ESERCIZI PER IL CORSO DI INTELLIGENZA ARTIFICIALE

II parte del corso – Rappresentazione della conoscenza

Anno accademico 2003/2004

M. Simi (*)

(*) La dott.ssa Chiara Renso ha contribuito alla stesura di questi esercizi nell'a.a. 1997/98. Il dott. Andrea Bracciali ha contribuito alcuni esercizi e soluzioni negli anni 1998-2000. Il dott. Fabio Ganovelli ha contribuito alcuni esercizi e soluzioni nell'anno 2001/02.

Calcolo Proposizionale

Tabelle di verità

Mostrare con le tabelle di verità che le seguenti formule sono valide:

1. $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$
2. $\neg (p \vee q) \Leftrightarrow \neg p \wedge \neg q$
3. $(p \Rightarrow q) \Leftrightarrow \neg p \vee q$
4. $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
5. $p \wedge \neg p \Leftrightarrow$ false

p	q	$\neg(p \wedge q)$	$\neg p \vee \neg q$	1
f	f	t	t	t
f	t	t	t	t
t	f	t	t	t
t	t	f	f	t

p	q	$\neg (p \vee q)$	$\neg p \wedge \neg q$	2
f	f	t	t	t
f	t	f	f	t
t	f	f	f	t
t	t	f	f	t

p	q	$(p \Rightarrow q)$	$\neg p \vee q$	3
f	f	t	t	t
f	t	t	t	t
t	f	f	f	t
t	t	t	t	t

p	q	r	$p \vee (q \wedge r)$	$(p \vee q) \wedge (p \vee r)$	4
f	f	f	f	f	t
f	f	t	f	f	t
f	t	f	f	f	t
f	t	t	t	t	t

p	q	$(p \Rightarrow q)$	$\neg p \vee q$	3
f	f	t	t	t
f	t	t	t	t
t	f	f	f	t
t	t	t	t	t

p	$p \wedge \neg p$	false	5
t	f	f	t
f	f	f	t

Validità e insoddisfacibilità

Dire se le seguenti formule sono valide o insoddisfacibili o nè l'una nè l'altra.

1. Smoke $\Rightarrow$ Smoke
2. Smoke $\Rightarrow$ Fire
3. (Smoke $\Rightarrow$ Fire) $\Rightarrow$ ($\neg$Smoke $\Rightarrow$ $\neg$Fire)
4. Smoke $\vee$ Fire $\vee$ $\neg$Fire
5. ((Smoke $\wedge$ Heat) $\Rightarrow$ Fire) $\Leftrightarrow$ ((Smoke $\Rightarrow$ Fire) $\vee$ (Heat $\Rightarrow$ Fire))
6. (Smoke $\Rightarrow$ Fire) $\Rightarrow$ ((Smoke $\wedge$ Heat) $\Rightarrow$ Fire)
7. ((Smoke $\wedge$ Heat) $\Rightarrow$ Fire) $\Rightarrow$ (Smoke $\Rightarrow$ Fire)
8. Big $\vee$ Dumb $\vee$ (Big $\Rightarrow$ Dumb)
9. (Big $\wedge$ Dumb) $\vee$ $\neg$Dumb

La formula 1 è valida.
La formula 2 è altro perché c'è un modello in cui è vera e uno in cui è falsa. Infatti per Smoke = T e Fire = T, 2 è vera, mentre per Smoke = T e Fire = F, 2 è falsa.

La formula 3 è altro perché per Smoke = T e Fire = T ho vero e per Smoke = F e Fire= T ho falso.

La formula 4 è valida.

La formula 5 è valida.

La formula 6 è valida.

La formula 7 è altro perché per Smoke = T, Heat = T e Fire = T la formula è vera, mentre per Smoke = T, Heat = F e Fire = F la formula è falsa.

La formula 8 è valida.

La formula 9 è altro perché per dumb = F è vera mentre per Dumb = T e Big = F è falsa.

Equivalenza e conseguenza logica

Dire quali delle seguenti affermazioni sono equivalenti tra di loro e quali non lo sono, motivandole conclusioni:

1. $(p \vee q) \Leftrightarrow \neg (p \wedge q)$
2. $\neg((p \vee q) \Leftrightarrow (p \wedge q))$
3. $\{(p \vee q)\} \models \neg(p \wedge q)$ e $\{\neg (p \wedge q)\} \models (p \vee q)$

Le prime due sono formule equivalenti, come si vede dalle loro tabelle di verità, in quanto assumono gli stessi valori di verità in tutte le interpretazioni.

p	q	p $\vee$ q	$\neg$(p $\wedge$ q)	1
t	t	t	f	f
t	f	t	t	t
f	t	t	t	t
f	f	f	t	f

p	q	p $\vee$ q	p $\wedge$ q	2
t	t	t	t	f
t	f	t	f	t
f	t	t	f	t
f	f	f	f	f

La terza è diversa dalla 1 (e dalla 2) in quanto è una meta-asserzione che dice che $(p \vee q)$ e $\neg(p \wedge q)$ sono vere esattamente nelle stesse interpretazioni, un'affermazione falsa, mentre la verità della 1 (o della 2) dipende dall'interpretazione.

Unicorno

Il fatto che l'unicorno è mitico, che l'unicorno è magico e che l'unicorno ha le corna, sono conseguenza logica dei seguenti fatti?
" Se l'unicorno è mitico allora è immortale. Se non è mitico allora è un mammifero mortale. Se l'unicorno è o immortale o un mammifero allora ha le corna. L'unicorno è magico se ha le corna."

Soluzione:

1. Mitico $\Rightarrow \neg$Mortale
2. $\neg$Mitico $\Rightarrow$ Mortale $\wedge$ Mammifero
3. $\neg$Mortale $\vee$ Mammifero $\Rightarrow$ Corna
4. Corna $\Rightarrow$ Magico

'Mitico' non è conseguenza logica, 'Corna' e 'Magico' lo sono. Infatti costruendo la tabella di verità e limitandoci alle interpretazioni che sono modelli per la base di conoscenza, dalle colonne 1, 4 e 5 trovo che 'Mitico' non è sempre vero, mentre 'Corna' e 'Magico' lo sono.

mitico	mortale	mammifero	corna	magico
f	t	t	t	t
t	f	t	t	t
t	f	f	t	t

Quante interpretazioni?

Considerare un mondo in cui ci sono solo 4 possibili fatti: A,B,C,D
1. quante sono le possibili interpretazioni?
2. quanti sono i modelli di $(A \wedge B)$?
3. quanti i modelli di $(A \vee B)$?
4. quanti i modelli di $(A \wedge B \wedge C)$?

Soluzione:

1) Le possibili interpretazioni sono $2^4 = 16$
2) I modelli di $(A \wedge B)$ sono 4: cioè devo avere A e B veri e C e D qualunque.
3) I modelli di $(A \vee B)$ sono 16-4=12: tutti tranne quando ho A falso e B falso
4) I modelli di $(A \wedge B \wedge C)$ sono 2: devo avere A e B e C veri e D può essere sia vero che falso.

Quali interpretazioni?

Siano A e B definite come segue

A. $\forall x \exists y (x \geq y)$
B. $\exists y \forall x (x \geq y)$

1. Dire cose significano A e B, assumendo che le variabili hanno valore sui naturali e il simbolo $\geq$ significhi maggiore o uguale.
5. A è vera sotto questa interpretazione?
6. B è vera sotto questa interpretazione?
7. A |= B?
8. B |= A?

Soluzione:

1. Il significato di A e B in questa interpretazione è:
A. Per ogni numero naturale x esiste un numero naturale y che è più piccolo o uguale di x.
Ogni naturale è maggiore o uguale ad un altro.
B. Esiste un naturale x tale che per ogni naturale y, x è maggiore o uguale a y.
Esiste un naturale minore o uguale a tutti.

2. A è vera sotto questa interpretazione? SI

3. B è vera sotto questa interpretazione? SI

4. A |= B?
NO. Supponiamo di cambiare interpretazione e di interpretare il simbolo $\geq$ come minore o uguale
Allora A è vera, B è falsa.

5. B |= A ? SI. Si veda la dimostrazione più avanti.

Ragionamento di tipo model checking

Jones, Smith e Clark sono programmatore, ingegnere della conoscenza e manager (non necessariamente in quest'ordine). Jones ha un debito di lire 10.000 con il programmatore. La moglie del manager non vuole che suo marito prenda in prestito dei soldi. Smith non è sposato. Che lavoro fa ciascuno?

Soluzione:

Indichiamo con

J = Jones	M = Manager
S = Smith	P = Programmatore
C = Clark	K = Ingegnere

Abbiamo nove simboli di proposizione e quindi 6 possibili combinazioni legali del tipo
$Jx \wedge Sy \wedge Cz$ con $x,y,z \in \{M,P,K\}$ e $x \neq y \neq z$

1) $(JM \wedge SP \wedge CK) \vee$
$(JP \wedge SM \wedge CK) \vee$
$(JK \wedge SP \wedge CM) \vee$
$(JK \wedge SM \wedge CS) \vee$

$(JM \wedge SK \wedge CP) \vee$
$(JP \wedge SK \wedge CM)$

Inoltre:

2) $\neg JP$ Jones non è il programmatore avendo un debito con lui
3) $\neg JM$ Jones non è il manager perché ha
4) $\neg SM$ Smith non è il manager perché questi è sposato

Da 1) 2) 3) e 4) deriva che l'unica combinazione possibile è $(JK \wedge SP \wedge CM)$ e quindi: Jones è ingegnere, Smith è programmatore e Clark è manager.

