Tecniche di rappresentazione della conoscenza

Borgo, Guarino
Laboratory for Applied Ontology, ISTC-CNR, www.loa-cnr.it
Corso AI, Univ. di Trento 2007

Argomenti trattati

- ▷ Logiche per possibilità, attitudini, obblighi...
- ▷ Logiche per azioni
- ▷ Rappresentare
- ▷ Strutturare una base di conoscenza
- ▷ Qualità e proprietà di una base di conoscenza

Qualche estensione della logica

Abbiamo visto come formalizzare una proposizione in termini di oggetti, relazioni e funzioni e come legare proposizioni tra loro.

Talvolta non è sufficiente stabilire la verità o falsità di una proposizione. Vogliamo dire che una situazione è possibile, che una proposizione è saputa, che una proposizione è ritenuta vera (senza sapere se lo è o no), che una certa cosa deve essere fatta...

- ▷ "Giorgio potrebbe essere ricco"
- ▷ "Giorgio sa di avere 80 anni"
- ▷ "Giorgio crede di avere 80 anni"
- ▷ "Giorgio deve fare i compiti"

Possibilità e tempo in logica

Per catturare questi casi si usano operatori detti 'modali'. Questi sono operatori che si applicano ad enunciati (formule chiuse):

1. $\Diamond Ricco(Giorgio)$ = 'È possibile che Giorgio sia ricco'

2. lo stesso simbolo si usa spesso anche per la logica temporale (noi qui aggiungiamo un indice t)
   $\Diamond_t Ricco(Giorgio)$ = 'Giorgio diventerà ricco'
   lett.: esiste un momento futuro in cui Giorgio sarà ricco

2-*bis* In altri casi, il tempo è una categoria del dominio e quindi non servono nuovi operatori
$\exists t\,(Ricco(Giorgio, t) \wedge t > ora)$ = 'Giorgio diventerà ricco'

e ancora: credenza, conoscenza, obbligo

3. $Bel_{Giorgio}(Eta'(Giorgio, 80)) =$
   'Giorgio crede di avere 80 anni'
4. $K_{Giorgio}(Eta'(Giorgio, 80)) =$ 'Giorgio sa di avere 80 anni'
5. $\bigcirc Finito(Giorgio, compiti) =$
   'È obbligatorio che Giorgio finisca i compiti'

...e nozioni derivate - 2

▷ Definiamo la necessità come “non è possibile che non”

$\neg\Diamond\neg Ricco(Giorgio)$ = ‘È necessario che Giorgio sia ricco’

▷ Necessità temporale: “non sarà mai il caso che non”

$\neg\Diamond_t\neg Ricco(Giorgio)$ = ‘Giorgio sarà sempre ricco’

▷ Libertà di fare/decidere/agire: “non è obbligatorio che non”

$\neg \bigcirc \neg Finito(Giorgio, compiti) =$
‘Giorgio può (gli è permesso di) finire i compiti’

NB: di solito si usa il simbolo $\Box$ per la sequenza $\neg\Diamond\neg$.

Abbiamo usato vari significati del termine ‘possibilità’ !
Attenzione a non confonderli.

Formule modali classiche

$C)$ $(\Box A \wedge \Box B) \rightarrow \Box(A \wedge B)$

$M)$ $\Box(A \wedge B) \rightarrow (\Box A \wedge \Box B)$

$K)$ $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$

$D)$ $\Box A \rightarrow \Diamond A$

$T)$ $\Box A \rightarrow A$

$B)$ $A \rightarrow \Box\Diamond A$

$4)$ $\Box A \rightarrow \Box\Box A$

$5)$ $\Diamond A \rightarrow \Box\Diamond A$

Abbiamo visto tre nozioni di modalità.
Sai dire che cosa significano queste formule nei vari casi?

Azioni

▷ Nella logica dei predicati le azioni sono relazioni o oggetti particolari:

1. $Uccidere(Bruto, Cesare)$
   gli argomenti sono agente e paziente (per azioni transitive). Possiamo aggiungere il tempo (nel 44 A.C.), il mezzo (con un pugnale), il modo (con forza) etc.
2. $Atto(uccidere, Bruto, Cesare)$
   arg.: azione, agente, paziente (ed estensioni come sopra).

▷ Nella logica modale le azioni sono operatori modali (*logica dinamica*)
se scriviamo α_C per l'azione di uccidere Cesare
$[\alpha_C]_{Bruto}\, Morto(Cesare)$
se scriviamo β per l'azione di bere
$[\beta]_{Giorgio}\, Dissetato(Giorgio)$

Cosa e come rappresentare

La logica ci ha fornito un linguaggio espressivo e non ambiguo.

Ora dobbiamo imparare ad usarlo.

- ▷ Come si deve rappresentare la conoscenza?
- ▷ Quale vocabolario adottare?
- ▷ Che cosa dire e come dirlo?

L'**ingegneria della conoscenza** studia la metodologia e il processo di costruzione di una base di conoscenza. Si tratta di analizzare il dominio di interesse, selezionare i concetti importanti e creare una rappresentazione formale di oggetti e relazioni di quel dominio.

Una prima fase si concentra sull' **acquisizione della conoscenza**.

Desiderata

Lo scopo è ottenere una base di conoscenza che sia:
- **espressiva** (possiamo fare tutte le distinzioni di interesse),
- **concisa** (minimo uso delle perifrasi),
- **non ambigua** (quello che è espresso ha un significato preciso),
- **slegata dal contesto** (le espressioni non cambiano con il contesto),
- **efficace** (capace di fornire risposte velocemente),
- **chiara** (comprensibile all'utente),
- **corretta** (senza contraddizioni).

Lasciamo da parte il problema dell'**efficienza** che coinvolge nozioni non coperte in questo corso. Ci concentriamo sul *contenuto*.

Distinzione cruciale:
il contenuto e l'inferenza sono aspetti distinti in una KB (però entrambi dipendono dal linguaggio). Vanno considerati in modo separato.

