Knowledge Based Systems

(Introduzione all'Intelligenza Artificiale, Parte 2)

versione 1.0

Delf

Super Engineer
Email: delfme76@hotmail.com

Knowledge Based Systems.....*1*
Disclaimer 3
Note 4
Introduzione all'Intelligenza Artificiale (parte 2):.....*6*
La logia proposizionale (e suo utilizzo in PROLOG): 11

Disclaimer

Io inizierei con una affermazione molto in voga nel panorama underground internazionale:

" …bla…bla…bla…. Le persone dovrebbero essere giudicate in base alle loro conoscenze piuttosto che ai propri denari….bla…bla…bla….le multinazionali….bla bla bla…circolazione delle idee….bla bla…".

Io questa cosa qui non l'ho mai capita. Si vuole combattere la globalizzazione dei saperi, la standardizzazione delle menti che questa società cerca di imporre, fino ad arrivare ad individuare nella differenziazione dei modi di ragionare e della qualità dei risultati intellettuali prodotti un elemento di distinguo, confronto tra gli uomini, ma anche un catalizzatore delle sinergie realizzabili tra menti diverse. Tutto questo senza rendersi conto che di fatto si sta arrivando a contrapporre un elite ad un'altra elite!! Assurdo…! L'idea di base, come spesso accade, è stata stravolta oppure viene usata come bandiera propagandistica o come scudo di menti avide che succhiano come sanguisughe un sapere maturato da altri con enorme sacrificio personale e di cui non sono degni! Ma questo è il web….si posta e via…non si può vedere il volto di chi allunga la mano e prende…scarica bit d'informazione, fa un sapiente collage, ne ricava parole con cui inizia ad esprimersi pur continuando a non sapere un cazzo di niente. Ci sarebbero molte cose da dire ma ancor di più molte cose da fare….Vedete, un sistema chiuso è come un figlio nato male già dal principio. Esso può essere efficiente ed anche efficace, capace di produrre cose buone per un certo periodo di tempo, ma alla lunga è destinato a morire. L'unica cosa che non muore mai, bensì evolve sempre, è un sistema aperto dinamico….evolve perché scambia energia con l'esterno e scambia energia con

l'esterno perché evolve. Questo è il web e ciò che esso scambia è l'informazione. E' con questo spirito che vi invito ad accogliere questi scritti che vengono concepiti con l'intento primario di non fornire l'informazione in sé ma l'informazione accompagnata da una giusta chiave di lettura. Ciò che vogliono realizzare non è la diffusione di un determinato sapere tecnologico o altro ma lo stimolo all'apertura mentale, incidere sulla forma mentis del lettore il fine più ambizioso. Perciò prendete tutto con le pinze ed "INDAGATE" …. Indagate sempre. Buona lettura a tutti.

Note

Livello di difficoltà: avanzato??!
Requisiti necessari per la lettura di questo articolo: stabilitelo voi :-)
Per tutto ciò che veniamo a sapere: ringraziamo il nostro Padre Celeste che conosce l'esigenze della nostra anima,del nostro corpo e della nostra mente, e non trascura nessuna di esse.

Questa serie di tutorial vengono concepiti con lo scopo di fornire una conoscenza di base per lo sviluppo dei K.B.S. che rappresentano, unitamente allo sviluppo di reti neurali, una delle principali tecnologie applicative dell' I.A. Il materiale che fornisco è scritto nei ritagli di tempo in cui mi diletto nel partecipare alla diffusione di quelle conoscenze difficilmente raggiungibili sia per le loro difficoltà intrinseche sia per un voluto tentativo di erezione di barriere all'entrata nel sistema "conoscenza tecnologica". Per quanto riguarda le difficoltà intrinseche alla comprensione di determinati argomenti, può essere limitata tramite l'uso di una didattica plasmata opportunamente fino

all'inverosimile. Questo comporta una enorme difficoltà nello scrivere i tutorial caratterizzati da enormi tempi di analisi e concepimento degli stessi. In più il fatto che il tempo dedicato alla loro stesura sia molto ridotto da una attività lavorativa sempre più incalzante e caratterizzata da ritmi frenetici e da un dispendio di enormi energie mentali, implica una "uscita" irregolare dei tutorial stessi e una accuratezza della forma stilistica appena sufficiente ad evitare eventuali figuracce! :-))