Algoritmo GSAT

SAT è l'abbreviazione per il problema della soddisfacibilità: data una formula proposizionale determinare se è soddisfacibile e se lo è dire quali proposizioni devono essere vere perché la formula lo sia. 3SAT è SAT per formule in 3-CNF cioè formule del tipo:

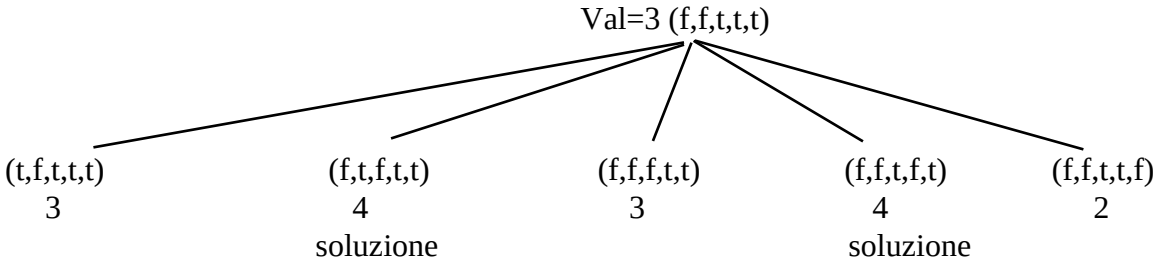
$$(P \vee Q \vee \neg S) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee \neg R \vee \neg S) \wedge (P \vee \neg S \vee T)$$

una forma a clausole in cui ogni clausola ha esattamente 3 disgiunti (con clausole ternarie).

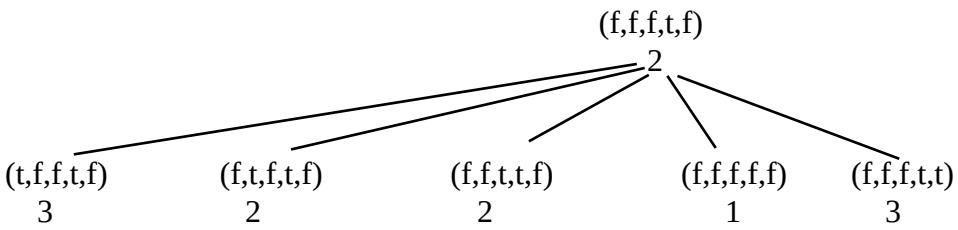
GSAT è un algoritmo per problemi 3SAT che implementa hill-climbing con random restart. Si parte da un assegnamento a caso di valori di verità ai simboli proposizionali. La funzione di valutazione misura il numero di clausole soddisfatte (4 sul goal). L'operatore consiste nel cambiare il valore di verità di un simbolo proposizionale (con 5 simboli di proposizione il fattore di diramazione è n=5).

a) Tracciare l'algoritmo sull'esempio.

Indichiamo lo stato con una quintupla che corrisponde ai valori di verità dei simboli di proposizione nell'ordine (P,Q,R,S,T). Partiamo con (f,f,t,t,t) e cioè P=Q=false, R=S=T=true



Consideriamo un altro caso:



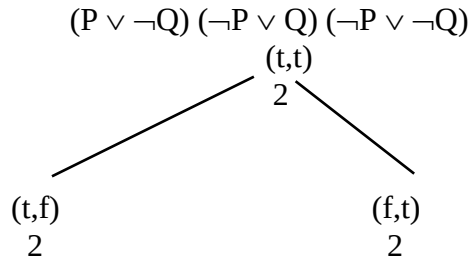
a) L’algoritmo GSAT è corretto?

Si, se trova che una interpretazione è una soluzione allora effettivamente lo è.

b) L’algoritmo GSAT è completo? Se la formula di partenza è soddisfacibile, trova una soluzione?

c) È possibile una situazione in cui ogni operatore non migliora la valutazione, anche se la soluzione esiste?

Esempio:



Qui la soluzione è (f,f), ma l’algoritmo non migliora. Occorre ripartire con uno stato generato a caso.

Adamo e i suoi figli (conseguenza logica)

Date la seguente KB di formule del FOL:

$\forall x$ Uomo(x) $\Rightarrow$ Uomo(figlio(x))

Uomo(figlio(Adamo))

dire se le formule seguenti formule sono conseguenza logica delle precedenti, giustificando formalmente la risposta:

1. x Uomo(x)
2. $\forall x$ Uomo(Adamo) $\Rightarrow$ Uomo(figlio(x))
3. $\exists x$ Uomo(x) $\Rightarrow$ Uomo(figlio(x))

Soluzione:

1. La formula $\exists$ x Uomo(x) è conseguenza logica della KB in quanto in tutti i possibili modelli della KB deve esistere almeno un individuo nel dominio di interpretazione, l’individuo denotato da figlio(Adamo), che soddisfa il predicato Uomo. Diversamente la seconda formula della KB non sarebbe vera.

2. La formula $\forall x$ Uomo(Adamo) $\Rightarrow$ Uomo(figlio(x)) non è conseguenza logica della KB, in quanto esistono modelli di KB in cui Uomo(Adamo) (l’antecedente) è vero ma per qualche individuo i del dominio Uomo(i) può essere falso.

3. La formula $\exists x$ Uomo(x) $\Rightarrow$ Uomo(figlio(x)) è conseguenza logica della KB, in quanto in tutti i modelli è vero Uomo(figlio(Adamo)), e quindi la formula è soddisfatta quantomeno da Adamo [Uomo(Adamo) $\Rightarrow$ Uomo(figlio(Adamo))] essendo il conseguente vero.

Rappresentazione della conoscenza e logica del primo ordine

Rappresentazione in FOL di frasi in linguaggio naturale

Rappresentare in FOL, con un vocabolario consistente da definire:

a) ”Non tutti gli studenti scelgono IA1 e IALab”
b) “Solo uno studente è stato bocciato”
c) “Ogni persona a cui non piace nessun fumatore è intelligente”
d) “Nessuno ama un fumatore maleducato”
e) “C’è una donna che ama tutti gli uomini vegetariani”
f) “C’è una donna che ama solo uomini vegetariani”
g) “Nessuno ama un professore a meno che questo non sia intelligente”
h) “I politici possono ingannare alcuni in tutte le occasioni e possono ingannare tutti in qualche occasione, ma non possono ingannare tutti in tutte le occasioni.”
i) “Tutti i tedeschi parlano lo stesso linguaggio”

Soluzioni:

a) $\neg\ \forall x\ \text{Studente}(x) \Rightarrow \text{Sceglie}(x,\text{IA1}) \wedge \text{Sceglie}(x, \text{IAlab})$

b) $\exists x\ (\text{Bocciato}(x) \wedge\ \forall y\ \text{Bocciato}(y) \Rightarrow y=x)$
oppure, con una notazione alternativa,
$\exists!\ x\ \text{Bocciato}(x)$

d) $\forall x\ \text{Persona}(x) \wedge (\forall y\ \text{Fumatore}(y) \Rightarrow \neg\text{Ama}(x, y)) \Rightarrow \text{Intelligente}(x)$

c) Dato un qualunque fumatore maleducato, non c’è nessuno che lo ami
$\forall y\ \text{Fumatore}(y) \wedge\ \text{Maleducato}(y) \Rightarrow \neg\ \exists x\ \text{Ama}(x, y)$
Nota: $\neg\ \exists x\ (\forall y\ \text{Fumatore}(y) \wedge\ \text{Maleducato}(y) \Rightarrow \text{Ama}(x, y))$ può essere letto come “Non esiste qualcuno che ama tutti i fumatori maleducati”.

e) $\exists\ x\ \text{Donna}(x) \wedge \forall y\ \text{Vegetariano}(y) \wedge \text{Uomo}(y) \Rightarrow \text{Ama}(x,y)$

f) $\exists\ x\ \text{Donna}(x) \wedge \forall y\ \text{Ama}(x, y) \Rightarrow \text{Uomo}(y) \wedge\ \text{Vegetariano}(y)$

g) $\forall x\ \forall y\ \text{Ama}(x, y) \wedge\ \text{Professore}(y) \Rightarrow \text{Intelligente}(y)$
Nota: la g) è equivalente alla formulazione:
$\forall y\ \neg\exists\ x\ \text{Ama}(x, y) \wedge\ \text{Professore}(y) \wedge \neg\ \text{Intelligente}(y)$

h) $\forall x\ \text{Politico}(x) \Rightarrow \exists y\ \forall o\ \text{Inganna}(x, y, o) \wedge$
$\forall x\ \text{Politico}(x) \Rightarrow\ \exists o\ \forall y\ \text{Inganna}(x, y, o) \wedge$
$\forall x\ \neg(\text{Politico}(x) \Rightarrow \forall y\ \forall o\ \text{Inganna}(x, y, o)$

i) $\exists\ l\ \ \forall x\ \text{Tedesco}(x) \Rightarrow \text{Parla}(x, l)$ e non
$\forall x\ \exists\ l\ \text{Tedesco}(x) \Rightarrow \text{Parla}(x, l)$
che vuol dire che ogni tedesco parla un linguaggio che può essere diverso da individuo a individuo.

Scimmie e tartarughe

Tradurre i seguenti enunciati nella logica dei predicati; in caso di frase ambigua, e la prima lo è, dare le due possibile trascrizioni.

a. Tutte le scimmie sono fuggite su un albero.

A1. $\forall x\ (\text{scimmia}(x) \Rightarrow (\exists y\ \text{albero}(y) \wedge \text{fuggita\_su}(x, y)))$ [8]
Per ogni scimmia esiste un albero su cui questa è fuggita

A2. $\exists y\ (\text{albero}(y) \wedge (\forall x\ \text{scimmia}(x) \Rightarrow \text{fuggita\_su}(x, y)))$ [5]
Esiste un albero su cui tutte le scimmie sono fuggite

b. Esiste una tartaruga che è più vecchia di ogni essere umano.

B1. $\exists x\ (\text{tartaruga}(x) \wedge (\forall y\ \text{uomo}(y) \Rightarrow \text{più\_vecchia}(x, y)))$

Tra parentesi quadre il numero di studenti che hanno proposto la soluzione. Riporto qui sotto una serie di errori comuni e tentativo di rendere la formula in linguaggio naturale.

Per le scimmie:

$\forall x\ (\text{scimmia}(x) \wedge \exists y\ (\text{albero}(y) \Rightarrow \text{fuggita\_su}(x, y)))$ [3]
Tutti sono scimmie e se esiste un albero ogni scimmia ci fugge sopra. Questo eventuale albero non può che essere un albero-scimmia, visto che esistono solo scimmie, e deve essere in grado di fuggire su se stesso.

