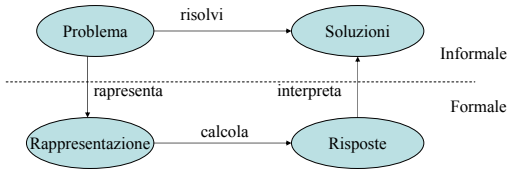
Lezione 16

Rappresentazione della conoscenza

1. Rappresentazione della conoscenza: problema trasversale

- Il problema della rappresentazione della conoscenza non riguarda solo l'ambito della IA
- è un problema generale, relativo alla costruzione di sistemi software; si può affermare che una parte rilevante dell'Ingegneria del Software riguarda la rappresentazione della conoscenza.



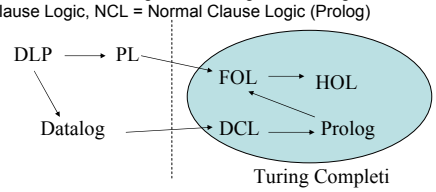
- Aspetti tipici
  - analizzare il problema e individuare le soluzioni
  - individuare il livello di astrazione
  - quale RRS (rappresentazione della conoscenza e RS)?
  - Come acquisire la conoscenza?
  - Come mantenere la conoscenza?

2. Le soluzioni: qualità/… cenni

- Qualità delle soluzioni
  - ottimale
    - misura ordinale: ordinamento delle soluzioni secondo una misura
    - misura cardinale: funzione di utilità, quasi ottimalità
  - soddisfacente (adeguata)
  - probabile (es. riconoscimento di forme)
- Alcuni trade-off
  - Qualità / disponibilità di informazioni
  - Qualità / tempo di calcolo (se la risposta deve avvenire in tempi brevi, any time algorithms)

3. Scelta del linguaggio di rappresentazione

- Le "logiche": logica = (linguaggio, RS)
- L1 più espressiva di L2 se i problemi esprimibili in L1 contengono quelli esprimibili in L2.
- Esempio: DLP = DataLog Proposizionale, PL = Propositional Logic, FOL = First Order Logic, HOL = Higer Order Logic, DCL = Definite Clause Logic, NCL = Normal Clause Logic (Prolog)



- I Turing-completi sono ugualmente espressivi; per distinguere fra essi occorrono altri criteri, ad esempio
  - naturalezza e livello di astrazione del linguaggio
    - esempio: linguaggi di alto livello rispetto ad Assembler o Macchina di Turing (vedi figura sul testo)
  - capacità di risolvere con complessità ragionevole classi di problemi
    - esempio: dimostratori diversi dimostrano "facilmente" classi di formule diverse
  - gradi di insolubilità
    - esempio: gerarchia di Kleene, l'alternanza di quantificatori misura gradi di insolubilità via via maggiori: grado($\exists x\, A$, $\forall x\, A$) < grado($\forall x \exists y\, A$, $\exists x \forall y\, A$) < ….

4. Scelta del livello di astrazione

- Livelli in una analisi in termini di RRS
  - livello della conoscenza: cosa
  - livello "simbolico" (RS): come, con quale ragionamento simbolico
- Quale livello di astrazione
  - alto
    - naturalezza, semplicità, complessità computazionale accettabile
    - scarsa predittività, genericità
  - basso
    - maggior predittività
    - problemi a livello di semplicità, comprensibilità e complessità computazionale

5. Dal problema alla rappresentazione

- La bontà di una rappresentazione dipende (ad es.) da:
  - a livello di rappresentazione della conoscenza
    - semplicità
    - aderenza al mondo
    - generalità, adattabilità, possibilità di modifiche, ...
  - a livello simbolico (di RS)
    - capacità risolutiva, ampiezza e bontà delle soluzioni
    - complessità di calcolo
    - ampiezza delle soluzioni

ESEMPIO

Si consideri il gioco del tic-tac-toe

X	O	O
	X	
X		O

Rappresentare le configurazioni del gioco:
comprensibilità della rappresentazione
x_muove_e_vince(Stato)
X_muove_e_non_perde_subito(Stato)

X	O	O
	X	
X		O

[[x,o,o], [b,x.b],[x,b,o]]

[x,o,o, b,x.b,x,b,o]

6	7	2
1	5	9
8	3	4

Quadrato magico

qm([7,2,4],[6,5,8])

Dal problema alla rappresentazione dipendenza dalla scelta del RRS

- In un approccio logicamente basato
  - IRF: quali Individui, quali Relazioni e quali Funzioni
  - **Ontologia:** quali individui
    - individuo o relazione, funzione ?

