# Esercitazione n. 7

## 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.

T-BOX

Padre ≡∃haFiglio.⊤

PadreFelice ≡ Padre *∏* ∀haFiglio.Bravo

MoltoPadre ≡ (≤ 4 haFiglio)

PadreDisperato ≡MoltoPadre *∏* ∀haFiglio.¬Bravo

A-BOX

Giorgio: (≥ 1 haFiglio)

Giorgio: ∃haFiglio.Bravo

KB = T-BOX A-BOX

Domande:

1) Giorgio è un padre felice?

Che problema decisionale è? È un caso di Instance Checking (IC).

KB |= Giorgio: PadreFelice

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

¬KBS (KB Giorgio: ¬PadreFelice})

KB =  Giorgio: (≥ 1 haFiglio),

Giorgio: ∃haFiglio.Bravo,

Giorgio: ¬(∃haFiglio.T *∏* ∀haFiglio.Bravo) }

Normalizzando l’ultima asserzione.

Giorgio: ¬∃haFiglio.T ⨆ ¬∀haFiglio.Bravo

Giorgio: ∀haFiglio.⊥ ⨆ ∃haFiglio.¬Bravo

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

Che problema decisionale è? Sussunzione.

(≤ 1 Figlio) sussume Padre ?

KB |= Padre ⊑ (≤ 1 Figlio)

riduzione a KBS

¬ KBS (KB {*a*: Padre *∏* ¬(≤ 1 haFiglio)

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

È ancora un problema di sussunzione ibrida:

(≥ 1 haFiglio) sussume Padre ?

KB |= Padre ⊑ (⩾ 1 Figlio)

Riconducibile a:

¬ KBS (KB {*a*: Padre *∏* ¬(≥ 1 haFiglio)

4) Può un “molto padre” essere padre felice?

Che problema è? Di consistenza del concetto:

MoltoPadre *∏* PadreFelice

Concetto espanso con le definizioni nella KB:

(≥ 4 Figlio) *∏* ∃haFiglio.T *∏* ∀haFiglio.Bravo

Riconducibile a:

KBS ({*a*: (≥ 4 Figlio) *∏* ∃haFiglio.T *∏* ∀haFiglio.Bravo})

 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:

KB |= (≥ 5 Figli) *∏* (≤ 5 Figli) *∏* (∃haFiglio (¬Bravo)) ⊑ (PadreDisperato)

oppure, alternativamente, possiamo aggiungere alla KB:

i:(≥ 5 Figli) *∏* (≤ 5 Figli) *∏* (∃haFiglio (¬Bravo))

e chiedere se

KB |= i:PadreDisperato

ossia se

¬KBS (KB ¬ PadreDisperato})

**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 ≡ (∃haMoglie.T) *∏* (≤ 1 haMoglie)

Poligamo ≡ (≥ 2 haMoglie)

Coniugato ≡ (∃haMoglie.T)

Celibe ≡ ¬Coniugato

(Poligamo U Monogamo) equivale a Coniugato ?

Dobbiamo controllare la sussunzione nelle due direzioni:

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

KB Poligamo U Monogamo) |= i:Coniugato

¬KBS(KB Poligamo U Monogamo)}  {i:¬Coniugato)})

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

KB {i:Coniugato[i]|= i:(Poligamo U Monogamo)

¬KBS(KB Coniugato}  {i:¬(Poligamo U Monogamo)})

**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:

G: (Persona *∏* ∀haFiglio.Studente *∏* ¬**(**∃haConiuge.T))

t((Persona *∏* ∀haFiglio.Studente *∏* ¬**(**∃haConiuge.T)), G)

t(Persona, G) ⋀ t(∀haFiglio Studente, G) ⋀ t(¬**(**∃haConiuge), G)

Persona(G) ⋀∀y haFiglio(G, y) => Studente(y) ⋀ ¬∃zhaConiuge(G, z)

**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 ≡¬Uomo

Callas: Soprano

Dimostrare formalmente che Callas non può essere un tenore.

È un problema di Instance Checking. Vogliamo vedere se:

KB |= Callas: ¬Tenore

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

¬KBS(KB {Callas:Tenore})

Sostituendo le definizioni si ottiene:

Callas: Soprano

Callas: C\* *∏* Donna

Callas: C\**∏* ¬Uomo

Callas: Tenore

Callas: C+ *∏* Uomo

L’insieme di vincoli iniziale è pertanto:

L(Callas)={ C\* *∏* ¬Uomo, C+ *∏* Uomo}

Applichiamo la regola →and due volte e otteniamo:

L(Callas)={ C\*, ¬Uomo, C+,Uomo}

che presenta un CLASH (Uomo, ¬Uomo)

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 => Laureato *∏* ...