$\forall x\ (\text{scimmia}(x) \wedge \exists y\ (\text{albero}(y) \wedge \text{fuggita\_su}(x, y)))$ [2]
Tutti sono scimmie ed esiste un albero (necessariamente un albero-scimmia) su cui ogni scimmia fugge sopra (compreso lui).

$\forall x\ \forall y\ ((\text{scimmia}(x) \wedge \text{albero}(y)) \Rightarrow \text{fuggita\_su}(x, y))$ [3]
Ogni scimmia sale su ogni albero

$\forall x\ \exists y\ (\text{scimmia}(x) \wedge \text{albero}(y) \wedge \text{fuggita\_su}(x, y)$ [2]
Tutti sono scimmie ed esistono alberi (magari diversi da scimmia a scimmia) su cui fuggono.

$\exists y\ \forall x\ (\text{scimmia}(x) \wedge \text{albero}(y) \wedge \text{fuggita\_su}(x, y)$ [3]
Tutti sono scimmie ed esiste un albero su cui tutti fuggono.

$\forall x\ \forall y\ (\text{scimmia}(x) \Rightarrow \text{albero}(y) \wedge \text{fuggita\_su}(x, y))$ [1]
Se x è una scimmia, un qualunque y è un albero su cui x fugge sopra (e quindi anche lei stessa).

$\forall x\ \exists y\ (\text{scimmia}(x) \wedge \text{albero}(y) \Rightarrow \text{fuggita\_su}(x, y))$ [4]
Per ogni scimmia se esiste un albero questa ci fugge sopra (ma l'albero potrebbe non esistere).

$\forall x\ \exists y\ (\text{scimmia}(x) \vee \text{albero}(y) \Rightarrow \text{fuggita\_su}(x, y))$ [1]
Tutte le scimmie fuggono su qualcosa e se esiste un albero tutti ci fuggono sopra.

$\exists y\ \forall x\ (\text{albero}(y) \wedge \text{scimmia}(x) \Rightarrow \text{fuggita\_su}(x, y))$ [2]

Esiste un individuo che, se albero, tutte le scimmie ci fuggono sopra (ma questo potrebbe non essere un albero)

$\exists$albero $\forall$scimmia fuggita_su(scimmia, albero) [2]
Non si parla né di scimmie né di alberi ma esiste qualcosa su cui tutti fuggono.

$\forall$scimmia $\exists$albero fuggita_su(scimmia, albero) [2]
Non si parla né di scimmie né di alberi ma ogni cosa ha qualcosa su cui fuggire.

$\exists$y albero(y) $\Rightarrow$ $\forall$x scimmia(x) $\wedge$ fuggita_su(x, y)) [3]
Se esiste un albero, tutti sono scimmie e ci fuggono sopra.

Altre soluzioni più fantasiose non sono da considerarsi rappresentative.
Analoghe osservazioni valgono per le tartarughe.

Indecidibilità del FOL

Vi viene richiesto da un cliente di realizzare in dimostratore di teoremi per un linguaggio di R.C. dello stesso potere espressivo della logica del primo ordine con il requisito che il sistema deduttivo sia corretto, completo e che per ogni formula sia in grado di dire sempre se questa è un teorema oppure no. Che cosa fate:

a. accettate senza esitazione sicuri di poter svolgere questo incarico senza problemi;
b. accettate precisando meglio quello che realizzerete;
c. accettate senza esitazione (da buon venditore) sapendo che sarà impossibile soddisfare i requisiti ma fiduciosi che il cliente non se ne accorgerà;
d. cercate di convincere il cliente che la cosa è impossibile;
e. cercate di vendere al cliente un interprete PROLOG.

Senz'altro sbagliata è la risposta a). Le altre vanno bene se motivate correttamente.

Metodo di risoluzione

Unificatore più generale

Trovare l'unificatore più generale tra le seguenti coppie di espressioni o dire perché non sono unificabili..

Ricordiamo le regole:

1. f(s1, … , sn) = f(t1, … , tn) $\rightarrow$ s1= t1, … , sn = tn
2. f(s1, … , sn) = g(t1, … , tm) $\rightarrow$ fail se f$\neq$g
3. x = x $\rightarrow$ cancella
4. t = x $\rightarrow$ x = t
5. x = t, t $\neq$ x, x non occorre in t $\rightarrow$ applica [t/x] a tutte le altre equazioni
6. x = t, t $\neq$ x, x occorre in t $\rightarrow$ fail (*occur check*)

Nota: come caso particolare della 2, quando n=0, si fallisce su due costanti diverse

a) P(a, b, b) , P(x, y, z)

Passo 0: P(a, b, b) , P(x, y, z)

Passo 1: a=x (regola 1)
b=y
b=z

Passi 2,3,4: x=a (regola 4, tre volte)
y=b
z=b

mgu = {a/x, b/y, b/z}

b) Q(y, g(a, b)), Q(g(x, x),y)

Passo 0: Q(y, g(a, b))=Q(g(x, x),y)

Passo 1: y=g(x,x) (regola 1)
g(a,b)=y

Passo 2: y=g(x,x)
g(a,b)=g(x,x) (regola 5)

Passo 3: y=g(x,x)
a=x (regola 1)
b=x

Passo 4: y=g(x,x)
x=a (regola 4)
b=x

Passo 5: y=g(x,x)
x=a (regola 5)
b=a

Passo 6: FAIL per la regola 2

Le due espressioni non sono unificabili.

c) Older(father(y),y), Older(father(x), john)

Passo 0: Older(father(y),y)=Older(father(x), john)

Passo 1: father(y)=father(x) (regola 1)
y=john

Passo 2: y=x (regola 1)
y=john

Passo 3: john=x (regola 5)
y=john

Passo 3: x=john (regola 4)
y=john

mgu = {john/x, john/y}

d) Knows(father(y), y), Knows(x, x)

Passo 0: Knows(father(y), y)=Knows(x, x)

Passo 1: father(y)=x (regola 1)
y=x

Passo 2: x=father(y) (regola 4)
y=x

Passo 3: x=father(x) (regola 5)
y=x

Passo 4: FAIL per la regola 6 (fallisce l'*occur check*)

Le due espressioni non sono unificabili.

"Mary ama John" con il metodo di risoluzione

Dimostrare con il metodo di risoluzione che "Mary ama John" è conseguenza logica delle premesse: "Tutti amano chi ama qualcuno" e "John ama Mary".

Possiamo scriverlo in FOL nel modo seguente:

$\forall x \; \forall y \; (\exists z \; Ama(y, z)) \Rightarrow Ama(x, y)$
Ama(john, mary)

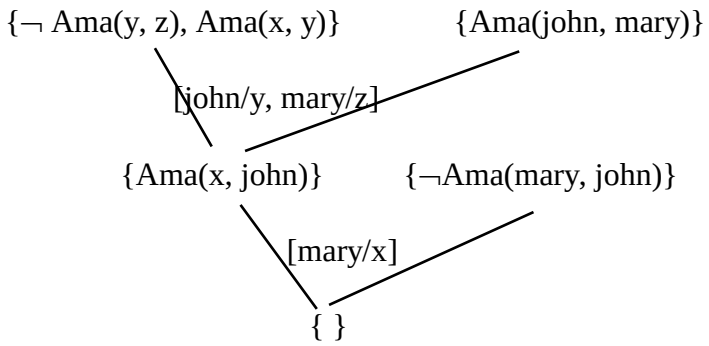
In forma a clausole abbiamo:

$\forall x \; \forall y \; \neg(\exists z \; Ama(y, z)) \vee Ama(x, y)$
$\forall x \; \forall y \; \forall z \; \neg Ama(y, z) \vee Ama(x, y)$
$\{\neg Ama(y, z), Ama(x, y)\}$
{Ama(john, mary)}

Il goal negato è

$\{\neg Ama(mary, john)\}$

Usando la refutazione abbiamo:



Metodo di risoluzione e strategie

Jack possiede un cane. Tutti i proprietari di cani sono amanti degli animali. Nessun amante degli animali uccide animali. Jack o la curiosità hanno ucciso la gatta Tuna. È stata la curiosità ad uccidere Tuna?

1. Formalizzare in FOL.
2. Trasformare in forma a clausole.
3. Mostrare che il fatto "Tuna è stata uccisa dalla curiosità" è conseguenza logica dei fatti asseriti con il metodo di risoluzione.
4. Che tipo di strategia di risoluzione avete adottato: unitaria? da input? lineare da input? all'indietro dal goal?

1. Formalizziamo in FOL

A) $\exists$ x Cane(x) $\wedge$ Possiede(Jack, x)
B) $\forall$x($\exists$ y Cane(y) $\wedge$ Possiede(x, y)) $\Rightarrow$ AA(x)
C) $\forall$x $\forall$y AA(x) $\wedge$ Animale(y) $\Rightarrow$ $\neg$ Uccide(x, y)
D) Uccide(Jack, Tuna) $\vee$ Uccide (Curiosità, Tuna)
E) Gatta(Tuna)
F) $\forall$x Gatta(x) $\Rightarrow$ Animale(x)

2. Trasformiamo in forma a clausole (una clausola è una disgiunzione di letterali)

A1) {Cane(D)}
A2) {Possiede(Jack, D)}

B) $\forall$x ($\exists$ y Cane(y) $\wedge$ Possiede(x, y)) $\Rightarrow$ AA(x)
$\forall$x $\neg$ ($\exists$ y Cane(y) $\wedge$ Possiede(x, y)) $\vee$ AA(x)
$\forall$x $\forall$y $\neg$ Cane(y) $\vee$ $\neg$ Possiede(x, y) $\vee$ AA(x)
$\neg$ Cane(y) $\vee$ $\neg$ Possiede(x, y) $\vee$ AA(x)
{$\neg$ Cane(y1), $\neg$ Possiede(x1,y1), AA(x1)}

C) $\forall$x $\forall$y AA(x) $\wedge$ Animale(y) $\Rightarrow$ $\neg$ Uccide(x,y)
$\forall$x $\forall$y $\neg$ AA(x) $\vee$ $\neg$ Animale(y) $\vee$ $\neg$ Uccide(x,y)
{$\neg$ AA(x2), $\neg$ Animale(y2), $\neg$ Uccide(x2,y2) }