Un esempio

Supponiamo che io dia agli studenti nelle prime due file la seguente (molto semplice) base di conoscenza:

$$\forall b\, OrsoDalCervelloMoltoPiccolo(b) \rightarrow Stupidino(b)$$

e che io dia agli altri studenti la sequente base di conoscenza:

$$\forall b\, P(b) \rightarrow Q(b)$$

Ho dato basi di conoscenza diverse ai due gruppi?
In altre parole, c'è differenza tra queste due espressioni formali?

Prendere delle decisioni

1. decidere di cosa parlare
   (capire il dominio e selezionare ciò di cui parlare)
2. decidere il vocabolario dei predicati, funzioni, costanti (stabilire cosa va inserito come relazione, che nome dargli,...)
3. codificare la conoscenza generale sul dominio
   (vincolare il vocabolario)

→ R&N usa il termine 'ontologia' in modo diverso da noi!

4. inserire le istanze specifiche
5. interrogare la base di conoscenza ottenuta

Il brodo della conoscenza

Una seria analisi della conoscenza è cruciale per i primi tre punti elencati precedentemente.

Immaginiamo di fissare un dominio in cui ci sentiamo a nostro agio (l'università, la famiglia, uno sport...)

La nostra conoscenza di questo dominio può essere verbalizzata in proposizioni, descritta in immagini o attraverso schemi, condivisa usando analogie. Talvolta non può essere esplicitata (alcune cose le sappiamo come intuizioni o sensazioni). Tutta questa conoscenza è complessa, disorganizzata e spesso inconsistente. Per questo si parla di "brodo della conoscenza" (in analogia con il "brodo primordiale").

Selezionare e strutturare

Selezionare non basta.
È estremamente importante decidere la struttura o organizzazione della base di conoscenza per garantire la qualità della formalizzazione, la sua coerenza, la sua generalità.

La formalizzazione logica si concentra su tre tipologie lessicali: relazioni, funzioni e costanti. Quindi, dobbiamo costruire una classificazione basandoci su queste distinzioni. Ma non basta elencare relazioni, funzioni e costanti del dominio.

- ▷ Sono tutte le relazioni (funzioni, costanti) egualmente importanti?
- ▷ Sono tutte *strutturali*?

Selezionare e strutturare - 2

Come individuare gli elementi centrali (o strutturali, appunto) che devono essere usati per organizzare in modo omogeneo e coerente il resto della conoscenza?

Vediamo due esempi:

- ▷ Giorgio è il più giovane dei Dalla Vecchia
- ▷ Giorgio è uno dei Dalla Vecchia

- ▷ Questo olio è denso
- ▷ Questo olio è vegetale

Principali relazioni strutturanti

- Essere-membro-di
  "La mia auto è registrata presso il P.R.A."
- Essere-parte-di "La mia auto ha un motore"
- Essere-istanza-di "La mia auto è un mezzo di locomozione"
- Essere-una-sottoclasse-di
  "Le auto sono mezzi di locomozione"

La relazione 'essere-parte-di'

Espressioni linguistiche che usano, in modo implicito o esplicito, la relazione di parte:

- il motore è parte dell'auto
- la porta ha una maniglia
- la maniglia della porta
- questo sciroppo è fatto di acqua e zucchero
- un membro della commissione
- il davanti dell'auto

La relazione ‘essere-parte-di’ - 2

- una pecora del gregge
- dammi dell’acqua
- ci sono 10gr di zucchero in questo sciroppo
- un litro d’acqua
- sedie, due sedie, una delle sedie
- Nicola e Stefano

La relazione ‘essere-parte-di’: specializzazioni

Diverse relazioni di parte che sono state caratterizzate formalmente:

- ▷ membro-collezione: *questa pecora / le pecore*
  non transitiva
- ▷ sottocollezione-collezione: *Italia / Unione Europea*
  transitiva
- ▷ parte-intero: *un litro d’acqua / l’acqua; un pezzo di torta / la torta*
  transitiva
- ▷ sostanza-intero: *10ml di latte / capuccino*
  transitiva
- ▷ pezzo-intero: *la parte sinistra della lavagna / la lavagna*
  transitiva
- ▷ componente-intero: *la maniglia / la porta; il motore / la mia auto*
  si divide in relazioni più specifiche non tutte transitive

Transitività nelle relazioni strutturanti

- Essere-membro-di

  non è transitiva

- Essere-parte-di

  (dipende da quale specifica relazione di parte si usa, vedi slides precedenti)

- Essere-istanza-di

  non è transitiva

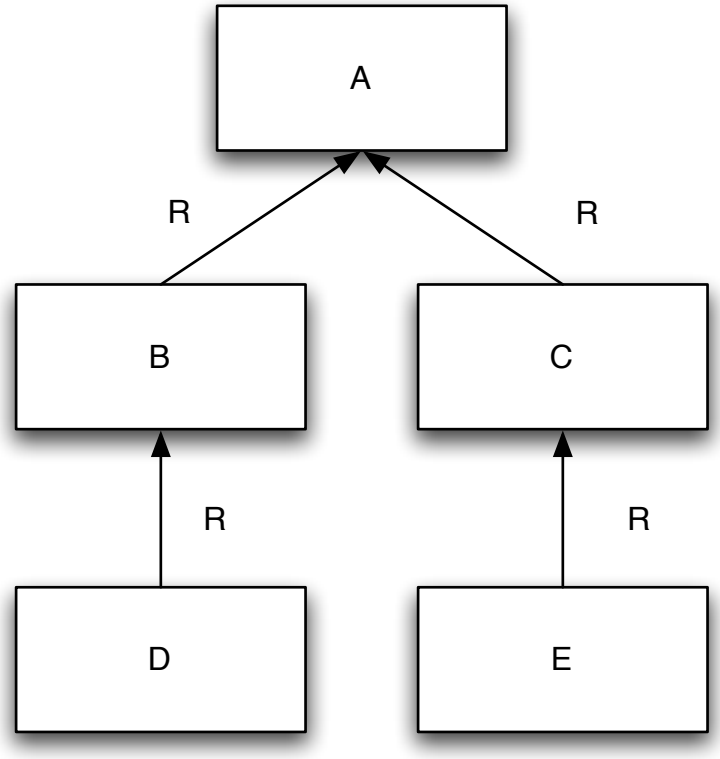
- Essere-una-sottoclasse-di

  transitiva

Grafici

Un grafo può essere usato per rappresentare relazioni strutturanti.
Ma bisogna stare attenti...