Perciò personalmente consiglio la lettura di questi testi a gente fornita di enorme pazienza e realmente interessate all'argomento trattato. Fiducioso di riuscire a portare a termine l'impegno preso (anche perché opportunamente stimolato da Quequero), spero che per voi rappresenti una gradita lettura (altrimenti fermiamo subito la macchina perché il tempo è denaro). Per quanto riguarda eventuali software da utilizzare più avanti vi saranno forniti direttamente senza dovere andare in cerca sul web, anche perché, pur usando lo stesso linguaggio per la scrittura di codice, useremo, nel tempo, diversi ambienti di sviluppo. Voglio ricordare che tutto ciò che ho scritto in questo capitolo è di dominio pubblico (almeno di quel pubblico che lo considera attendibile ed interessante) :))

Eventuali copie di questo materiale possono essere scambiate liberamente a patto di non violare il senso ed i contenuti originari. Confido solo nel rispetto delle proprietà intellettuali. Se qualcuno di voi avesse una preparazione in materia tale da voler contribuire alla stesura di questa serie di articoli si faccia vivo (più si è, meno si lavora!) ;)

I contenuti che saranno esposti sono da considerarsi una prima stesura ed, ovviamente, caratterizzati da una serie di difetti di cui sono già al corrente e che provvederò a correggere nel tempo! :)

Introduzione all'Intelligenza Artificiale (parte 2):

Scarica i sorgenti degli esempi

Scarica il Turbo ProLog

Nella puntata precedente abbiamo introdotto i concetti fondamentali riguardanti i K.B.S. Si è parlato molto di conoscenza e di basi di conoscenza. Proviamo a fare una classificazione della "conoscenza":

1) Associativa (questa è quella che interessa a noi poiché significa che un "esperto" fa delle associazioni su dei fatti che ricorrono simultaneamente o sequenzialmente attraverso espressioni che legano opportunamente coppie di input-output "causa-effetto").
2) Abilità motoria (questa è difficile da rappresentare. Immaginiamo il rovescio nel tennis, in tal caso la "parola" può ben poco. Anche se ti descrivono come fare tu non lo fai, è un problema di automazioni ed infatti diventa quasi un riflesso quando riesce.
3) Teoriche (per es problemi di progettazione che richiedono conoscenza teorica. Anche questa è difficile da rappresentare).

Quando descriviamo una base di conoscenza ci riferiamo ad un determinato tipo di conoscenza (per noi quella associativa) ed un determinato dominio.
Cerchiamo ora di analizzare i vari moduli di una struttura KBS. Abbiamo già visto che la prima caratteristica di tale sistema era quella di avere una netta distinzione tra la rappresentazione della conoscenza e l'uso che se ne fa.

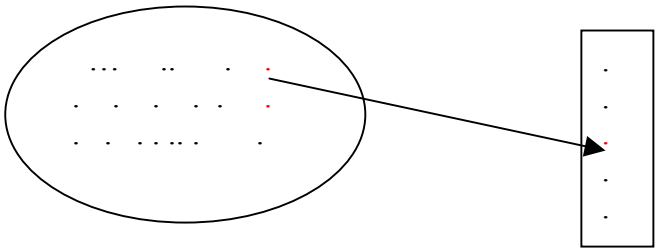
La seconda caratteristica è che nell'inferenza viene usato un approccio euristico piuttosto che algoritmico.

Vi è poi una terza caratteristica che rende pratici e applicabili i sistemi: il dominio della conoscenza (micromondo).

Ciò porta a parlare di E.S. (sistemi esperti) piuttosto che K.B.S.

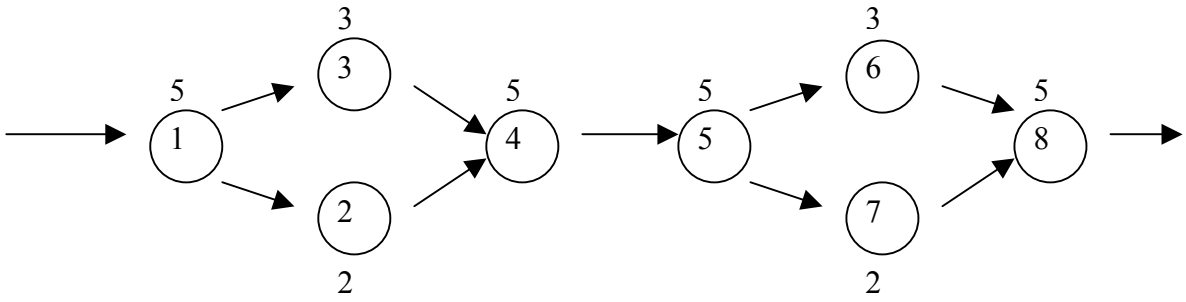
Nel micromondo ci sono decisioni da prendere che dipendono da stati di cose che avvengono nella realtà. Un ultima importante caratteristica, che è anche una ragione economica sul perché di un sistema esperto, è che deve esserci una certa complessità di rappresentazione dei legami tra decisioni e valori (ciò, infatti, giustifica il concepimento di un K.B.S.).