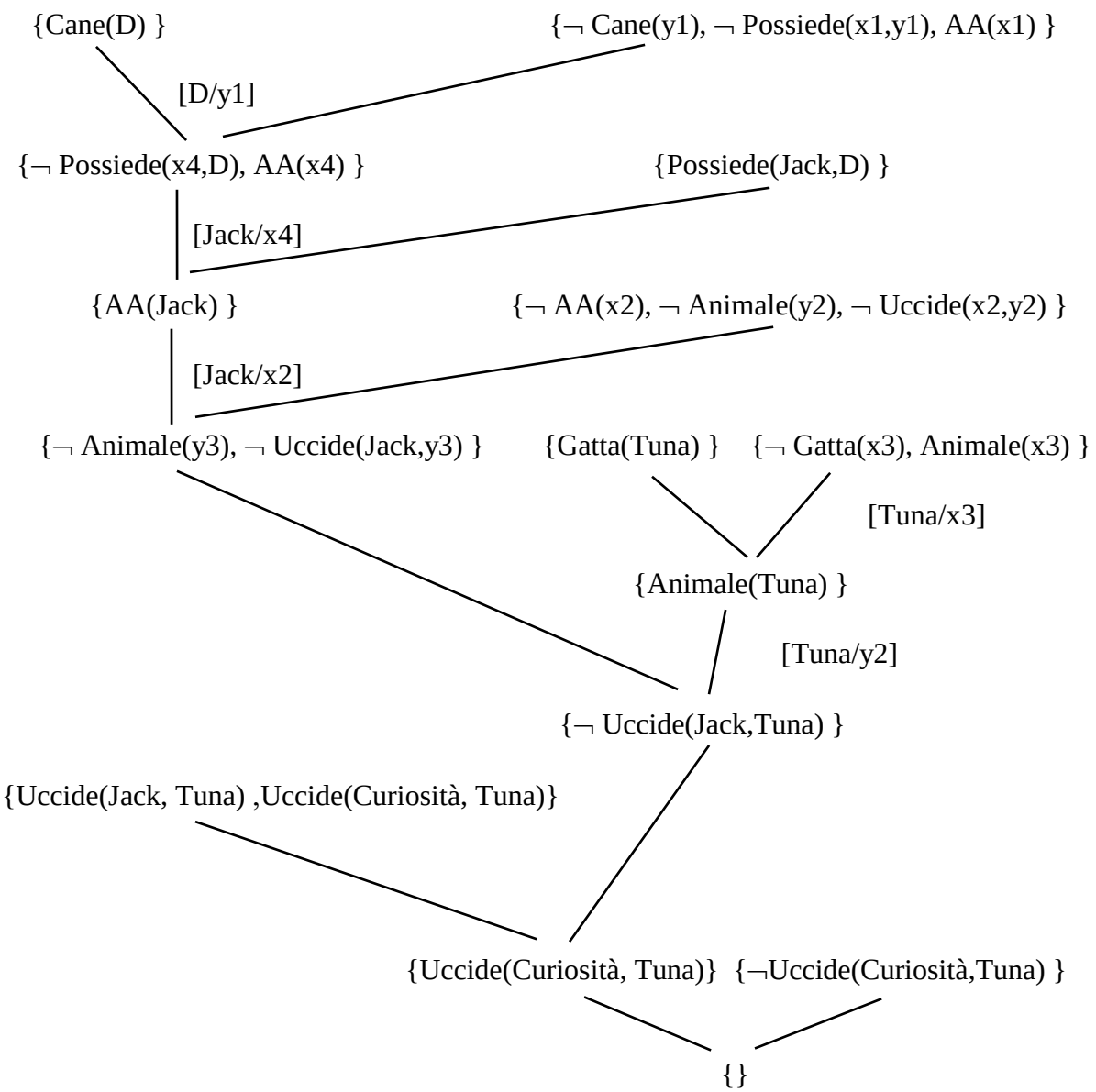
D) {Uccide(Jack, Tuna) ,Uccide(Curiosità, Tuna)}

E) {Gatta(Tuna) }

F) ¬ Gatta(x) ∨ Animale(x)
{¬ Gatta(x3), Animale(x3)}

G) {¬Uccide(Curiosità,Tuna)}

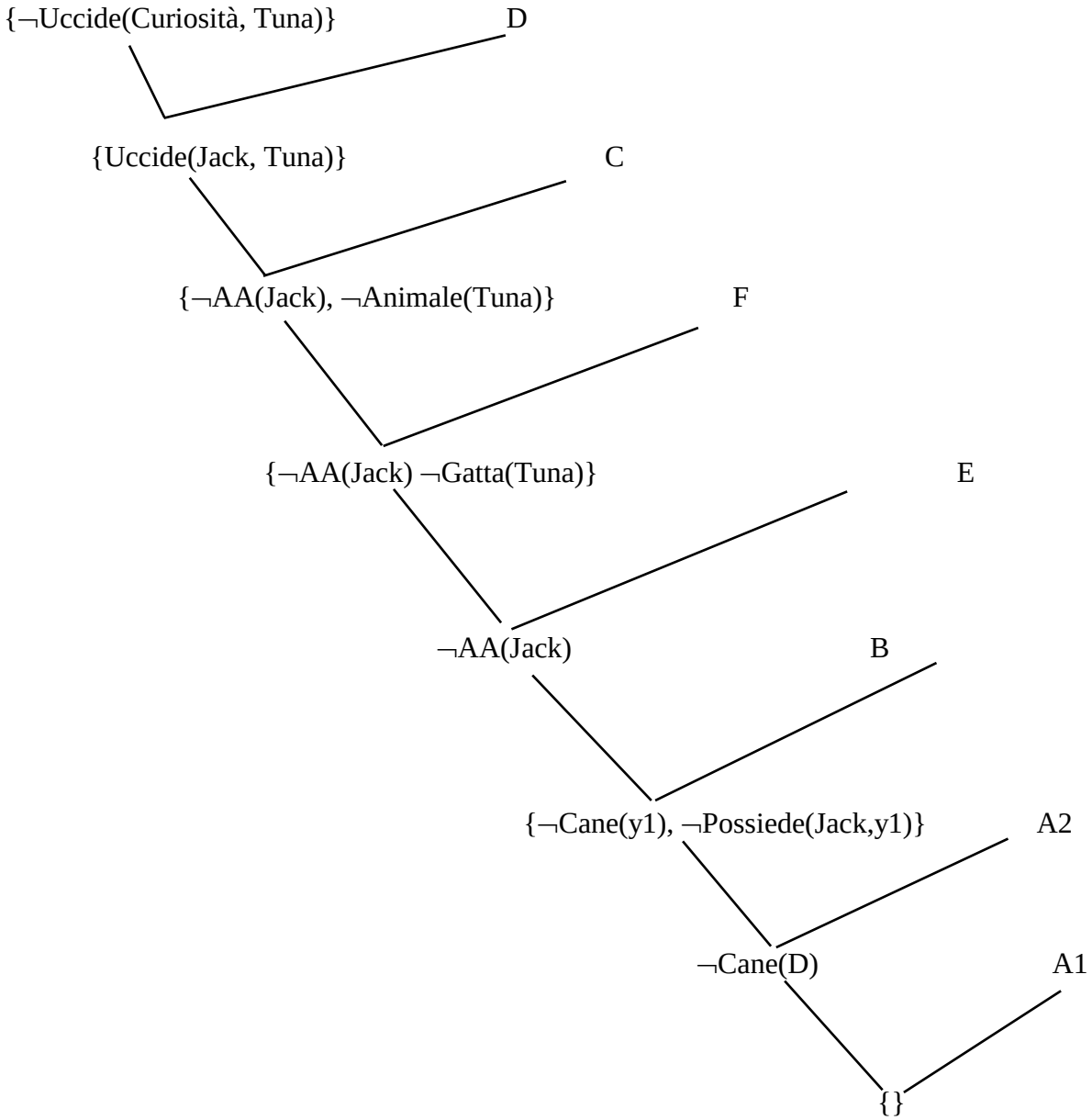
Grafo di risoluzione



Che tipo di strategia abbiamo adottato?

Unitaria? SI
Da input? NO Anche se avremmo potuto.
Lineare da Input? NO Anche se avremmo potuto.
All'indietro dal goal? NO

Esempio di risoluzione all'indietro dal Goal:

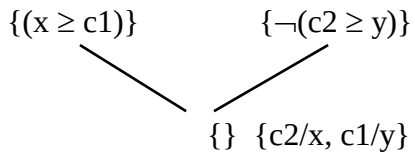


Conseguenza logica e risoluzione

Siano A e B definite come segue
A. $\forall x \exists y (x \geq y)$
B. $\exists y \forall x (x \geq y)$

Dimostrare che B |= A.

Possiamo dimostrarlo con il metodo di risoluzione. Essendo questo un metodo corretto se FC(B) $|-_{res}$ FC(A) allora sicuramente B |= A.
Portiamo B in forma a clausole $\{(x \geq c1)\}$
Portiamo ¬A in forma a clausole $\{\neg(c2 \geq y)\}$



Si noti cosa succede se proviamo a dimostrare che A |- B per risoluzione

Portiamo A in forma a clausole {(x ≥ f(x))}
Portiamo ¬B in forma a clausole {¬(g(y) ≥ y)}

{(x ≥ f(x))} {¬(g(y) ≥ y)}

non riesco a unificare perchè fallisce l'occur check

Programmazione Logica

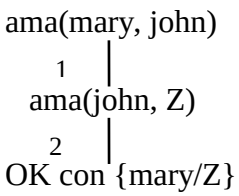
"Mary ama John" in programmazione logica

Esprimere in programmazione logica "Tutti amano chi ama qualcuno" e "John ama Mary" e chiedere se è conseguenza logica che "Mary ama John".

