proprietà in conflitto possono essere ordinati totalmente all'interno della gerarchia is-a
- Il conflitto può essere risolto sulla base della *locazione* identificando il concetto “più vicino” (il *più specifico*) e utilizzando l'informazione in esso (*overriding*)

# AMBIGUITA'

---

- Il conflitto *non* può essere risolto sulla base della locazione dei concetti nella gerarchia (si veda l'esempio di Nixon)

Possibili soluzioni:

- *Approccio credulo*
  - Enumerazione delle risposte possibili
  - Stessi risultati ottenuti con la traslazione nella logica default
  - *Nixon è Pacifista*
  - *Nixon non è Pacifista*
- *Approccio scettico*
  - Non trae alcuna conclusione
  - *Non posso concludere nulla su Nixon*
- **NOTA: l'ereditarietà scettica ideale dovrebbe trarre quelle conclusioni che sono vere in ogni estensione credula**
- *Priorità*
  - Aumento della potenza espressiva del linguaggio
  - È più probabile che un Repubblicano sia Pacifista piuttosto che un Quacchero sia non Pacifista

# EREDITARIETA' PROCEDURALE

---

- Usata nei sistemi a frame e ad oggetti
- È definita solo in termini di un algoritmo che opera sulla struttura dati rappresentante il grafo di ereditarietà
- Algoritmo:  
Dato un oggetto A e una proprietà P, trovarne il valore V:
  1. Se c'è un valore per la proprietà P in A restituisci quel valore.
  2. Altrimenti, sia Class l'insieme di tutte le classi che sono puntate da archi che escono da A;
  3. Se Class è vuoto restituisci fallimento;
  4. Altrimenti seleziona una classe C in Class. Se in C c'è un valore per la proprietà P restituisci quel valore. Altrimenti aggiungi a Class tutte le classi puntate da archi che escono da C.
  5. Ritorna al passo 3.
- Nel caso di ereditarietà semplice si percorre un unico cammino nella gerarchia
- Nel caso di ereditarietà multipla, la struttura dati (ad esempio coda o pila) utilizzata per memorizzare l'insieme Class corrisponde alle due strategie di ricerca breadth-first e depth-first

## Abbiamo visto che:

---

- Molti dei ragionamenti che si fanno sono sulle categorie piuttosto che sugli individui
- Se organizziamo la conoscenza in categorie (e sottocategorie) è sufficiente classificare un oggetto, tramite le proprietà percepite, per inferire le proprietà della categoria a cui appartiene (ereditarietà)

## In conclusione...

---

- Le relazioni di sottoclasse organizzano la conoscenza in tassonomie (come in botanica, biologia, nelle scienze librarie ...)
- Ontologie di dominio = tassonomie specializzate
- Molte delle idee delle reti semantiche e dei frame sono state raccolte in logiche specializzate (terminologiche o descrittive)
- Queste logiche sono alla base delle proposte per il *Web semantico*