Per es un sistema esperto di supporto al medico potrebbe essere quello che lega sintomi a diagnosi (in tal caso le decisioni).



Non posso matchare in modo estensivo. Non posso conoscere tutti i possibili valori dei sintomi (non servirebbe o cmq potrei pensare che nel frattempo il paziente muore!) per decidere. Il sistema è tanto buono quanto più necessita di poche conoscenze (fattori) per dedurre una diagnosi (decisione).

Altro es: rete di distribuzione di un bene



I nodi sono punti in cui controllo il passaggio di un bene. Nodi di controllo (in cui si adoperano opportuni sistemi di controllo).

Supponiamo che non arrivi il bene al nodo 8 allora devo intervenire con la diagnosi (in quale tratto si è verificato il guasto?). Le decisioni saranno del tipo "ripara i seguenti tratti….".

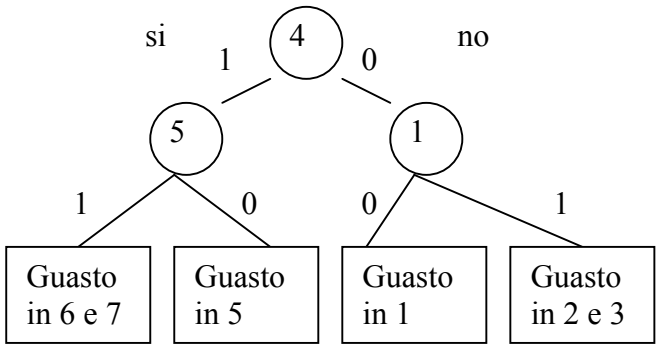
Come identificare il tratto? Potremmo fare ∀ coppia di nodi la domanda "c'è il bene? Si/No". Ma sarebbe poco efficiente !! S tratta di strutturare il modo in cui la decisione dipende da tali fattori. Ci sono diversi modi per strutturare; la conoscenza sta nel fatto di trovare il migliore. Ogni volta si verifica un guasto io devo fare un numero minimo di controlli e non tutti. Una misura dell'efficienza del sistema può essere appunto il numero medio di controlli che

faccio per trovare il guasto (oppure il caso peggiore). Nell'esempio di sopra i valori sono di tipo binario (c'è o non c'è il bene nel nodo?). Provate a strutturare una conoscenza….

Una volta fatto il sistema posso applicarlo anche in altre situazioni (tolgo la vecchia conoscenza e metto la nuova) ⇒ Qui stanno le economie di scala di chi produce tali sistemi (fare shell e venderle). Fatta una shell per lo specifico problema poi si può passare in data-base qualsiasi rete e cmq avere lo stesso funzionamento.

Parentesi: Volevo evitare di schematizzare l'esempio precedente ma mi dicono di essere meno vago e quindi…..