Programma Logico:
ama(X, Y) := ama(Y, Z)
ama(john, mary)

Goal
:= ama(mary, john)

In questo caso dobbiamo usare la risoluzione SLD, quindi iniziando dal goal costruiamo l'albero SLD:



Dal linguaggio naturale alle programmazione logica attraverso il FOL

Formalizzare le seguenti affermazioni in FOL e poi in programmazione logica.

a. I cavalli sono mammiferi
b. Il figlio di un cavallo è un cavallo
c. Barbablù (B) è un cavallo
d. B è il padre di Charlie (C)
e. Genitore e figlio sono relazioni inverse
f. Ogni mammifero ha un genitore

Formalizziamo in FOL e in forma a clausole

a. ∀x cavallo(x) ⇒ mammifero(x)
a1. {¬ cavallo(X1), mammifero(X1)}

b. ∀x ∀y cavallo(x) ∧ figlio(y, x) ⇒ cavallo(y)
b1. {¬ cavallo(X2), ¬figlio(Y2, X2) , cavallo(Y2)}

c. {cavallo(b)}

d. {genitore(b, c)}

e. $\forall x\ \forall y$ genitore(x, y) $\Rightarrow$ figlio(y, x)
e1. {$\neg$genitore(X3, Y3), figlio(Y3, X3)}

f. $\forall x$ mammifero(x) $\Rightarrow \exists$ y genitore(y, x)
f1. {$\neg$mammifero(X4), genitore(f(X4), X4)}

Possiamo riscrivere le clausole nella notazione della programmazione logica. Infatti notiamo che le clausole hanno un solo letterale positivo.

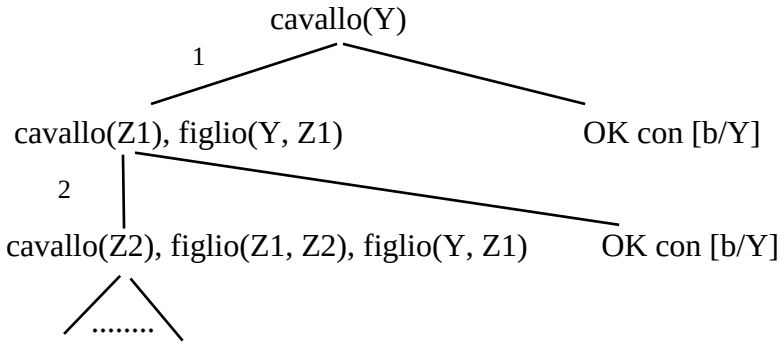
1. mammifero(X1) := cavallo(X1)
2. cavallo(X2) := cavallo(Y2), figlio(X2, Y2)
3. cavallo(b)
4. genitore(b, c)
5. figlio(X3, Y3) := genitore(Y3, X3)
6. genitore(f(X4), Y4) := mammifero(X4)

Strategia SLD e Prolog

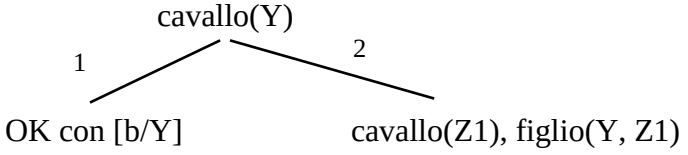
Con riferimento al programma logico dell'esercizio precedente dire come l'interprete PROLOG si comporta davanti alla domanda "Esistono cavalli?". Se ci sono problemi aggiustare il programma logico in modo che risponda correttamente alla domanda.

La domanda viene formalizzata come:
-? cavallo(Y)

Visita in profondità dell'albero SLD:



La computazione non termina perché la strategia sceglie sempre la stessa regola ricorsiva. Se scambio l'ordine delle clausole risolvo il problema. Infatti se sposto la 3 e 4 prima della 1 e 2 ho che:



Studenti e insegnanti

Sapendo che:

T*utti gli studenti sono in grado di risolvere alcuni problemi e non sono in grado di risolvere alcuni problemi.*

Alcuni insegnanti sono in grado di risolvere tutti i problemi.

a. Dimostrare con il metodo di risoluzione che:
   Alcuni insegnanti non sono studenti.
b. Scrivere il programma logico corrispondente ai primi due fatti o spiegare perché non si può.

Soluzione:

a1. Formalizzazione:

1. $\forall$x Stud(x) $\Rightarrow \exists$ y Solve(x, y) $\wedge \exists$ z $\neg$Solve(x, z)
2. $\exists$ x Ins(x) $\wedge \forall$y Solve(x, y)
3. Da dimostrare: $\exists$ x Ins(x) $\wedge \neg$ Stud(x)

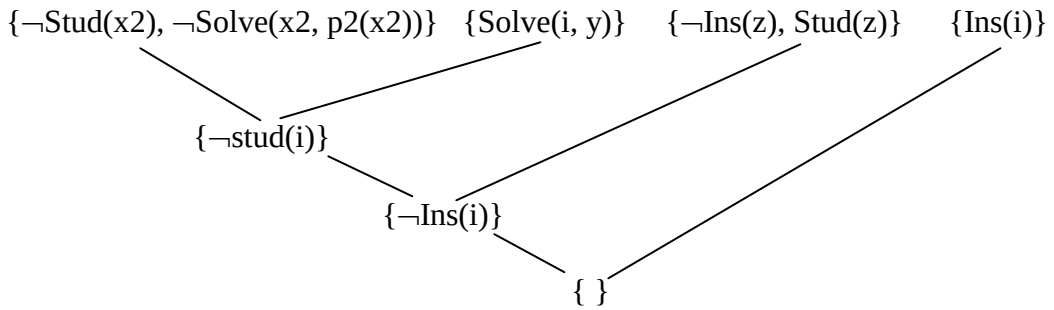
a2. Trasformazione in forma a clausole:

1. $\forall$x Stud(x) $\Rightarrow \exists$ y Solve(x, y) $\wedge \exists$ z $\neg$Solve(x, z)
$\forall$x $\neg$Stud(x) $\vee$ ($\exists$ y Solve(x, y) $\wedge \exists$ z $\neg$Solve(x, z)) [*eliminazione* $\Rightarrow$]
$\forall$x $\neg$Stud(x) $\vee$ (Solve(x, p1(x)) $\wedge$ $\neg$Solve(x, p2(x)))
[*skolemizzazione*]
$\neg$Stud(x) $\vee$ (Solve(x, p1(x)) $\wedge$ $\neg$Solve(x, p2(x))) [*eliminazione* $\forall$]
($\neg$Stud(x) $\vee$ Solve(x, p1(x))) $\wedge$ ($\neg$Stud(x) $\vee$ $\neg$Solve(x, p2(x)))
1.1 {$\neg$Stud(x1), Solve(x1, p1(x1))}
1.2 {$\neg$Stud(x2), $\neg$Solve(x2, p2(x2))}

2. $\exists$ x Ins(x) $\wedge \forall$y Solve(x, y)
Ins(i) $\wedge \forall$y Solve(i, y)
[*skolemizzazione*]
{Ins(i)}
{ Solve(i, y)}

Goal negato: $\neg\exists$ x Ins(x) $\wedge \neg$ Stud(x)
$\forall$ x $\neg$Ins(x) $\vee$ Stud(x)
{$\neg$Ins(z), Stud(z)}

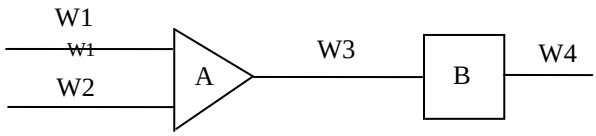
a3. Dimostrazione per refutazione:



b. Non è possibile rendere come programma logico la KB iniziale in quanto la prima formula, trasformata in forma a clausole, non è una clausola Horn **definita** (non ci sono letterali positivi).

Analisi di circuiti

Il seguente circuito ha 4 fili, W1, W2, W3 e W4. Ha una porta AND, la porta A, e un invertitore, la porta B. I fili di ingresso, W1 e W2 possono essere o non essere *on*. Se la porta A funziona correttamente, il filo W3 è *on* se e solo se i fili W1 e W2 sono entrambi *on*. Se l'invertitore B funziona correttamente, il filo W4 è *on* se e solo se il filo W3 non è *on*.



1. Si usino espressioni del tipo Ok(A), On(W1), ¬On(W3) per descrivere nella logica dei predicati il funzionamento del circuito rappresentato in figura.
2. Assumendo che gli ingressi W1 e W2 siano *on*, e che l'uscita W4 sia anch'essa *on*, si usi il metodo di risoluzione per dimostrare che la porta A oppure l'invertitore B non funzionano correttamente. Si usi una strategia *lineare da input* o, se impossibile, *lineare*.

1. $Ok(A) \Rightarrow (On(W3) \Leftrightarrow (On(W1) \wedge On(W2))$
2. $Ok(B) \Rightarrow (On(W4) \Leftrightarrow \neg On(W3))$

1.1 $Ok(A) \Rightarrow (On(W3) \Rightarrow (On(W1) \wedge On(W2))$
1.1.1 $\neg Ok(A) \vee \neg On(W3) \vee On(W1)$
1.1.2 $\neg Ok(A) \vee \neg On(W3) \vee On(W2)$
1.2 $Ok(A) \Rightarrow (On(W1) \wedge On(W2)) \Rightarrow On(W3))$
1.2.1 $\neg Ok(A) \vee \neg On(W1) \vee \neg On(W2) \vee On(W3)$
1.3 $Ok(B) \Rightarrow (On(W4) \Rightarrow \neg On(W3))$
1.3.1 $\neg Ok(B) \vee \neg On(W4) \vee \neg On(W3)$
1.4 $Ok(B) \Rightarrow (\neg On(W3) \Rightarrow On(W4))$
1.4.1 $\neg Ok(B) \vee On(W3) \vee On(W4)$

1. $\{\neg Ok(A), \neg On(W3), On(W1)\}$
2. $\{\neg Ok(A), \neg On(W3), On(W2)\}$
3. $\{\neg Ok(A), \neg On(W1), \neg On(W2), On(W3)\}$
4. $\{\neg Ok(B), \neg On(W4), \neg On(W3)\}$
5. $\{\neg Ok(B), On(W3), On(W4)\}$
6. $\{On(W1)\}$
7. $\{On(W2)\}$
8. $\{On(W4)\}$
9. $\{Ok(A)\}$
10. $\{Ok(B)\}$

$\{\neg On(W1), \neg On(W2), On(W3)\}$ [da 3 e 9]
$\{\neg On(W2), On(W3)\}$ [con 6]
$\{On(W3)\}$ [con 7]

{¬Ok(B), ¬On(W4)} [con 4]
{¬On(W4)} [con 10]
{ } [con 8]

Programmi logici e basi di dati

Vogliamo codificare in uno dei formalismi di R.C. visti, un catalogo di auto usate, cioè un elenco del tipo:

Marca	Modello	Anno	Prezzo
Fiat	punto	1992	8.000
VW	Golf	1993	12.000
Ford	Fiesta	1993	10.000

Vogliamo rispondere a domande del tipo:
a) Quanto vale la mia macchina? (Ad es. una Fiat punto del 92)
b) Quanto costa una Fiesta del '93?
c) Cosa posso comprare spendendo esattamente 8.000?
d) Cosa posso comprare avendo a disposizione 12000?