ESEMPIO. Sul tavolo sono appoggiate una mela ed una matita rosse, una pera verde ed una gomma verde. Uno dei problemi a cui siamo interessati è distinguere i colori.

su(tavolo,mela).
su(tavolo,pera).
su(tavolo,matita).
su(tavolo,gomma).
rosso(mela).
rosso(matita).
verde(pera)
verde(gomma).

Al primo ordine:
query: ? rosso(Oggetto)
? verde(Oggetto)

Al secondo ordine:
query: ? rosso(Oggetto)
? verde(Oggetto)
ma anche
? Colore(mela)

REIFICAZIONE.
Se il nostro RRS è al primo ordine, possiamo cambiare ontologia, reificando i predicati rosso, verde, cioè facendoli diventare oggetti di tipo colore.

```
su(tavolo,mela).
su(tavolo,pera).
su(tavolo,matita).
su(tavolo,gomma).
colore(rosso).
colore(verde).
diColore(rosso,mela).
diColore(rosso,matita).
diColore(verde,pera)
diColore(verde,gomma).
```

Ora possiamo scrivere al primo ordine:
? diColore(Colore,mela)

Ma non ancora:
? Proprietà(Val,mela)
che avrebbe risposte
Proprietà = su, Val = tavolo;
Proprietà = diColore, Val=rosso

6. Rappresentazione oggetto-attributo-valore e reti semantiche

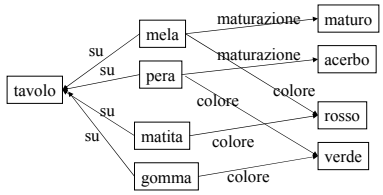
- Una rappresentazione standard è
  prop(Oggetto,Attributo,Valore)
  dove gli attributi sono la reificazione di ***proprietà*** (***predicati unari***) a cui siamo interessati
- Vantaggio:
  - mediante reificazione risponde a domande altrimenti di ordine superiore
  - elasticità, adattabilità
    - è sempre possibile variare il numero e tipo degli attributi


```
prop(mela,su,tavolo).
prop(pera,su,tavolo).
prop(matita,su,tavolo).
prop(gomma,su,tavolo).
prop(mela,colore,rosso).
prop(matita,colore,rosso).
prop(pera,colore,verde).
prop(gomma,colore,verde).
```

Ora
? prop(mela, Proprietà,Val)
che avrebbe risposte
Proprietà = su, Val = tavolo;
Proprietà = diColore, Val=rosso

6.1. Le reti semantiche

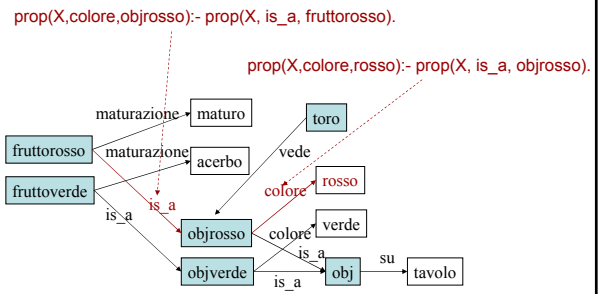
- La rappresentazione oggetto-attributo-valore è visualizzabile graficamente mediante le reti semantiche



6.2. Classificazione, Generalizzazione, reti semantiche gerarchiche

- Si possono raggruppare gli oggetti in classi con caratteristiche comuni: classificazione
- Si possono individuare proprietà comuni a diverse classi di oggetti e introdurre una classe più generale, che comprende tali classi: generalizzazione
- Ciò porta a
  - reti semantiche gerarchiche
  - proprietà derivate per eredità
    - economia nella raccolta della conoscenza
    - riuso
    - adattabilità
    - …..
  - gerarchie di classi (nel senso OO)

Esempio. Struttura ereditaria mediante is_a

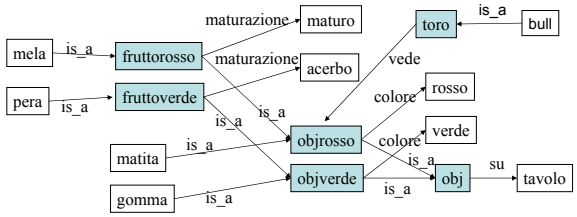


La rete delle proprietà derivate(regole generali) diventa

```
prop(X,colore,rosso):- prop(X, is_a, objrosso).
prop(X,colore,verde):- prop(X, is_a, objverde).
prop(X, su, tavolo) :- prop(X, is_a, obj).
prop(X,maturazione,maturo):- prop(X, is_a, fruttorosso).
prop(X,is_a, objrosso) :- prop(X,is_a, fruttorosso).
prop(X,maturazione,acerbo):- prop(X, is_a, fruttoverde).
prop(X,is_a, objverde) :- prop(X,is_a, fruttoverde).
prop(toro,vede,X) :- prop(X, is_a, objrosso).
```