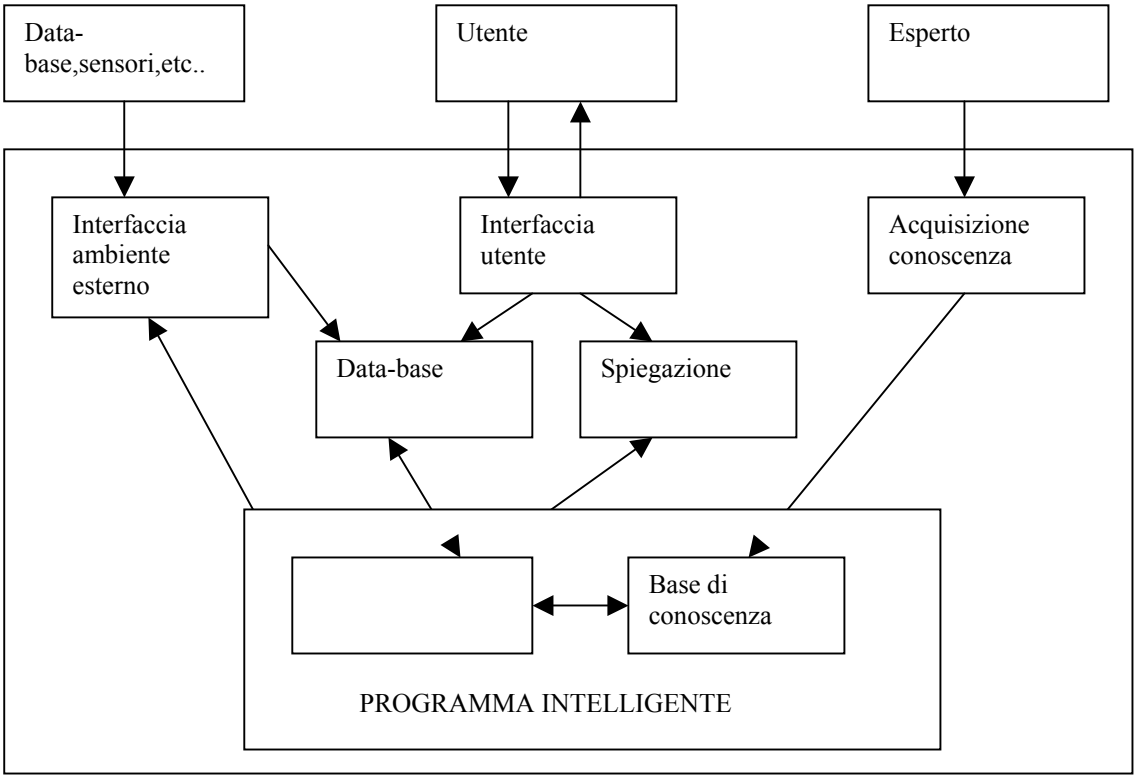
Secondo voi una rappresentazione del genere andrebbe bene?



Con due domande il sistema arriva ad una decisione.

A breve introdurremo il PROLOG ed avremo modo di scrivere del codice (le prime cazzatelle). Ma la domanda è…dobbiamo imparare un linguaggio o una scienza? Per imparare il linguaggio vi consiglio di scaricarvi qualche ambiente di sviluppo di un linguaggio di programmazione e di leggervi quale manuale di

prolog allegato! Ma tutti possono programmare realizzando il codice di un algoritmo, ma pochi creano l'algoritmo da passare al programmatore! Capite cosa intendo? ;) Quindi passiamo all'architettura di un K.B.S:



Il motore usa ciò che vi è nella base di conoscenza secondo i passi fondamentali già visti: nella base di conoscenza vi sono fatti e regole, il motore deve, ogni qual volta ha una info sullo stato dall'esterno, vedere quali regole sono applicabili ed applicarle. Tale coppia (motore,base di conoscenza) che è il cuore del sistema lo chiamiamo "programma intelligente".
Il modulo di acquisizione della conoscenza serve per acquisire la conoscenza.

Data-base: ciò che appartiene alla base di conoscenza ma riguarda solo il caso corrente. Tali dati possono anche venire dall'esterno (tramite sensori,altro KBS, data-base esterni) o dall'utente ⇒ Ho bisogno di un interfaccia utente che passa info sul caso corrente ma vuole sapere anche i risultati (decisione finale + spiegazione sul come si è arrivato al risultato).
Ci sarebbe da dire molto su ogni singolo modulo e sulle figure professionali coinvolte ma, ritenendo tale parte molto pallosa, preferisco sorvolare e dire giusto il necessario.

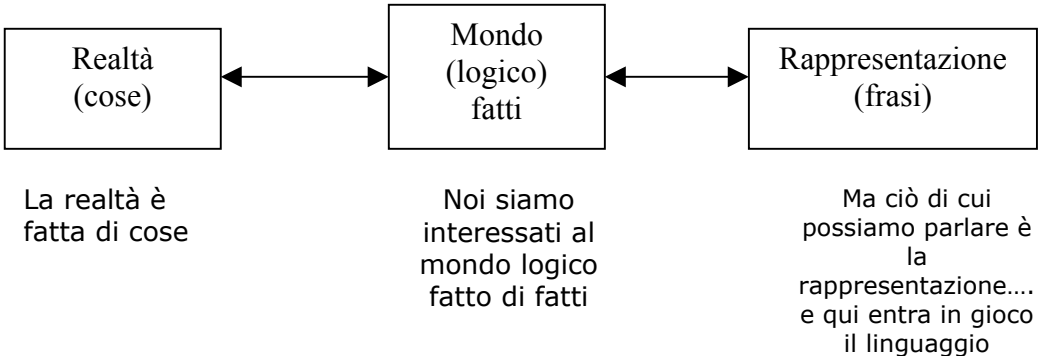
FIGURE PROFESSIONALI COINVOLTE:

Utente: Interagisce col sistema già sviluppato.
Esperto: Non conosce nulla di linguaggi e shell per rappresentare la conoscenza però è l'esperto sul dominio.
Ing. Della Conoscenza: Lavora in connubio con l'esperto. Usa metodi e strumenti per travasare la conoscenza nel sistema.
Informatico: Fornisce gli strumenti da usare all'ing. della conoscenza.

LA LOGICA PROPOSIZIONALE (e suo utilizzo in PROLOG):

1. Il mondo è tutto ciò che accade.
1.1 Il mondo è la totalità dei fatti, non delle cose.
2. Ciò che accade, il fatto, è il sussistere di stati di cose.
2.1 Ciò che accade, il fatto, è il sussistere di stati di cose.

Wittgenstein- Trattato logico-filosofico (un bel testo!)



Il linguaggio è un sistema di segni di tipo composito (cioè hanno 2 parti fondamentali):

- sintassi
- semantica

Attraverso regole di composizione, a partire da segni semplici arriviamo a rappresentazioni complesse. Di tali regole si occupa la sintassi (come formulare frasi). La semantica si occupa del significato (come si rapportano le frasi ai fatti (agli stati di cose)).

I segni atomici (simboli) devono avere un significato di per sé ma il significato è determinato dall'uso del segno nella realtà (non è univoco ma ha senso a seconda di dove lo uso). Mentre il significato delle frasi (complesso di tali simboli) dipende dalla composizione.
Mmmmm …. Vi vedo perplessi! :-))

Beh…era l'introduzione alla *logica preposizionale* che ci permetterà di rappresentare il nostro prima problema sviluppato in prolog (faremo un robot-guida). Ma basta anticipazioni….

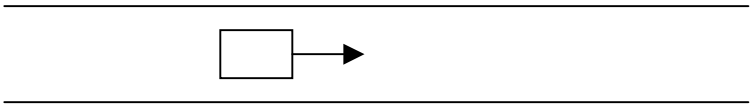
LOGICA PROPOSIZIONALE ED ASSIOMI:

1) I simboli sono chiamati a rappresentare intere proposizioni (i fatti). Rinuncio a distinguere le cose (ok ok non è il massimo ed, infatti, verrà spesso sostituita dalla logica del 1° ORDINE che è più potente. Ma ricordatevi di non sparare mai alle mosche coi cannoni e, quindi, di distinguere quando usare l'una o l'altra. Per ora essa è il primo passo da fare per iniziare a capire di programmazione logica…)
2) Le proposizioni possono essere VERE o FALSE

A partire da questo posso comporre tramite connettivi logici (logica booleana) fino ad arrivare da simboli atomici a frasi complesse.

Un KBS basato su tale logica può essere interessante perché noi con la nostra esperienza possiamo derivare la verità o falsità di un fatto a partire da altri.
Esempio: Robot-Guida

Tale robot guida un veicolo in una strada che ha una sola corsia ed un unico senso di marcia (un caso molto facile). Ovviamente vi sono altri veicoli su tale strada ed un limite di velocità. Conosciamo anche la distanza di sicurezza da mantenere e la distanza per indurre a frenare il nostro robot.



Vincoli sulla distanza di sicurezza:
100 m: rallento
50 m: mi fermo

Voglio usare la logica preposizionale per rappresentare tale conoscenza e poi la voglio esprimere in un KBS. Devo individuare i fatti atomici (il sussistere di stati di cose) e poi rappresentare frasi complesse (proposizione) tramite l'uso di connettivi logici.