1) Codificare con un sistema a regole all'indietro tipo Prolog e dire come formulare le query.

I fatti:
auto(fiat, punto, 92, 8000).
auto(vw, golf, 93, 12000).
auto(ford, fiesta, 93, 10000).

Query:
a) auto(fiat, punto, 92, X).
b) auto(ford, fiesta, 93, X).
c) auto(X, Y, Z, 8000).
d) auto(X,Y, Z, P), P <= 12000.

Voli arei

Supponiamo di voler costruire un programma Prolog con conoscenza sui voli con l'intenzione di volere porre domande del tipo: "C'è un volo diretto da A a B?", "Posso volare da C a D?", "Quali destinazioni posso raggiungere da E?".

Supponiamo siano definiti i seguenti fatti:

direct(pisa, roma).
direct(roma, newyork).
direct(roma, amsterdam).
direct(newyork, anchorage).

Definiamo le connessioni come segue:

connection(X, Y) :- direct(X, Y).

connection(X, Y) :- direct(X, Z), connection(Z, Y).

a. Costruire l'albero SLD per la query
?- connection(pisa, anchorage).

b. Dire che cosa succede se aggiungo al programma come primo fatto:
direct(newyork, newyork).
e pongo la query
?- connection(newyork, X).

c. Dire che cosa succede con la stessa query se inverto l'ordine delle regole nel programma del punto b).

Adamo e i suoi figli (programmazione logica)

Date la seguente KB di formule del FOL:

$\forall x$ Uomo(x) $\Rightarrow$ Uomo(figlio(x))
Uomo(figlio(Adamo))

scrivere il programma logico corrispondente alla KB e verificare se le seguenti formule sono deducibili dal programma, tramite un interprete Prolog:

1. uomo(adamo)
2. uomo(figlio(adamo)
3. Z uomo(Z)

Nota: si richiede che le formule siano convertite in query Prolog opportune, di tracciare il comportamento dell'interprete Prolog e di dire in quali casi l'interprete restituisce la risposta corretta e in quali no.

Programma logico:

```
uomo(figlio(X)) :- uomo(X).
uomo(figlio(adamo)).
```

1. :- uomo(adamo).
   NO

La risposta è corretta e infatti uomo(adamo) non è conseguenza logica della KB.

2. :- uomo(figlio(adamo))
   :- uomo(adamo)
   fail
   :-uomo(figlio(adamo))
   YES

La risposta è corretta e infatti uomo(figlio(adamo)) è conseguenza logica della KB.

3. :- uomo(Z) [figlio(X1)/Z]
   :- uomo(Z1) [figlio(X2)/Z1]
   :- uomo(Z2) .... [figlio(X3)/Z2]

Il programma non termina e non risponde YES come avrebbe dovuto essendo che $\exists$ Z uomo(Z) è conseguenza logica della KB.

Analisi di circuiti come programma logico

Utilizzando il seguente vocabolario:

In(x, y) : x è in input alla componente y
Out(x, y): x è l'output della componente y
On(x): il filo x è *on*
Off(x): il filo x non è *on*
And(x): x è un componente di tipo And
Inv(x): x è un invertitore

a. scrivere un programma logico che descrive il comportamento normale (senza i problemi di malfunzionamento) della generica porta AND e del generico invertitore più l'insieme di fatti necessari a descrivere il circuito dell'esercizio precedente
b. Dire come si potrebbero formulare le seguenti interrogazioni:
   - W4 è *on*?
   - Esistono porte con fili di input *off*?
c. Spiegare che problemi ci sarebbero utilizzando ¬*On* al posto di *Off*.

Soluzione

a)

```
On(Y) :- And(A), In(X1, A), In(X2, A), Out(Y, A), On(X1), On(X2)
Off(Y) :- And(A), In(X, A), Out(Y, A), Off(X)
On(Y) :- Inv(B), In(X, B),  Out(Y, B), Off(X)
Off(Y) :- Inv(B), In(X, B), Out(Y, B), On(X)
And(a).
Inv(b).
In(w1, a).
In(w2, a).
Out(w3, a).
In(w3, b).
Out(w4, b).
```

b)

```
:- On(w4).
:- In(X, Y), Off(X)
```

Nota che il significato logico di :- On(w4) è ¬On(w4) (il goal negato) ovvero ¬On(w4) ∨ True, ovvero On(w4) => true.
Nel secondo caso il goal è ∃x ∃y (In(x, y), Off(x)); il goal negato ¬(∃x ∃y (In(x, y) ∧ Off(x))), ovvero ∀x ∀y ¬ (In(x, y) ∧ Off(x)), che poi scritto come regola diventa :- In(X, Y), Off(X).

c)

Nella programmazione logica classica non c'è modo di esprimere le regole necessarie con clausole Horn definite. Il seguente è un programma Prolog che usa la negazione come fallimento e che raggiunge lo scopo desiderato. Invece di chiedere se Off(X) bisognerà ovviamente chiedere se è ¬On(X). In tal caso il programma restituirà true in caso di fallimento del goal On(X)

```
On(Y) :- And(A), In(X1, A), In(X2, A), Out(Y, A), On(X1), On(X2).
On(Y) :- Inv(B), In(X, B),  Out(Y, B), ¬On(X).
```

Sistemi a Regole

Missionari e cannibali come sistema a regole

Riprendiamo il problema dei missionari e dei cannibali. Formalizzare le mosse come regole di produzione.

Indichiamo con m i missionari, con c i cannibali e con b la barca.
Quindi la terna (m,c,b) denota uno dei possibili stati formalizzati nell'esercizio

(m,c,b) & (m>1) & non-mangia(m-1,c,b*) $\Rightarrow$ (m-1,c,b*)
(m,c,b) & (c>1) & non-mangia(m,c-1,b*) $\Rightarrow$ (m,c-1,b*)
(m,c,b) & (m>2) & non-mangia(m-2,c,b*) $\Rightarrow$ (m-2,c,b*)
(m,c,b) & (c>2) & non-mangia(m,c-2,b*) $\Rightarrow$ (m,c-2,b*)
(m,c,b) & (m>1) & (c>1) & non-mangia(m-1,c-1,b*) $\Rightarrow$ (m-1,c-1,b*)

dove b*=0 se b=1 e b*=1 se b=0
e non-mangia(m,c,b) sse (m >= c or m=0) & ((3-m) >= (3-c) or (3-m) =0)

Termostato come sistema a regole

Scrivere, mediante un sistema di produzioni, il programma agente di un termostato che controlla una caldaia da riscaldamento. Le percezioni che il termostato riceve dall'ambiente sono:

- Valore della temperatura da mantenere (*temperatura di riferimento*)
- Temperatura dell'ambiente

Il termostato ha un suo stato interno in cui tiene traccia dello stato della caldaia (accesa o spenta) e della temperatura di riferimento. Il termostato può compiere le seguenti azioni:

- Accendere la caldaia
- Spegnere la caldaia

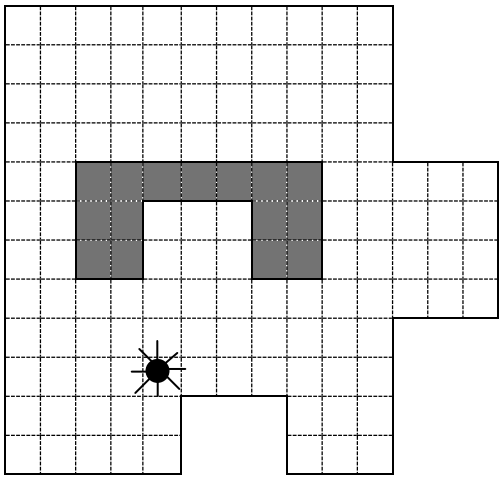
Si vuole che l'agente termostato accenda la caldaia se la temperatura ambiente scende di almeno 5 gradi al di sotto della temperatura di riferimento e che spenga la caldaia se la temperatura ambiente supera di almeno 5 gradi la temperatura di riferimento; altrimenti non fa niente.

Il robot che segue i contorni

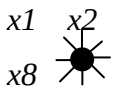
Un robot abita in un mondo a griglia, con muri tutto intorno, e in cui possono esistere oggetti poligonali. Il robot percepisce se le 8 celle che lo circondano sono libere o no. Le azioni che può compiere sono: spostarsi di una casella verso nord (N), verso sud (S), verso est (E) o verso ovest (O). Se il robot si sposta in una cella non libera (occupata da un muro o da un oggetto), l'azione non ha effetto. Inoltre, per semplicità, in questo mondo non esistono strettoie, cioè passaggi larghi una cella (vedi figura per un esempio di scenario). Il comportamento che vogliamo per il robot è il seguente: a partire da una cella qualsiasi deve portarsi in una cella adiacente a un muro o a un oggetto e da lì iniziare a seguire il muro o il confine dell'oggetto senza più discostarsene.

a. Si implementi il comportamento del robot con un sistema a regole
b. Si dica se si tratta di un agente:

☐ Reattivo
☐ Con stato
☐ Con obiettivo



Soluzione. *Assumendo di codificare le percezioni come in figura*



con xi $\in$ *{0, 1}, le regole per il comportamento richiesto potrebbero essere le seguenti:*

$(x2=1) \wedge (x4=0) \Rightarrow E$
$(x1=1) \wedge (x2=0) \Rightarrow N$
$(x8=1) \wedge (x2=0) \Rightarrow N$
$(x4=1) \wedge (x6=0) \Rightarrow S$
$(x3=1) \wedge (x4=0) \Rightarrow E$
$(x7=1) \wedge (x8=0) \Rightarrow W$
$(x6=1) \wedge (x8=0) \Rightarrow W$
$(x5=1) \wedge (x6=0) \Rightarrow S$
$T \Rightarrow N$

In caso di conflitto si sceglie la prima regola applicabile. La condizione T è la condizione sempre vera.