Un'istanza particolare (proprietà primitive)

```
prop(bull, is_a, toro).
prop(mela, is_a, fruttorosso).
prop(pera, is_a, fruttoverde).
prop(matita, is_a, objrosso).
prop(gomma, is_a, objverde).
```



6.2.1. Proprietà primitive e derivate; ereditarietà

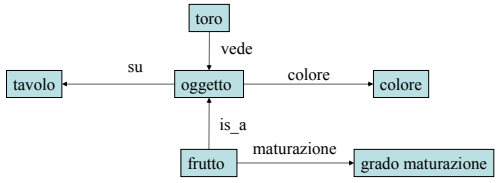
- Le proprietà primitive sono quelle che vanno inserite nella base di conoscenza per rappresentare una realtà specifica;
- Quelle derivate sono quelle che possiamo ottenere dalle primitive applicando regole generali
- L' ereditarietà deriva dalla classificazione mediante is_a e dalle regole che attribuiscono ad ogni oggetto le proprietà in base alla sua classificazione:
  - prop(X, attr, val) :- prop(X,is_a, classif)

Regole da seguire

- Associando degli attributi ad un individuo, scegliere la classe più generale alla quale appartiene con quegli attributi
  - Es: prop(mela,is_a,fruttorosso), prop(matita,is_a,objrosso)
- Non associare alle classi proprietà contingenti degli oggetti di tali classi: possono cambiare
- Assiomatizzare nella direzione causale: se modifiche in b causano modifiche in a, a ← b.

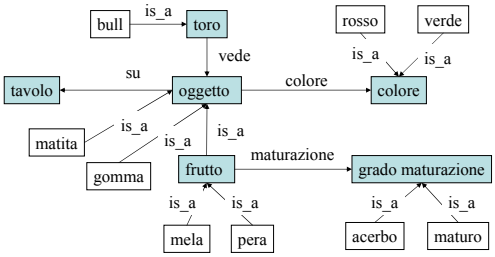
NOTA: Somiglianza con la gerarchia di classi nella programmazione ad oggetti

- Le gerarchie di classi e gli oggetti nella programmazione ad oggetti possono essere visti come una variante
- Un modo di vedere ciò è illustrato nell'esempio che segue




```
prop(X,is_a, oggetto) :- prop(X, is_a, frutto).
prop(F,maturazione,acerbo) :- prop(F, is_a, frutto),
                              prop(F, colore, verde).
prop(F,maturazione,maturo) :- prop(F, is_a, frutto),
                              prop(F, colore, rosso).
prop(A,vede,O) :- prop(A, is_a, toro),
                  prop(O, is_a, oggetto),
                  prop(O,colore,rosso).
prop(X,su,tavolo) :- prop(X, is_a, oggetto).
```

RETE GENERALE E PROPRIETA' GENERALI



La rete indica che:
matita, gomma, mela, pera, in quanto oggetti, devono avere un colore;
similmente, mela e pera, in quanto frutti, devono avere una maturazione.
Dettagliando:


```
prop(bull, is_a, toro).
prop(rosso, is_a, colore).
prob(verde, is_a, colore).
prop(acerbo, is_a, grado_maturazione).
prop(maturo, is_a, grado_maturazione).
```

```
prop(mela, colore, rosso).
prop(pera, colore, verde).
prop(matita, colore, rosso).
prop(gomma, colore, verde).
```


7. Individuazione del RRS

- Scelta di un RRS:
  - Epistemologicamente adeguato: in grado di esprimere i concetti e le relazioni necessarie per risolvere il problema
    - Espresività della “logica” usata
  - Euristicamente adeguato: in grado di usare l’informazione contenuta con risorse computazionali ragionevoli
    - Proprietà computazionali del RS
    - Euristiche e spazio di ricerca in algoritmi di ricerca
    - Rappresentazione dei dati

Principi generali per governare la complessità

- Compilare in un altro linguaggio per efficienza
- Esaminare le restrizioni del linguaggio compatiblili con il livello di astrazione per motivi di efficienza.
- Esaminare la struttura del problema per derivare procedure di inferenza specializzate.
- Ritardare le scelte (ad esempio nell’ordine delle azioni da eseguire) fino a quando possibile.
- Cache (con giudizio): memorizzare risultati che probabilmente avranno un alto grado di riutilizzo.