Che cosa indica un rettangolo?
- una entità,
- una proprietà,
- una classe di entità,
- ...



Grafici in formule

Il grafico della pagina precedente può avere vari sgnificati.
Per esempio, corrisponde formalmente alle seguenti formule:

1. $R(D, B) \wedge R(B, A) \wedge R(E, C) \wedge R(C, A)$

2. $\forall x(D(x) \rightarrow B(x)) \wedge \forall x(B(x) \rightarrow A(x)) \wedge$
   $\forall x(E(x) \rightarrow C(x)) \wedge \forall x(C(x) \rightarrow A(x))$

3. $\forall x(D(x) \rightarrow \exists y(B(y) \wedge R(x, y))) \wedge$
   $\forall x(B(x) \rightarrow \exists y(A(y) \wedge R(x, y))) \wedge$
   $\forall x(E(x) \rightarrow \exists y(C(y) \wedge R(x, y))) \wedge$
   $\forall x(C(x) \rightarrow \exists y(A(y) \wedge R(x, y)))$

Grafici in formule - 1

$$R(D, B) \wedge R(B, A) \wedge R(E, C) \wedge R(C, A)$$

▷ Esempio: R è **parte-di**

- il mio dito (D) è parte della mia mano (B) e la mia mano è parte del mio corpo (A)
- Il rosso scarlatto (D) è una parte dell'area del rosso (B) e il rosso è una parte dello spazio dei colori (A)

▷ Esempio: R è **membro-di**
Giorgio (D) è un membro di ABC Nuoto (B) e ABC Nuoto è un membro di Federazione Nazionale Nuoto (A)

Fin qui i rettangoli indicano tutti *singole entità* o *singole proprietà*, o *concetti*. Vediamo ora un caso diverso.

Grafici in formule - 1 bis

$$R(D, B) \wedge R(B, A) \wedge R(E, C) \wedge R(C, A)$$

- ▷ Esempio: R è **istanza-di**
  - l'arrotino Giorgio (D) è un'istanza di Produttore (B) e Produttore è un'istanza di Concetti per l'economia (A)
  - Questa sedia (E) è un'istanza di Manufatto (C) e Manufatto è un'istanza di Concetti per l'economia (A)

In questo caso il rettangolo più in basso indica una *singola entità*, il secondo un *concetto* e quello in alto una *classe di concetti*.

Grafici in formule - 2

In altri casi i rettangoli indicano *classi di entità (o di proprietà)*

In formule: $\forall x(D(x) \rightarrow B(x)), \quad \forall x(B(x) \rightarrow A(x))$
$\forall x(E(x) \rightarrow C(x)), \quad \forall x(C(x) \rightarrow A(x))$

▷ Esempio: R è **sottoclasse-di**
[nota che R viene esplicitato nella formula dall'implicazione]

i gatti (D) sono felini (B) e i felini sono animali (A)

Grafici in formule - 2 bis

Altro caso con i rettangoli che indicano *classi* di entità o di proprietà.

In formule:

$$\begin{aligned}
&\forall x(D(x) \rightarrow \exists y(B(y) \wedge R(x,y))) \wedge \\
&\forall x(B(x) \rightarrow \exists y(A(y) \wedge R(x,y))) \wedge \\
&\forall x(E(x) \rightarrow \exists y(C(y) \wedge R(x,y))) \wedge \\
&\forall x(C(x) \rightarrow \exists y(A(y) \wedge R(x,y)))
\end{aligned}$$

- ▷ Esempio: R è **parte-di**
  un atomo di H_2O (D) è parte di una quantità d'acqua (B) e ogni quantità d'acqua è parte di una quantità di materia (A)
- ▷ Esempio: R è **membro-di**
  ogni giocatore (D) è membro di una squadra (B) e ogni squadra è membro di una federazione (A)
  [Nota la differenza con l'esempio nella slide 23]

Caratteristiche e proprietà

Un'altra relazione importante è: Avere-attributo.
Il linguaggio naturale ci dà modi diversi di esprimere caratteristiche delle entità e proprietà di entità.

- ▷ "La mia auto è rossa"
- ▷ "La mia auto ha colore rosso"
- ▷ "Il colore della mia auto"
- ▷ "Il colore che ha la mia auto"

Partizionare

Partizionare è una forma particolare di strutturare. Si ottiene attraverso relazioni unarie (predicati) che permettono di dividere gli elementi del dominio in tipi o classi omogenee.

$Umano(x)$ vale se e solo se x ha la proprietà di essere umano e quindi il predicato $Umano$ individua le entità che sono uomini e le separa da quelle che non lo sono.

Non tutte le partizioni sono egualmente utili, interessanti e stabili

"Essere materiale", "Essere blu", "Essere bagnato", "Essere buono"

La scelta delle partizioni da usare dipende in parte dal dominio. Alcune sono generali e sono considerate fondamentali per una buona base di conoscenza, ad es. la partizione data dai predicati di Oggetto e di Evento. Altre sono legate al dominio, ad es. quelle date da Cliente e da Fornitore sono spesso usate nelle KB sul mercato.

Classi di entità - 1

Abbiamo detto che un predicato individua una classe di entità. Confrontiamo due classi di entità: persona e lavoratore

- ▷ Ogni lavoratore è una persona ma non ogni persona è un lavoratore
  → *persona è più generale di lavoratore*
- ▷ Data una entità, se è una persona ad un qualche momento allora lo è durante tutta la sua esistenza
  → *persona è una classificazione stabile*
- ▷ La stessa entità è un lavoratore in certi periodi e non lo è in altri
  → *lavoratore non è una classificazione stabile*

Classi di entità - 2

Abbiamo così scoperto la differenza fra classi motivate da proprietà ontologiche e classi motivate da proprietà contestuali.