Bisogna anche suddividere i fatti atomici e dire quali di questi sono di competenza dell'utente; gli altri devono essere derivati dal sistema. Ovviamente devo assolutamente supporre che sappiate fare un analisi dei requisiti di un sistema e l'analisi concettuale a fini implementativi. Per quello che andremo a fare, tale analisi è così semplice da risultare immediata… ma andando avanti non sarà più così. Perciò, se non avete già acquisito tali conoscenze vi suggerisco di dedicare meno tempo alla stesura di programmi che fanno port-scanning e di procurarvi materiale didattico più impegnativo (con meno codice e più sostanza *(ROTFL, ndQuequero)*).
Descriviamo la conoscenza (vediamo i nostri enunciati):

Fatti acquisibili dall'esterno (in genere nel nostro caso sono dei sensori, ma supponiamo che sia un semplice utente finale):
f1: c'è un veicolo a meno di 100 m
f2: c'è un veicolo a meno di 50 m
f3: viaggia al di sotto del limite
f4: viaggia al di sopra del limite
f5: viaggia al limite
f6: è fermo
oss: poiché f3,f4,f5 sono tre fatti, non posso dire che sono 2 fatti uno l'opposto dell'altro.

Decisione dedotte dal sistema:
f7: accelera
f8: rallenta
f9: mantieni velocità
f10: fermarsi
f11: ripartire

Ora abbiamo una rappresentazione del dominio (cioè dei fatti che possono accadere o non accadere). Per farlo useremo un linguaggio che ha sintassi (che si occupa di come collegare gli enunciati) e semantica (che dice che significato ha una formula) e che non è arbitraria ma composizionale ⇒ se assegno un significato ai simboli semplici ⇒ rimane assegnato il significato alla formula ottenuta dalle loro combinazioni.
Le occorrenze di tutti i fatti nel dominio sono 2^11 ma non occorrono tutti

Oss:

f3,f4,f5,f6 sono mutuamente esclusive

f1,f2 sono legate anch'esse da implicazione

f7,…,f11 sono mutuamente esclusive

Tale osservazione ci porta ad un concetto ed un metodo per descrivere gli stati del dominio.

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
0	0	0	1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	0
0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0
0	1	0	0	1	0	0	0	0	1	0

Etc etc…..la tab continua ma completatela voi (oggi mi sento pigro!)

Quando avrete finito avrete rappresentato le possibili occorrenze nel dominio (tutte le possibile interpretazioni sul dominio delle formule atomiche f1,…,f11). Usando la logica preposizionale ora posso scrivere le formule che sono vere qualunque sia lo stato del dominio (per es f2 $\Rightarrow$ f1, f2 implica f1). Però ve ne

sono altre che non usano l'implicazione (per es f7 xor f8 xor f9 xor f10 xor f11, cioè 1 solo tra questi può essere vero).
Ora troviamo una forma utile ai nostri scopi che è la forma usata dal prolog.

If ------ then

Es: a1,a2,…..,a3 ⇒ A

Questo è il modo di formulare che useremo da qui in avanti. Il sistema vuole dedurre (inferire), allora ora bisogna stabilire quali siano i fatti (enunciati dei fatti) che possono essere interpretati dall'utente e quali, invece, deve dedurre il sistema.
Per es:
f3 xor f4 xor f5 xor f6
Se l'utente dice f4=V (vera), allora il sistema deduce il resto f3,f5,f6 =F
Alla fine avrò una serie di regole che (attenzione!) devono essere vere qualunque sia lo stato del sistema!.

Parentesi: notazioni prolog
If: è espresso con ":-"
And: è espresso con ","
Or: è espresso con ";"
Etc etc…. vabbe per ora vi arrangiate leggendo qualcosa dai manuali di qualche ambiente di sviluppo prolog che trovate per la rete e poi col tempo vi allego qualcosa di specifico (un quick start prolog scritto da me).

Usando la metodologia prolog:

f7:- f4, not(f1).

….

….

….

Cioè accelera se viaggia al di sotto del limite e non c'è nessuno veicolo a meno di 100 m.

ATTENZIONE: il costrutto if ":-" che usiamo in prolog non ha una mazza in comune con il costrutto if then del c/c++ e cazzi vari.

Quello significava: se accade "questo" allora fai "quello" mentre questo è usato nella forma "questo" se e solo se accade "quest'altro"!. La differenza è sostanziale anche se sottile.

Alla fine di tutte le formule devo arrivare a garantire la "completezza" (per ogni situazione prende una decisione)e la "consistenza" (coerenza).

Costruita la base di conoscenza pensiamo ad un interfaccia utente (ricordate che l'accadimento di alcuni fatti viene dichiarato dall'utente).

Per es il fatto che ci sia o meno un veicolo a meno di 100 m:

f1:-write("c'è un veicolo a meno di 100m? Rispondi s/n"), readln(X),X="s".