Una soluzione più compatta, ma che presuppone un linguaggio dei pattern più espressivo, è la seguente:

$((x2=1) \vee (x3=1)) \wedge \neg((x4=1) \vee (x5=1)) \Rightarrow E$
$((x4=1) \vee (x5=1)) \wedge \neg((x6=1) \vee (x7=1)) \Rightarrow S$
$((x6=1) \vee (x7=1)) \wedge \neg((x8=1) \vee (x1=1)) \Rightarrow W$
$((x8=1) \vee (x1=1)) \wedge \neg((x2=1) \vee (x3=1)) \Rightarrow N$
$T \Rightarrow N$

In entrambe le soluzioni l'agente è reattivo, in quanto decide l'azione unicamente basandosi sulle sue percezioni.

La formica e il feromone

Una formica artificiale vive in un mondo a griglia a due dimensioni ed è capace di seguire una scia di feromone (un percorso di caselle consecutive largo una casella, che può anche curvare). La formica occupa una sola casella e può essere rivolta in sù, in giù, a destra o a sinistra. È capace delle seguenti azioni: muoversi avanti di una casella (m), girarsi a sinistra rimanendo nella stessa casella (s), girarsi a destra rimanendo nella stessa casella (d). La formica sente se nella casella immediatamente davanti a lei (nella direzione verso cui è rivolta) c'è traccia di feromone.

a) Assumendo che la formica parta da una situazione in cui percepisce la traccia di feromone, scrivere il programma agente della formica sotto forma di regole di produzione, in modo tale che la formica segua la scia di feromone fino alla fine, senza mai tornare indietro.
b) Dire se l'agente formica può essere realizzato come un agente completamento reattivo o un agente con stato o un agente con obiettivo e caratterizzare l'ambiente in cui la formica si muove.

Aspirapolvere

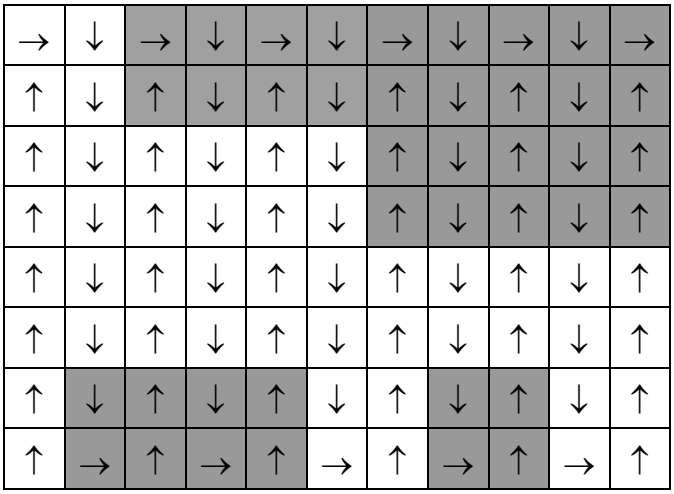
Sia dato un'agente aspirapolvere del tipo di quello visto a lezione, che opera in un mondo a griglia bidimensionale. Percepisce se la sua casella è sporca (dirt) e un bump se va a sbattere contro un muro (parete esterna della griglia). L'agente può spostarsi in avanti (F), girarsi a destra (R), a sinistra (L), aspirare lo sporco (S) e fermarsi (O). Il suo obiettivo è ripulire tutte le caselle della griglia. Nello stato iniziale l'agente si trova nella casella in basso a sinistra, rivolto verso l'alto:

3. Scrivere un insieme di regole che consentano all'agente di raggiungere il suo obiettivo e fermarsi.
4. Fornire un esempio di collocazione di mobili nella stanza che consenta all'agente di raggiungere comunque il suo obiettivo con le regole fornite.

In caso di conflitto scelgo la prima regola. All'inizio assumo ***north*** *nella Working Memory.*

dirth => S
north, ¬bump => F
north, bump => R, F, retract(north), retract(bump), assert(tr)
tr, bump => stop
tr, ¬bump => R, retract(tr), assert(south)
south, ¬bump => F
south, bump => L, F, retract(south), retract(bump), assert(tl)
tl, bump => stop
tl, ¬bump => L, retract(tl), assert(north)

Un esempio di collocazione di mobili che funziona con le regole precedenti è mostrata sotto. È sufficiente che i mobili sul lato nord inizino con colonne dispari (quelli sul lato sud con colonne pari) e siano larghi un numero pari di caselle.



La scuola del futuro

La nuova scuola Bishop adotta delle regole molto severe: se uno studente marina la scuola, viene espulso. La scuola vi ha commissionato un sistema esperto che deve determinare se e quali studenti debbano essere espulsi. Vi vengono date le seguenti specifiche : "*La scuola vi comunica quali studenti fanno assenze strategiche e quali studenti hanno voti bassi. La scuola inoltre vi mette a disposizione un telefono con il quale potete chiamare i genitori e verificare se sono a conoscenza delle assenze del figlio (per sapere se lo studente marina la scuola) raccomandandovi di non esagerare con le telefonate, cioè di chiamare solo se il figlio è sospettato*". Mentre andate via il direttore Gioia Biondetti aggiunge la raccomandazione: "*Se lo studente è molto ricco dimentichiamoci delle assenze strategiche e dei voti bassi!*".
Realizzate il sistema esperto come un sistema a regole, scrivendo le regole di produzione necessarie e specificando la strategia di risoluzione dei conflitti da adottare. Potete usare il seguente vocabolario:

AS: assenze strategiche
VB: voti bassi
M: marina la scuola
R: ricco

CHIAMA: azione di chiamare i genitori
ESPELLI: azione di espellere l'alunno

Assumiamo che:

- L'azione (CHIAMA x) abbia come effetto di asserire (M x) nella WM nel caso i genitori rispondano che sono inconsapevoli delle assenze.

Una possibile soluzione:

1. (AS ?x) (VB ?x) => (assert (SOSPETTO ?x))
2. (SOSPETTO ?x)(R ?x) => (retract (AS ?x)) (retract (VB ?x))
3. (SOSPETTO ?x) => (CHIAMA ?x)
4. (M ?x) => (ESPELLI ?x)

2 deve avere la precedenza su 3. Si può ad esempio usare una strategia di risoluzione dei conflitti che sceglie sempre la prima regola applicabile o la più specifica.

Un'altra soluzione che poteva essere considerata accettabile:

1. (AS ?x)(VB ?x)(not (R ?x)) => (assert (SOSPETTO ?x))
3. (SOSPETTO ?x) => (CHIAMA ?x)
4. (M ?x) => (ESPELLI ?x)

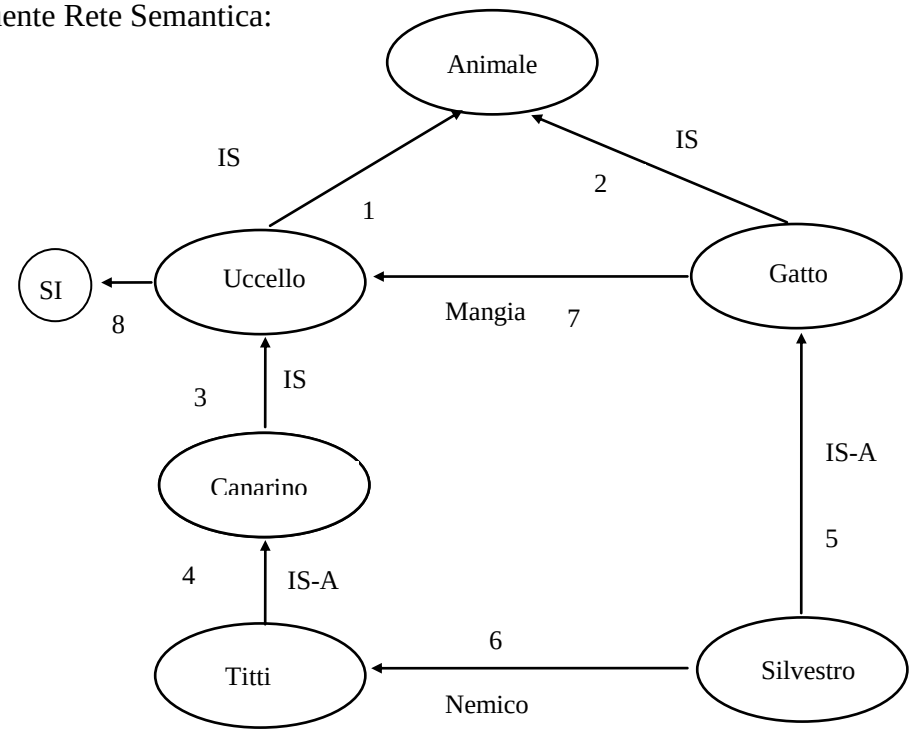
o anche la più essenziale:

1. (AS ?x)(VB ?x)(not (R ?x)) => (CHIAMA ?x)
4. (M ?x) => (ESPELLI ?x)

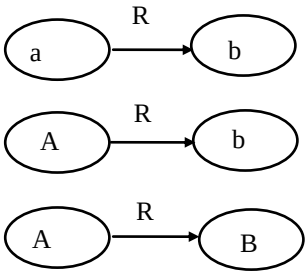
Reti semantiche

Reti Semantiche e FOL

Data la seguente Rete Semantica:



Darne una lettura ragionevole in FOL ricordando che:



e che le relazioni IS e IS-A hanno una lettura particolare (sottoinsieme e appartenenza). Con riferimento alla teoria FOL ottenuta, rispondere alle domande:
a) Titti vola?
b) Silvestro mangia Uccelli?
c) Silvestro mangia Titti?

