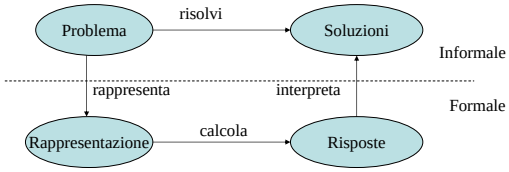
Lezione del 7 giugno

Rappresentazione della conoscenza

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

1. Il problema e le sue soluzioni
   - Qualità delle soluzioni
2. Dal problema alla rappresentazione (aspetti epistemologici):
   - Descrizione informale → rappresentazione formale
   - Livello astrazione
     - Linguaggio rappresentazione
     - Aspetti importanti per risolvere il problema
     - Conoscenze necessarie
     - Livello di dettaglio
3. Scelta di un RRS adeguato
   - Naturalezza / capacità espressiva del linguaggio di rappresentazione
   - Strategie di ragionamento appropriate
   - Aspetti critici performance

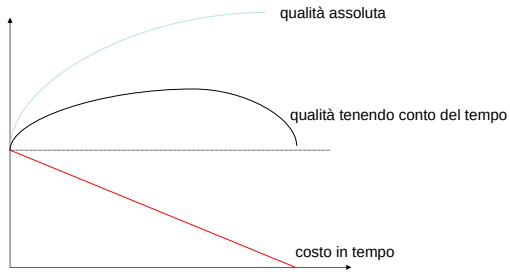
4. Aspetti collegati all'ingegneria della conoscenza
   - Come acquisire la conoscenza dagli esperti o dall'esperienza?
   - Come mantenere la conoscenza?
   - Come elaborare la conoscenza?
   - È importante che l'utente comprenda come la risposta è stata ottenuta?
   - Strumenti

1. Le soluzioni/qualità

- Dato un problema, occorre individuarne le soluzioni
  - poco è determinato, ma le scelte non sono arbitrarie: influenzano
    - la qualità delle soluzioni,
    - l'informazione contenuta
    - il metodo risolutivo stesso
  - in IA, importante il common sense reasoning
    - le soluzioni devono corrispondere a conclusioni ottenute da quanto si conosce secondo il senso comune

- Qualità delle soluzioni
  - soluzioni ottimali secondo una qualche misura della qualità
  - soluzioni soddisfacenti secondo un criterio di adeguatezza
  - approssimazioni (della soluzione ottima, secondo una misura delle qualità)
  - soluzioni probabili

- Soluzioni/decisioni
  - le soluzioni servono spesso per prendere delle decisioni (ad es. quale piano mettere in atto)
  - vi può essere un contrasto fra qualità e bontà delle decisioni
    - le informazioni contenute sono importanti per prendere decisioni, ma possono rendere più difficile il calcolo di soluzioni ottime, quasi ottime, ...
    - vi è un trade off qualità/tempo, illustrato dal grafico della pagina che segue
    - any time algorithms: meglio qualsiasi ma in tempo che ottima ma in ritardo



2. Dal problema alla rappresentazione

- Un agente possiede
  - un modello interno del mondo,
    - LIVELLO DELLA CONOSCENZA
  - attraverso il quale filtrare l'input, prendere decisioni, ...
    - LIVELLO SIMBOLICO, simboli attraverso i quali viene rappresentata e organizzata la conoscenza e si ragiona su di essa
- A livello progettuale la situazione è la solita:
  - problema --> specifica --> codifica --> computazione (soluz.)

- Nella scelta della rappresentazione simbolica (scelta RRS)
  - mappa di rappresentazione: mondo --> simboli
    - scelta oggetti e relazioni, comprensibilità
  - linguaggio di rappresentazione della conoscenza
    - naturalezza, modularità, manutenibilità
  - livello di astrazione
    - alto --> comprensibilità ma genericità, incapacità di predire
    - basso --> capacità di predire, ma complessità di calcolo
    - spesso conviene un'architettura a più strati (layers)

2.1. ESEMPIO: diverse scelte di rappresentazione

Si consideri il gioco del tic-tac-toe

X	O	O
	X	
X		O

Rappresentare le configurazioni del gioco:
comprensibilità della rappresentazione
o_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: somme in diagonale e in orizzontale = 15

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

2.2. Dipendenza dal RRS e Reti semantiche

- In un approccio logicamente basato
  - IRF: quali Individui, quali Relazioni e quali Funzioni
  - **Ontologia:** quali individui
    - individuo o relazione, funzione ?
  - RS: quali query si possono porre al RS?
    - SI/NO
    - Primo ordine
    - Ordine superiore al primo

ESEMPIO.