Le classi del primo tipo sono buone candidate per strutturare l'ontologia.

Ci sono classi che non sono motivate ontologicamente ma sono spesso utili in contesti specifici (come Lavoratore, Cliente, Risorsa etc.). Queste classi individuano **ruoli** e devono essere usate con cautela.

Le classi individuate come strutturanti per la KB si dicono *categorie*.

Principali categorie

- ▷ Entità che esistono nel tempo — "Questa sedia"
- ▷ Entità che si sviluppano nel tempo — "La vita di questa sedia"
- ▷ Entità che esprimono qualità — "il peso di questa sedia"
- ▷ Entità astratte — "Il numero 0"

Questa lista corrisponde ad una scelta ontologica ma non è l'unica:
- ad es. si possono escludere alcune entità (quelle che esistono nel tempo, quelle astratte...)
- si possono ridurre le entità: 'questa sedia' non esiste propriamente, ci sono solo degli atomi ragruppati in una certa forma...

Reificare

Reificare significa nominalizzare e può riferirsi a relazioni (con uno o più argomenti) o a classi. Reificare ci permette di parlare di proprietà di queste relazioni o classi.

Si usa molto reificare nel linguaggio naturale. Considera

- "i disoccupati sono diminuiti"
  (lett.: la classe dei disoccupati è diminuita di numero)
  Quello che vogliamo dire non è che ogni disoccupato è diminuito ma che la classe ha un numero minore di elementi rispetto a prima.
- "l'affitto incide sulla vita delle famiglie"
- "il cabernet 2004 è migliore del cabernet 2003"
  Non significa che ogni bottiglia di cabernet 2004 contiene un vino migliore di qualsiasi bottiglia di cabernet del 2003 ma che *complessivamente* il vino del 2004 è migliore di quello del 2003.

Qualità di una base di conoscenza

Abbiamo detto che una base di conoscenza dovrebbe essere

- ▷ **espressiva** $\mapsto$ scelta di un linguaggio espressivo
- ▷ **concisa** $\mapsto$ scelta di un linguaggio espressivo e ricco di vocaboli, attenzione alle formule
- ▷ **non ambigua** $\mapsto$ scelta di un linguaggio formale
- ▷ **slegata dal contesto** $\mapsto$ scelta di un linguaggio formale
- ▷ **efficace** $\mapsto$ scelta di un linguaggio con buone proprietà computazionali
- ▷ **chiara** $\mapsto$ scelta di un linguaggio vicino al linguaggio naturale
- ▷ **corretta** $\mapsto$ corrisponde al dominio e non contiene formule contraddittorie

Cosa vogliamo da una base di conoscenza

- ▷ *trasparenza*
  la conoscenza deve essere catturata in modo completo ed esplicito, l'utente non deve ricorrere a conoscenza esterna per capire né essere costretto a reinterpretare quanto formalizzato nella base di conoscenza.

- ▷ *riusabilità*
  la conoscenza deve essere organizzata in modo da applicarsi a ogni caso/problema in quel dominio

- ▷ *semplicità nell'elicitazione della conoscenza*
  la conoscenza deve essere organizzata in modo da facilitare l'acquisizione di nuova conoscenza

Mondo aperto e Mondo chiuso

Attenzione
Nella semantica delle basi di conoscenza, tradizionalmente si distinguono due tipi di modelli

- ▷ mondo chiuso: rappresenta la conoscenza totale su ogni cosa di un sistema.
  Ogni enunciato nel sistema è vero oppure è falso (non ci sono casi indefiniti o inconsistenti) e ogni cosa può essere conosciuta facendo inferenze.

- ▷ mondo aperto: rappresenta conoscenza parziale di un sistema, quindi alcune cose sono conosciute come vere, altre come false e di altre semplicemente non si sa.

Linguaggi ed esempi di basi di conoscenza

- ▷ Prolog
- ▷ KL-ONE
- ▷ UML

Programmazione logica: il Prolog

Il Prolog è un linguaggio di programmazione molto usato per costruire basi di conoscenza (mediche, legali, finanziarie, produttive etc.) che siano semplici, veloci ed efficienti.

Il programma e gli input sono visti come affermazioni logiche sul mondo. L'esecuzione del programma è il modo di esplicitare le conseguenze delle affermazioni date.

Un algoritmo (una procedura passo-passo che può essere eseguita automaticamente) si riduce ad un insieme di formule logiche più delle regole per controllare il processo di inferenza.

Prolog: il linguaggio - 1

- ▷ Un programma è una sequenza di formule *implicitamente* congiunte.
  Le variabili sono *implicitamente* vincolate dal quantificatore $\forall$.
  Le variabili in formule diverse sono considerate distinte
- ▷ Sono ammesse solo formule di Horn: formule atomiche oppure del tipo $A_1 \wedge \ldots \wedge A_n \rightarrow B$ con A_i e B tutte formule atomiche.
- ▷ I termini possono essere simboli di costante, variabili o termini funzionali.
- ▷ Le interrogazioni includono congiunzioni, disgiunzioni, variabili e termini funzionali.

Prolog: il linguaggio - 2

- ▷ C'è un operatore di *negazione come fallimento*: $\neg P$ è dimostrato se il sistema fallisce nel dimostrare P.
- ▷ Si assume che termini sintatticamente distinti indichino oggetti distinti: non si può affermare $A = B$ o $A = f(x)$ poiché A (il termine a sinistra) è un termine chiuso. Invece è possibile affermare $x = B$ e $x = F(y)$.
- ▷ Viene fornita una grossa libreria di predicati e funzioni predefinite.