Iniziate a ragionare da programmatori che usano un linguaggio di tipo logico. La formula f1 sarà vera solo se saranno veri tutti quegli "and" concatenati fra loro! Se l'utente risponde "n" ciò causerà la seguente situazione:

write("cazzi vari…."): è sempre vera per il prolog

readln(X): pure vera

X="s": falsa

Ciò implica la formula f1 è falsa!

Vabbè ci vorra un po' di tempo.

Ovviamente X è una variabile, write stampa su schermo, readln (legge).

Questa interfaccia utente è stupida! Perché? Mi richiede le cose ogni qual volta necessita di saperle e, quindi, mi richiede più volte cose a cui ho già risposto!. Se ricapiterà una decisione che per essere presa è legata ad un valore di verità o falsità di f1 lui mi rifarà la stessa domanda!.

Poi vi è una soluzione semi-intelligente (mette in un Database "db" se un fatto è vero o falso).

Poi una intelligente (non appena do un informazione su verità o falsità di un fatto tramite interfaccia lui da V o F anche ai fatti legati a quello immesso!). Avevamo visto infatti che se f3 è V allora f4,f5 sono F e non mi chiederà più f4 ed f5 poiché se le deduce da solo).

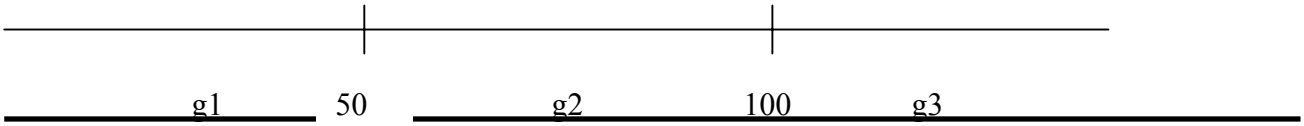
Vabbè… mi piace pensare che vi ci divertirete sopra da soli!

Io ora vi do il codice:

Ps:

Avete riflettuto su come esprimere al meglio f1 e f2 ? Tra di loro non erano mutuamente esclusivi!! Ma a noi ciò non sta bene…

Come fare?



Questi sono mutuamente esclusivi ;)

```
F1:-g1.
F1:-g2.
F2:-g1.
```

Vi avevo detto che l'or in prolog è espresso attraverso un ";"
In realtà ora avete capito che si può esprimere anche così. Per f1 scriverò due formule adiacenti (non importa chi prima e chi dopo) divise da un punto. Ciò significa che la verità di f1 (f1=V) può essere stabilità indifferentemente da una di quelle due formule (basta che una delle due sia vera) e la falsità da entrambe!.
Ricordate che nella lezione precedente avevo detto che il prolog mette a disposizione diverse aree di dichiarazione. Una per i Predicati (vanno dichiarati e scritti nella sezione PREDICATES), una per le clausole (sezione CLAUSES), una per il database (sezione DATABASE), una per gli obiettivi (sezione GOALS). Ps: ve ne sono altre ma per ora non sono necessarie.

ROBOT-GUIDA FILE:

```
PREDICATES
accelerare
rallentare
```

fermarsi
mantenere_v
ripartire
f1
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
g1
g2
conosci_g1
conosci_g2
conosci_f3
conosci_f4
conosci_f5
conosci_f6
run
database
g1c
g1v

```
g2c
g2v
f1c
f1v
f2c
f2v
f3c
f3v
f4c
f4v
f5c
f5v
f6c
f6v

CLAUSES
g1:- not(g1c),write("c'è un veicolo a meno di 50m? rispondi
s/n"),asserta(g1c),nl,readln(X),X="s",asserta(g1v),conosci_g2.
g1:-g1v.
g2:-not(g2c),write("c'è un veicolo tra 50 e 100m? rispondi
s(n"),asserta(g2c),nl,readln(X),X="s",asserta(g2v),conosci_g1.
g2:-g2v.
conosci_g1:-g1c.
conosci_g1:- asserta(g1c).
conosci_g2:-g2c.
conosci_g2:-asserta(g2c).
```

```
f1:-g1.
f1:-g2.
f1:-f2.
f2:-g1.
f3:- not(f3c),write("viaggia al di sopra del limite? rispondi
s/n"),nl,asserta(f3c),readln(X),X="s",asserta(f3v),conosci_f4,conosci_f5,conos
ci_f6.
f3:-f3v.
conosci_f3:-f3c.
conosci_f3:- asserta(f3c).
f4:-not(f4c),write("viaggia al di sotto del limte? rispondi
s/n"),nl,asserta(f4c),readln(X),X="s",asserta(f4v),conosci_f3,conosci_f5,conos
ci_f6.
f4:-f4v.
conosci_f4:-f4c.
conosci_f4:- asserta(f4c).
f5:-not(f5c),write("viaggia sul limite? rispondi
s/n"),nl,asserta(f5c),readln(X),X="S",asserta(f5v),conosci_f3,conosci_f4,conos
ci_f6.
f5:-f5v.
conosci_f5:-f5c.
conosci_f5:- asserta(f5c).
f6:-not(f6c),write("è fermo? rispondi
s/n"),nl,asserta(f6c),readln(X),X="s",asserta(f6v),conosci_f3,conosci_f4,conos
ci_f5.
f6:-f6v.
```

```
conosci_f6:-f6c.
conosci_f6:- asserta(f6c).
f7:-not(f1),f4.
f8:-f4,f1.
f9:-f5,not(f1).
f10:-f2,f3.
f10:-f2,f5.
f11:- f6,not(f2).
accelerare:-f7.
rallentare:-f8.
mantenere_v:-f9.
fermarsi:-f10.
ripartire:-f11.
run:-retractall(_),fail.
run:-accelerare,write("accelerare").
run:-rallentare,write("rallentare").
run:-mantenere_v,write("mantenere velocit… corrente").
run:-fermarsi,write("fermarsi").
run:-ripartire,write("ripartire").
GOAL
run.
```