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.

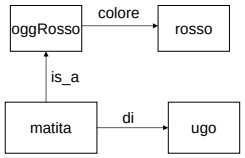
colore(rosso).
colore(verde).
diColore(rosso,mela).
diColore(rosso,matita).
diColore(verde,pera)
diColore(verde,gomma).

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

- **prop(Ogg, Attr, Val) e RETI SEMANTICHE**

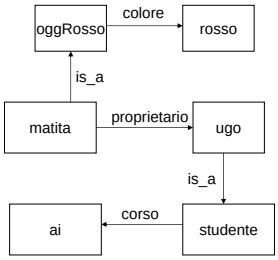
prop(Ogg, Attr, Val)

prop(Ogg, is_a, Ogg): eredità



più generale del libro:

prop(oggRosso,colore,rosso).
prop(studente,corso,ai).
prop(matita,proprietario,ugo).
prop(matita,is_a,oggRosso).
prop(ugo,is_a,studente).
prop(X,Y,Z) :- prop(X,is_a,Super),
prop(Super, Y, Z).



come il libro (più efficiente):

prop(oggRosso,colore,rosso).
prop(studente,corso,ai).
prop(matita,proprietario,ugo).
prop(matita,is_a,oggRosso).
prop(ugo,is_a,studente).
prop(X,colore,rosso) :- prop(X,is_a,oggRosso).
prop(X,corso,ai) :- prop(X,is_a,studente).

2.3. Condividere la conoscenza

- Vedere cenno su semantic WEB sul libro (o sulle slides in linea)
  - uso di Description Logic, essenzialmente una formalizzazione delle reti semantiche in una logica opportuna

3. Scelta di un RRS adeguato

3.1. Scelta di un RRS:

- Epistemologicamente adeguato: in grado di esprimere i concetti e le relazioni necessarie per risolvere il problema
  - Espressività 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

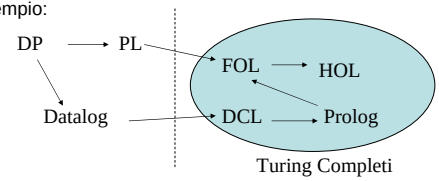
3.1.1. Sulle "logiche"

logica = (linguaggio, RS)

L1 più espressiva di L2

i problemi esprimibili in L1 contengono quelli esprimibili in L2.

Esempio:



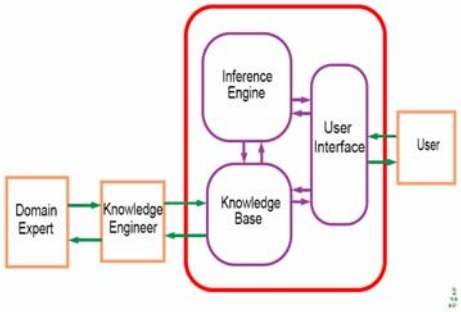
DP = DataLog Proposizionale, PL = Propositional Logic, FOL = First Order Logic, HOL = Higer Order Logic, DCL = Definite Clause Logic, NCL = Normal Clause Logic (Prolog)

- I Turing-completi sono ugualmente espressivi; per distinguere fra essi occorrono altri criteri, ad esempio
  - naturalezza e livello di astrazione del linguaggio
    - esempio: possibilità di usare la negazione, simboli di funzione, primo ordine/ordine superiore
  - capacità di risolvere con complessità ragionevole classi di problemi

Sul 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.

4. Ingegneria della conoscenza



Gli strumenti per l'ingegneria della conoscenza

- Vi sono diversi RRS
- Si può individuare un nucleo di strumenti comuni.
- Ne vedremo alcuni in termini astratti, indipendenti dal particolare RRS, utilizzando DCL come strumento base.

4.1. Meta-interpreti

- Per definire in DCL gli strumenti che considereremo, dobbiamo “implementare” in esso motori di inferenza con caratteristiche diverse.
- Uno strumento per far ciò è l’uso di un meta-interprete man mano arricchibile e adattabile.
- Partiremo da un meta-interprete di base e proseguiremo con vari arricchimenti.