Prolog: un esempio - 1

Definizione formale della relazione 'membro' (di una lista)

```
membro(X,[X,L]).
membro(X,[Y,L]) :- membro(X,L)
```

nella notazione logica a noi familiare, questo diventa

$\forall x, l \; Membro(x, [x, l])$

$\forall x, y, l \; Membro(x, l) \rightarrow Membro(x, [y, l])$

Nota l'inversione nella seconda formula che si legge:
per verificare `membro(X,[Y,L])` è sufficente verificare `membro(X,L)`.

Prolog: interpretazione di una regola

Prendiamo una regola

$$\mathtt{A} :- \mathtt{B_1}, \mathtt{B_2}, \ldots, \mathtt{B_n}$$

A è detta 'testa' della regola,
$B_1, \ldots, B_n$ formano il 'corpo' (congiunzione dei B_i)

- ▷ *interpretazione dichiarativa*
  perchè A sia vera è sufficiente che siano veri $B_1, B_2, \ldots$ e B_n
- ▷ *interpretaione procedurale*
  A (la 'testa') è una chiamata di procedura e $B_1, \ldots, B_n$ (il 'corpo') fornisce una serie di procedure da eseguire nell'ordine dato

Prolog: un esempio - 2

Il programma Prolog della slide 40 può rispondere alle seguenti domande

- è vero che `membro(2,[1,2,3])`?
  risultato: vero
- per quali valori di x si ha `membro(x,[1,2,3])`
  risultato: (elenco dei valori) 1, 2 e 3
- per quali valori di x si ha `membro(2,[1,x,3])`
  risultato: 2
- per quali valori di x si ha `membro(2,x)`
  risultato: (elenco vuoto)
  Più in generale, questa richiesta fornisce un elenco con le liste contenute nel sistema e che contengono 2

Prolog: un altro esempio - 1

Considera la base di conoscenza (KB)

1. `Genitore(X,Y) :- Padre(X,Y)`
2. `Genitore(X,Y) :- Madre(X,Y)`
3. `Antenato(X,Y) :- Genitore(X,Y)`
4. `Antenato(X,Y) :- Genitore(X,Z), Antenato(Z,Y)`
5. `Padre(Giorgio, Marco)`
6. `Padre(Giorgio, Luca)`
7. `Madre(Lia, Giorgio)`

Prolog: un altro esempio - 2

Vogliamo sapere se `Lia` è un antenato di `Marco`.
Per fare ciò, ipotizziamo il contrario e vediamo se otteniamo una contraddizione dalla KB.

`:- Antenato(Lia,Marco)` (goal negato)

Nota: questa è la notazione Prolog per esprimere la negazione di `Antenato(Lia,Mark)`.

Vediamo alla lavagna come il PROLOG ragiona con la base di conoscenza.

Prolog: inferenze

- ▷ Tutte le inferenze sono fatte con concatenazione all'indietro (se la deduzione scelta arriva ad un punto morto, si ritorna indietro passo-passo per provare delle alternative, se nessuna alternativa dà una risposta positiva, si ha il fallimento della prova)
- ▷ La deduzione utilizza le formule rispettando l'ordine (da sinistra verso destra).
  Le affermazioni nella base di conoscenza sono usate nell'ordine in cui sono elencate.
- ▷ Salta delle verifiche nell'applicare l'unificazione (in principio la deduzione non è corretta ma in pratica questo dà problemi solo raramente).

Limiti del Prolog: incompletezza, negazione

Prendiamo una base di conoscenza simile alla precedente

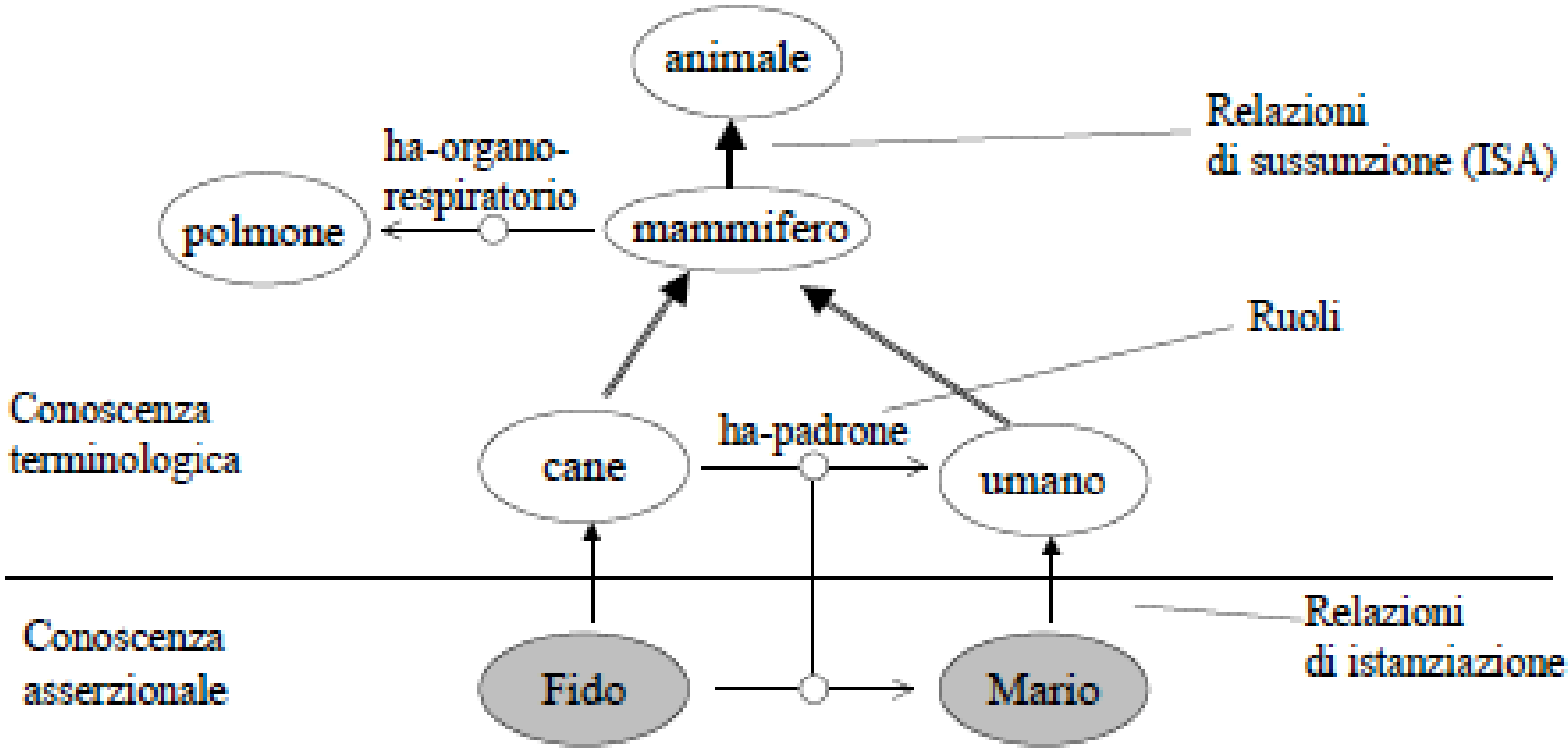
1. `Genitore(X,Y) :- Padre(X,Y)`
2. `Genitore(X,Y) :- Madre(X,Y)`
3. `Antenato(X,Y) :- Genitore(X,Y)`
4. `Antenato(X,Y) :- Antenato(Z,Y), Genitore(X,Z)`
5. `Padre(Giorgio, Marco)`
6. `Padre(Giorgio, Luca)`
7. `Madre(Lia, Giorgio)`