Studiatevi il codice….cercate di capirlo da soli (anche se non doveste riuscire sarà cmq uno sforzo molto produttivo al fine di abbattere il primo impatto con il linguaggio).

Nella prossima lezione troverete una spiegazione del codice fatta bene insieme ad una breve guida sul prolog…anche se vi invito a lavorarci sopra da voi (casomai mandate email se avete domande di senso compiuto).

Osservazioni:

- Il codice può essere migliorato (dite voi dove e come).
- Troverete dei costrutti nuovi come retractall , fail , assert.
- Come è stato concepito il goal?
- Come lavora l'interfaccia utente concepita?
- L'importanza dei simboli "conosci_fi" ?
- Discutete su problematiche e deduzioni che vi vengono in mente

Ps: In chat qualcuno mi ha detto che, non trovando il turbo-prolog, si era scaricato il visual prolog e chiedeva se andava bene. Ma certamente! Usate quello che volete ….tutto dipende da voi ….non abbiate limiti (solo che al furbone del visual prolog suggerisco di mettere il costrutto "nondeterm" davanti ai predicati altrimenti il prog non girerà mai!) ;)
Quindi ritenetevi di usare ciò che più vi piace e di andare dove vi tira di più ;)
Poiché il visual-prolog è un buon ambiente di sviluppo vi consiglio di scaricarvi questo programma qui. Però vi voglio dire qualcosa……

Ricordiamo i limiti sulla licenza del visual-prolog:
Visual Prolog v.5.2 Personal Edition.
For non-commercial and educational use only.
You are NOT allowed to distribute or give away executables created with this version.

You may NOT use this version for any commercial purpose, or in any entity which has a commercial purpose.
You are allowed to use it at home for your personal education, and you may use it at educational institutions for teaching Visual Prolog.
For a commercial version, see http://www.visual-prolog.com

Prego tutti di rispettare le volontà della casa distributrice del software.
E', infatti, una delle poche che permettono un utilizzo di un programma in modalità demo ma rilasciato con caratteristiche "full working" per uso personale o didattico. In più allego a questo documento una copia del turbo-prolog (ottimo tool qualora vi scazzaste di scaricare 18-20 MB di visual prolog).
Ritornando al nostro robot-driven…..
provate a dare tale sequenza:
n
n
n
n
(cioè rispondete no a tutte le prime domande che vi fa)
lui risponderà "no".
Perchè non matcha nessuna clausola per run, che sia utilizzabile!!
Invece se diamo:
s
s

Lui risponderà prima con "rallentare" e poi con yes (cioè ha matchato qualkosa per il costrutto run!). Ora mi fermo perché vi ho gia dato troppe dritte….e vi rimando alla prossima uscita del corso di A.I.

Ps-ultimo: se trovate difficoltà sia nella parte riguardante il codice che in quella riguardante la progettazione concettuale della base di conoscenza provate a contattarmi via email. Cercherò, dove possibile, di rispondere a tutti quelli che avranno bisogno (vi chiedo solo di racimolare tutti i dubbi in una sola email… cosi non mi intasate la posta).
Alla prox….
Delf