Teoria FOL corrispondente alla rete semantica:
1. ∀x Uccello(x) ⇒ Animale(x)
2. ∀x Gatto(x) ⇒ Animale
3. ∀x Canarino(x) ⇒ Uccello(x)
4. Canarino(titti)
5. Gatto(silvestro)
6. Nemico(silvestro,titti)

7. $\forall x$ Gatto(x) $\Rightarrow \exists$ y Uccello(y) $\wedge$ Mangia(x,y)
8. $\forall x$ Uccello(x) $\Rightarrow$ Vola(x,si)

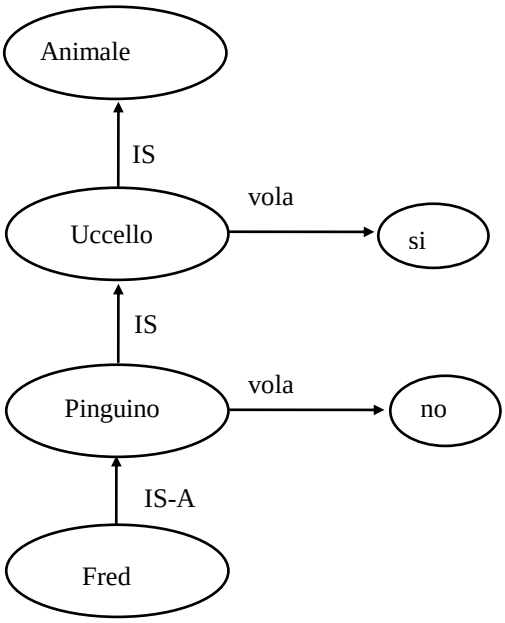
a) Titti vola? SI
   Canarino(titti), Uccello(titti), Vola(titti, SI)

b) Silvestro mangia Uccelli? SI
   Gatto(silvestro), $\exists$ y Uccello(y) $\wedge$ Mangia(silvestro,y)

c) Silvestro mangia Titti? Non si può dedurre!

Ereditarietà con Eccezioni

Data la seguente rete semantica:



a. Dire quale sarebbe la risposta alla domanda "Fred vola?", con il meccanismo deduttivo proprio delle reti semantiche.
b. Dire quale sarebbe la risposta nella teoria logica corrispondente.

Logiche Terminologiche

Padri felici e meno felici

Formalizzare con un linguaggio terminologico

Sono padri coloro che hanno un figlio. Sono padri felici i padri i cui figli sono bravi.
Sono *molto padri* coloro che hanno almeno quattro figli. Sono padri disperati i *molto padri* i cui figli non sono bravi. Inoltre sappiamo che Giorgio ha un solo figlio e questi è bravo.

MD
Padre $\Leftrightarrow$ (Some Figlio)
PadreFelice $\Leftrightarrow$ (and Padre (all Figlio Bravo))

MoltoPadre ⇔ (at least 4 Figlio)
PadreDisperato ⇔ (and MoltoPadre (all Figlio (not Bravo)))

MA
(at-most 1 Figlio) [Giorgio]
(c-some Figlio Bravo) [Giorgio]

KB = MD ∪ MA

Domande:

1) Giorgio è un padre felice?
Che problema decisionale è? È un caso di Instance Checking (IC).

KB |= (PadreFelice) [Giorgio]

Questo può essere ricondotto a soddisfacibilità della KB (KBS):

¬KBS (KB ∪ (not PadreFelice)[Giorgio])

KB = {(at-most 1 Figlio) [Giorgio],
(c-some Figlio Bravo) [Giorgio],
(not (and (some Figlio) (all Figlio Bravo)))[Giorgio]}

2) Un padre è una persona con al più un figlio?

Che problema decisionale è? Sussunzione ibrida (HSU).

HSU(KB,(atmost 1 Figlio), Padre)?

riduzione a KBS

¬ KBS (KB ∪ {(and Padre (not (atmost 1 Figlio)))[i]}

3) Un padre è una persona con almeno un figlio?

È ancora un problema di sussunzione ibrida:
HSU (KB, (atleast 1 Figlio), Padre)

Riconducibile a:

¬ KBS (KB ∪ {(and Padre (not (atleast 1 Figlio)))[i]}

4) Può un "molto padre" essere padre felice?

Che problema è? Di consistenza del concetto:

(and MoltoPadre PadreFelice)

Concetto espanso con le definizioni nella KB:

CS((and (atleast 4 Figlio) (and (some Figlio)(all Figlio Bravo)))

Riconducibile a:

¬ SU ((bottom), (and (atleast 4 Figlio) (and (some Figlio) (all Figlio Bravo))))

ovvero:

KBS ((and (and (atleast 4 Figlio)
(and (some Figlio) (all Figlio Bravo)))
(not (bottom))[i])

5) Una persona con 5 figli di cui uno non è bravo, è un padre disperato?

È un problema di sussunzione ibrida. Possiamo fare come prima e chiedere se:
(and (atleast 5 Figli) (almost 5 Figli) (c-some Figlio (not Bravo))) è sussunto da
(PadreDisperato)

oppure, alternativamente, possiamo aggiungere alla KB:

(and (atleast 5 Figlio) (atmost 5 Figlio) (c-some Figlio (not bravo))) [i]

e chiedere se

KB |= (PadreDisperato) [i]

ossia se

¬KBS (KB ∪ (not PadreDisperato) [i])

Monogami, poligami, coniugati e celibi

Definire con una logica terminologica i concetti di "Monogamo" (con esattamente una moglie), "Poligamo"(con più mogli), "Coniugato" (con moglie) e "Celibe" (senza moglie). Mostrare che il concetto di "Monogamo o Poligamo" è equivalente al concetto di "Coniugato" (che la sussunzione vale nelle due direzioni).

MD:

Monogamo ⇔ (and (some Moglie)(atmost 1 Moglie))
Poligamo ⇔ (at least 2 Moglie)
Coniugato ⇔ (some Moglie)
Celibe ⇔ not (Coniugato)

(or Poligamo Monogamo) equivale a Coniugato ?

Dobbiamo controllare la sussunzione nelle due direzioni:

a) HSU(KB, Coniugato, (or Poligamo Monogamo))

KB ∪ (or Poligamo Monogamo)[i] |= Coniugato[i]
¬KBS(KB ∪ (or Poligamo Monogamo)[i] ∪ (not Coniugato)[i])

b) HSU(KB, Coniugato, (or Poligamo Monogamo))

KB ∪ {Coniugato[i]} |= (or Poligamo Monogamo) [i]
¬KBS(KB ∪ Coniugato[i] ∪ (not (or Poligamo Monogamo))[i])

(not (or (atleast 2 Moglie) (and (some Moglie) (atmost 1 Moglie)))) [i]

Logiche terminologiche e calcolo dei predicati

Scrivere una espressione della logica dei predicati che caratterizza, mediante un predicato composto, l'insieme degli oggetti che sono nell'estensione della seguente espressione delle logiche terminologiche:

(**and** persona (**atleast** 2 figli) (**all** figli studenti)(**not (some** coniuge)))

Dire se la seguente descrizione sussume la precedente:

(**or** (**atleast** 1 figli) (**some** coniuge))

Incertezze elettorali

I votanti sono di destra, di sinistra o indecisi. I votanti di destra amano Berlusconi; i votanti di sinistra non amano Berlusconi. Montanelli è di destra ma non ama Berlusconi. Dimostare che Montanelli è un indeciso oppure non vota.

Formalizzare con una logica terminologica, aggiungendo eventualmente i fatti di senso comune che mancano, e impostare il problema come un problema decisionale delle logiche terminologiche.

Soddisfacibilità di una KB

Data la seguente KB, espressa in una logica terminologica:

Tenore => Uomo
Soprano => Donna
Donna <=> not Uomo

Soprano[Callas]

Dimostrare formalmente che Callas non può essere un tenore.

È un problema di Instance Checking. Vogliamo vedere se:

KB |= (not Tenore)[Callas]

Questo equivale a dimostrare che la KB con l'aggiunta di Tenore[Callas] diventa insoddisfacibile:

$\neg$KBS(KB $\cup$ Tenore[Callas])

Sostituendo le definizioni si ottiene:

Soprano[Callas]
(and C* Donna)[Callas]
(and C* (not Uomo))[Callas]

Tenore[Callas]
(and C+ Uomo)[Callas]

L'insieme di vincoli iniziale è pertanto:
θ_0={(and C* (not Uomo))[Callas],
(and C+ Uomo)[Callas]}

Applichiamo la regola $\rightarrow$and due volte e otteniamo:
θ_2={C*[Callas], (not Uomo))[Callas],
C+[Callas], Uomo)[Callas]}
che presenta un CLASH (Uomo[Callas], (not Uomo)[Callas]).

Quindi la KB è insoddisfacibile e possiamo concludere che Callas non è un tenore.

Come impostereste il problema di decidere se un soprano può essere un tenore?

Medici e pescivendoli

Data la seguente KB espressa nel linguaggio delle logiche terminologiche:
Medico => (AND laureato ...)
Pescivendolo => (AND (not laureato) ...)
dire come formulereste i seguenti problemi in termini di soddisfacibilità di una KB (KBS)
a. Un medico può essere pescivendolo?
b. Un pescivendolo può essere medico?
c. Uno stesso individuo può essere sia medico che pescivendolo?

Soluzione

Sono tutti e tre lo stesso problema e riconducibili al problema di vedere se il concetto (AND Medico Pescivendolo) è consistente, è cioè se l'intersezione dei Medici con i Pescivendoli è diversa dall'insieme vuoto.

Formalmente:

CS(AND Medico Pescivendolo)

Il problema può essere ricondotto alla soddisfacibilità di una KB in questo modo:

KBS(KB$\cup${(AND Medico Pescivendolo)[i]}

Cioè il problema di decidere se aggiungendo alla KB l'asserzione (AND Medico Pescivendolo)[i] (Nota: le asserzioni riguardano sempre individui) la KB è ancora soddisfacibile oppure no. Se l'intersezione fosse diversa dall'insieme vuoto infatti la KB sarebbe ancora soddisfacibile (se lo era).