8. consideriamo lo stesso goal negato
   `:- Antenato(Lia,Mark)`
   e vediamo alla lavagna cosa succede ora

Sistemi di logica descrittiva: KL-ONE

- ▷ L'idea delle reti semantiche è di considerare gli oggetti come nodi di un grafo e le relazioni binarie come archi tra in nodi.
- ▷ Il problema delle rete semantiche è che spesso mancano di una semantica chiara.
- ▷ Il KL-ONE (e altri sistemi simili) formalizza la nozione di rete semantica cioè il significato di nodi e archi.

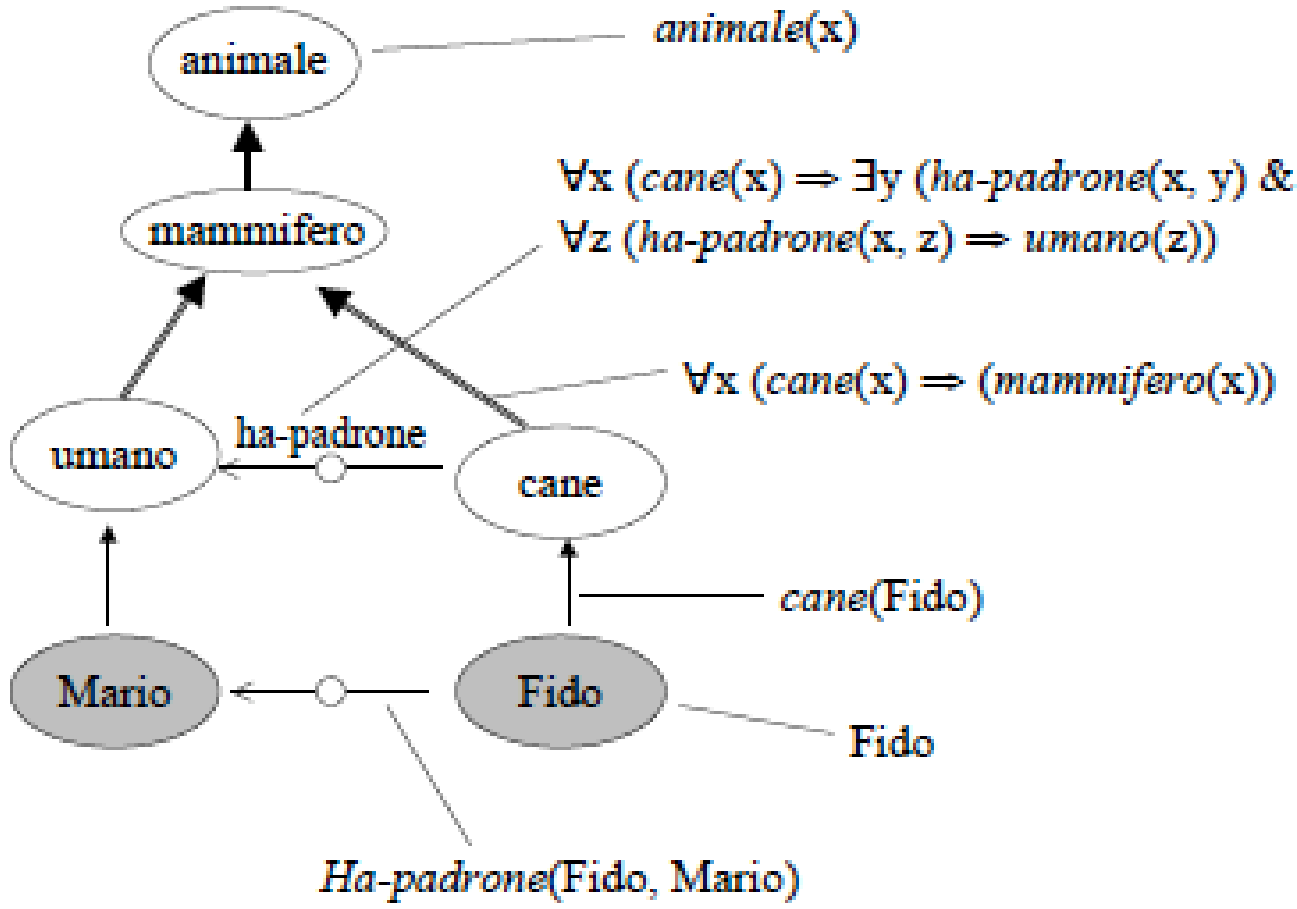
KL-ONE: uno schema



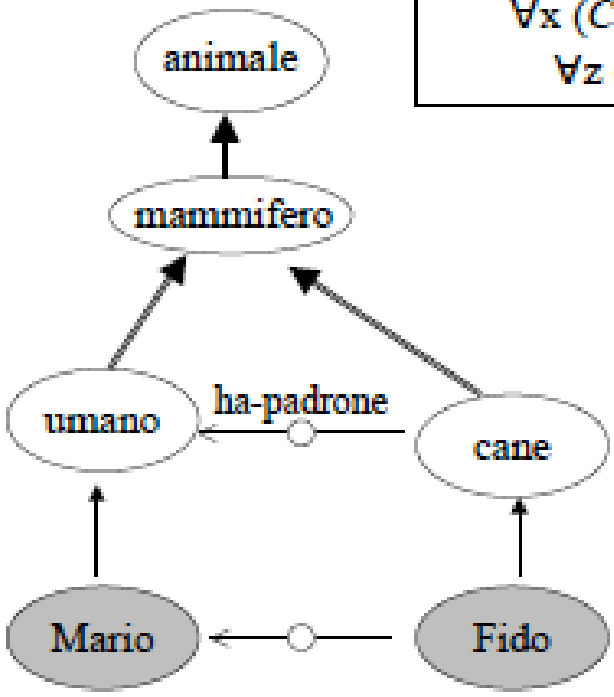
KL-ONE: primitive

- ▷ Concetti generici (concetti come *cane*)
- ▷ Concetti individuali (istanze come *Fido*)
- ▷ Relazione di sussunzione (sottoclasse-di, ISA)
- ▷ Relazione di istanziazione (istanza-di)
- ▷ Attributi di concetti generici
  (ruoli tra concetti come *essere padrone di*)
- ▷ Attributi di concetti individuali
  (ruoli tra istanze come *essere padrone di*)

KL-ONE e logica dei predicati



KL-ONE: attributi



$$\forall x\,(Cane(x) \Rightarrow \exists y\,(Ha\text{-}padrone(x, y)\ \&\ \forall z\,(Ha\text{-}padrone(x, z) \Rightarrow Umano(z))$$