4.1.1. Vanilla Meta-Interpreter

- E’ il meta-interprete di base

LINGUAGGIO OGGETTO e META LINGUAGGIO:
:- op(200, xfx, ‘<-’).
:- op(150, xfy, &).
Gli atomi a livello oggetto diventano termini nel meta-livello.

ESEMPIO:

La clausola oggetto p(f(X),Y) :- h(X), g(Y).
diventa il meta-fatto p(f(X),Y) <- h(X)&h(Y).

- Vanilla è

```
prove(true).
prove(A&B) :- prove(A),
              prove(B).
prove(H) :- H <- B,
            prove(B).
```

ESERCIZI.
Aggiungere la or nel corpo;
Aggiungere la and nella testa.

VEDIAMO ALCUNI USI

4.1.2 Introdurre la profondità di ricerca

- Potare i rami che eccedono una profondità fissata in chiamata

```
prove(true,_).
prove(A&B,D) :-prove(A,D),
               prove(B,D).
prove(H,D) :-  D >= 0,
               D1 is D-1,
               H <- B,
               prove(B,D1).
```

ESERCIZIO: Dare l’iterative deepening.

4.1.3. Ritardo dei goals

- Alcuni goal sono dichiarati ritardabili. I goal ritardabili non vengono dimostrati, ma messi in una lista.
- Per trattare efficientemente la lista, si possono usare le differenece lists.

- prove(G,D1-D2) : G conseguenza logica di KB e di D1-D2
- In particolare ci chiederemo prove(G,D-[])


```
prove(true,D-D).
prove(A&B,D1-D3) :-prove(A,D1-D2),
                   prove(B,D2-D3).
prove(H,[H|D]-D) :-  delay(H).
prove(H,D1-D2) :- H <- B,
                  prove(B,D1-D2).
```

4.1.3.1. Applicazioni

- Ritardare determinati goal aperti (che possono ad esempio entrare in loop se non ground) sino a quando non diventano ground.
- Fare ***abduzione***

4.2. Altri usi di meta-interpreti

- Vedi libro

5. Abduzione

- Deduzione: date delle assunzioni-assiomi-KB-…, quali conseguenze possiamo trarre?
  - per risolvere problemi, prendere decisioni, …
  - formalmente: KB |- ?Conseguenza?
- Abduzione: date delle osservazioni-dati di fatto, quali spiegazioni diamo di tali fatti
  - spiegazioni, cause
  - formalmente: ?Spiegaz? ∪ KB |- Osservazione

Come realizzare l'abduzione in DCL

- Abducibile: predicato che possiamo ipotizzare vero o falso (non lo sappiamo)
- Osservazione: predicato osservato come vero
- Spiegazione di Oss: S = [ab1, …, abk] tale che
  [ab1, …, abk] ∪ KB |- Oss
- Ritardando gli abducibili arrivo ad un albero di prova le cui assunzioni sono una spiegazione;
  - in DCL tutti gli alberi di prova danno tutte le spiegazioni, assumendo che una spiegazione sia un insieme finito di istanze di abducibili

- Usando l'abduzione:

```
delay(A) :- A =.. [P|_],
            abducible(P).

prove(Oss, Spiegazione-[])
```


Progetto 1

- Rendere piu' realistico l'esempio del sistema esperto Elettricista, con lo scopo di costruire un sistema di diagnosi dei guasti
  - per la parte diagnosi, applicare l'abduzione
  - si osservi che oggetti diversi hanno caratteristiche comuni (oggetti che collegano, oggetti che "lavorano"); rappresentare la conoscenza nel modo che ritenete più compatto, modulare e modificabile

Progetto 2

- Dare una rappresentazione interna dei diagrammi di classe UML e delle loro istanze
  - NB: si ha a che fare con la programmazione a oggetti, da cui rilevante la rappresentazione con frame e gerarchia di classi
  - Scopo: per semplici diagrammi, si otterranno le istanze, definendo opportunamente
    istanza(out Istanza, in Diagramma)
    Sara' possibile vincolare le istanze

Progetto X

- Fissate voi un problema di rappresentazione della conoscenza e concordate il progetto con il docente
- Aspettate la prossima lezione e poi decidete
  - ragionamento comune
  - pianificazione