Pescivendolo => ¬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 (Medico *∏* Pescivendolo) è consistente, è cioè se l’intersezione dei Medici con i Pescivendoli è diversa dall’insieme vuoto.

Formalmente:

CS(Medico *∏* Pescivendolo)

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

KBS(KB∪{i: Medico *∏* Pescivendolo})

Cioè il problema di decidere se aggiungendo alla KB l’asserzione i: Medico *∏* Pescivendolo (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).

Sia data la seguente base di conoscenza espressa nella logica descrittiva ALC.

K=<T, A> dove T={ } e A contiene le seguenti asserzioni:

(john, susan): Amico

(john, chris): Amico

(susan, chris): Ama

(chris, bill): Ama

susan: Femmina

bill: ¬Femmina

Il fatto che “*John ha per amico una femmina che ama un maschio* (*qualcuno ‘non femmina’*)” è conseguenza logica della base di conoscenza?

1. Si esprima il fatto come un’asserzione in ALC.
2. Si riconduca il problema a soddisfacibilità della base di conoscenza e si risolva formalmente applicando l’algoritmo per la soddisfacibilità per ALC.

Soluzione: Il problema si può formulare così:

K |= john: ∃Amico.(Femmina *∏* ∃Ama. ¬Femmina), un problema di Instance Checking equivalente a:

K ∪ { john: ¬(∃Amico.(Femmina *∏* ∃Ama. ¬Femmina)} insoddisfacibile

Normalizzando:

K ∪ { john: ∀Amico. ¬ (Femmina *∏* ∃Ama.¬Femmina)}

K ∪ { john: ∀Amico. (¬Femmina ⨆ ∀Ama. Femmina)}

Applicazione dell’algoritmo per la soddisfacibilità di ALC. All’inizio abbiamo:

(john, susan): Amico

(john, chris): Amico

(susan, chris): Ama

(chris, bill): Ama

susan: Femmina

bill: ¬Femmina

john: ∀Amico. (¬ Femmina ⨆ ∀Ama. Femmina)

John L={∀Amico. (¬Femmina ⨆ ∀Ama. Femmina)}

*amico*

*amico* Susan L={Femmina}

*ama*

Chris L={}

*ama*

Bill L={(¬Femmina}

John L={∀Amico. (¬Femmina ⨆ ∀Ama. Femmina)}

*amico*

*amico* Susan L={Femmina, ¬Femmina ⨆ ∀Ama. Femmina }

*ama*

Chris L={¬Femmina ⨆ ∀Ama. Femmina }

*ama*

Bill L={(¬Femmina}

4 casi che devono tutti dare CLASH.

Caso 1 e 2: Susan:¬Femmina

John L={∀Amico. (¬Femmina ⨆ ∀Ama. Femmina)}

*amico*

*amico* Susan L={Femmina, ¬Femmina} CLASH

*ama*

Chris L={¬Femmina ⨆ ∀Ama. Femmina }

*ama*

Bill L={(¬Femmina}

Caso 3: Susan ama solo femmine e Chris non Femmina

John L={∀Amico. (¬Femmina ⨆ ∀Ama. Femmina)}

*amico*

*amico* Susan L={Femmina, ∀Ama. Femmina }

*ama*

Chris L={¬Femmina, Femmina} CLASH

*ama*

Bill L={(¬Femmina}

Caso 3: Susan ama solo femmine e Chris ama solo femmine CLASH

John L={∀Amico. (¬Femmina ⨆ ∀Ama. Femmina)}

*amico*

*amico* Susan L={Femmina, ∀Ama. Femmina }

*ama*

Chris L={¬Femmina, ∀Ama. Femmina}

*ama*

Bill L={(¬Femmina, Femmina} CLASH

Visto che tutti i rami terminano con CLASH, K ∪ { john: ∀Amico. (¬ Femmina |⨆ ∀Ama. Femmina)} è insoddisfacibile e quindi abbiamo dimostrato

K |= john: ∃Amico.(Femmina ⨅ ∃Ama. ¬Femmina)