Può esistere un cane senza padrone?
Può esistere un cane il cui padrone non sia umano?

$$\forall x\,(Cane(x) \Rightarrow \exists y\,(Ha\text{-}padrone(x, y)\ \&\ Umano(y))$$

Questa formula non esclude un padrone non umano

Sistemi sviluppati per il software: UML

Il Linguaggio di modellazione unificato (UML) è un linguaggio di modellazione e specifica per sistemi software, processi economici e strutture organizzative. Viene ora usato anche per le basi di conoscenza.

La notazione è in parte grafica e in parte semi-formale (come visto con KL-ONE). Alla base ci sono i diagrammi.

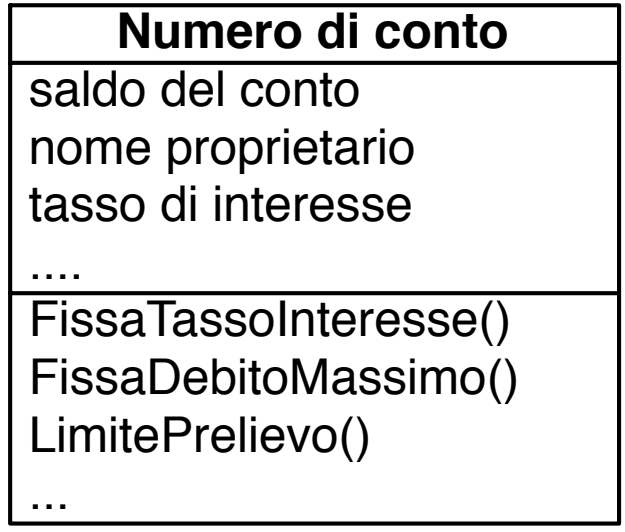
Un modello in UML può essere automaticamente trasformato in un programma software (ad es. i n Java)

UML: elementi di base

- ▷ **elementi di modellazione**: classi, interfacce, componenti...
- ▷ **relazioni**: associazioni, generalizzazioni, dipendenze...
- ▷ **diagrammi**: diagrammi di classe, diagrammi di interazione...

UML: classi

Una *classe* UML descrive un insieme di oggetti che hanno struttura, comportamento e relazioni simili.
Consiste di: *nome, attributi, operazioni*



UML: relazioni

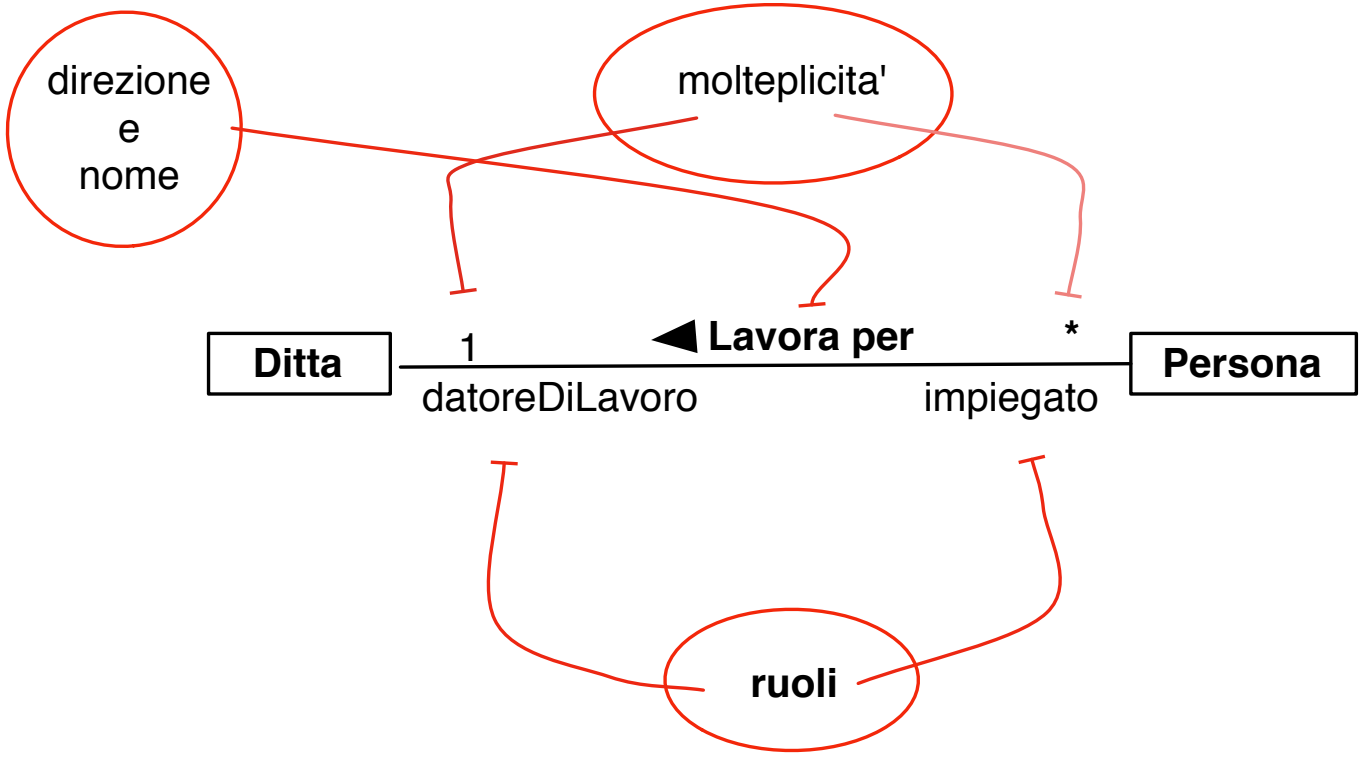
- ▷ *Associazioni*: relazioni strutturanti
- ▷ *Generalizzazioni*: relazioni di ereditarietà
- ▷ *Composizione e aggregazione*: speciali relazioni strutturali
- ▷ *Dipendenza*
- ▷ *Realizzazioni*: relazione tra una specificazione e la sua implementazione

UML: associazioni - 1

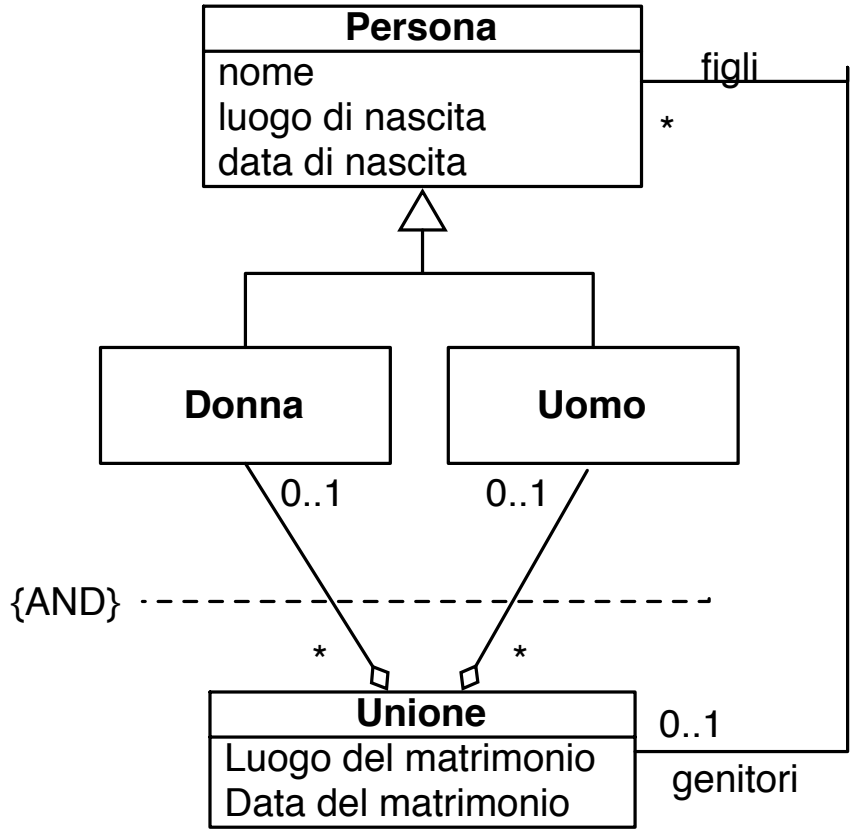
Molteplicità (per ogni istanza della relazione):

- ▷ l'espressione '$2 \ldots 5, 8$' indica che gli elementi della classe coinvolti in una istanza della relazione è un numero compreso tra 2 e 5 (compresi) oppure 8. Analogamente per '$1, 3 \ldots 5$'.
- ▷ l'espressione '$r \ldots *$' indica che r è il minimo numero di elementi della classe coinvolti e che non c'è un massimo
- ▷ '$*$' indica che il numero degli elementi della classe coinvolti nella relazione può essere zero o grande a piacere
- ▷ in caso di mancanza di indice si intende che la molteplicità è 1

UML: associazioni - 2



UML: esempio di diagramma di classi



Cosa si può dedurre?

L'inferenza in sé è un settore molto ampio che non trattiamo in questo corso. Diamo solo un paio di esempi ad indicare la varietà dei sistemi studiati.

▷ *Logica della rilevanza.*
Cattura una nozione di implicazione più vicina al senso comune
L'assunzione di base consiste nel limitare la logica classica:

a) supponi che dall'ipotesi A possiamo derivare C,

b) non segue che dalle ipotesi $A \wedge B$ possiamo derivare C

▷ *Logica non-monotona.*
Cattura la pratica comune di fare inferenze per *default* (su cui si puó cambiare idea se nuova conoscenza viene aggiunta).