Lezione 20

Termina la 19 e la prima parte del corso
Poi preparazione del II compitino Domande
Giovedì 29 ancora esercizi
Giovedì 6 II compitino

2.1. Il completamento di Clark

- Idea: assiomi impliciti che corrispondano all'idea di conoscenza completa:
  - KB stabilisce esplicitamente le conoscenze positive
  - Vi sono assiomi lasciati impliciti per economia, che asseriscono la negazione di quanto non asserito in positivo; ciò equivale ad assumere che la conoscenza positiva è completa

ESEMPIO.

$p(2) \leftarrow$ not q.

Non ho asserito q; implicitamente asserisco not q.

Completamento 1: not q.

Da not q posso derivare p(2), ma nulla dico sui valori diversi da 2.

Completamento 2: $p(X) \leftrightarrow X=2 \wedge$ not q
(trasformo se in se e solo se)

Il completamento implicito Comp($p(2) \leftarrow$ not q) è:

$p(X) \leftrightarrow X=2 \wedge$ not q.
not q.

Una piccola trasformazione fa capire il contenuto negativo di solo-se:

$p(X) \leftarrow X=2 \wedge$ not q.
$p(X) \rightarrow X=2 \wedge$ not q.
not q.

Contrapposizione: $A \rightarrow B$ equivale a not $B \leftarrow$ not A

$p(X) \leftarrow X=2 \wedge$ not q.
not $p(X) \leftarrow$ not$(X=2 \wedge$ not q).
not q.

Ad esempio not p(3) segue da not 3=2 per UNA.
Ma quali assiomi corrispondono all'assunzione UNA?

CET = assiomatizzazione di UNA

- CET: Clark's Identity Theory
- La parte positiva di CET è data dalle regole ID1 e ID2 già viste.
- La parte negativa, sotto forma di regole, è data da:
  - inj(f) ***iniettività,*** per ogni funtore f
  - div(f,g) ***diversità,*** per ogni coppia di funtori e/o costanti f,g diverse
  - ock ***occurr check*** per ogni termine T(X)

inj(f): $\dfrac{\text{not}(\ t_k = t'_k\ )}{\text{not}\ (f(t_1,..,t_k,..,t_n) = f(t'_1,..,t'_k,..,t'_n))}$

div(f,g): $\dfrac{}{\text{not}\ (f(\ldots\ldots) = g(\ldots..))}$

ock $\dfrac{}{\text{not}(\ T(X) = X\ )}$

Esempio di prova in CET

$$\dfrac{\dfrac{\dfrac{}{\text{div}(0,s)}}{\text{not}(0 = s(X))}\,\text{inj}(f)}{\text{not}(\ f(Y,0) = f(Y,s(X))\ )}\,\text{inj}(s)$$
$$\text{not}(s(\ f(Y,0)) = s(f(Y,s(X)))\ )$$

Teorema: Due ***termini*** A,B unificano con unificatore σ sse
CET |- A σ = B σ
A,B non unificano sse
CET |- not(A = B)

NB: not(A=B) significa $\forall$x1,…,xn: not(A=B)

Algoritmo che associa ad ogni P il completamento di Clark Comp(P)

- Si dice che una clausola definisce p se la sua testa contiene p(….)
- Esempio: pari(X) ← sum(A,A,X) definisce pari
- I passi della costruzione di Comp(P) sono:
  1. Aggiungere CET (lo assumeremo come scontato)
  2. Normalizzare tutte le clausole che definiscono uno stesso predicato p in modo che abbiano la stessa testa, usando l'identità; se un predicato non è definito, si asserisce la sua negazione in forma aperta.
  3. Passare *se e solo se* dopo esistenzializzazione e disgiunzione

Passo 2

p(T) ← BODY diventa p(V) ← V=T ∧ BODY
V nuove variabili, le stesse in ogni clausola

Se BODY è vuoto, diventa p(V) ← V=T.
Se p non è definito, si inserisce not p(X).

Esempio:

sum(X,0,X).
sum(X,s(Y),s(Z)) ← sum(X,Y,Z).

diventa

sum(V1,V2,V3) ← V1=X ∧ V2=0 ∧ V3=X.
sum(V1,V2,V3) ← V1=X ∧ V2=s(Y) ∧ V3=s(Z) ∧ sum(X,Y,Z).

Passo 3. Esistenzializzazione, disgiunzione e "se e solo se"

p(V) ← V = T1 ∧ Body1.
…
p(V) ← V = Tk ∧ Bodyk.

Diventano la definizione completa di p:

p(V) ↔ ∃y1(V=T1 ∧ Body1) ∨ … ∨ ∃yk(V=Tk ∧ Bodyk).

ESEMPIO. La somma diventa:

sum(V1,V2,V3) ↔ ∃ x (V1=x ∧ V2=0 ∧ V3=x) ∨
∃ x,y,z (V1=x ∧ V2=s(y) ∧ V3=s(z) ∧ sum(x,y,z)).

Il significato di solo se come informazione negativa

p(X) → ∃y1(x=T1 ∧ Body1) ∨ … ∨ ∃yn(x=Tn ∧ Bodyn)

Per contrapposizione, De Morgan e leggi sui quantificatori:
not p(X) ← ∀y1 not (X=T1 ∧ Body1) ∧ … ∧ ∀yn not (X=Tn ∧ Bodyn)

$$\dfrac{\forall y1\ \text{not}\ (X=T1 \wedge \text{Body1}) \quad \ldots\ \forall yn\ \text{not}\ (X=Tn \wedge \text{Bodyn})}{\text{not}\ p(X)}$$

$$\dfrac{}{\forall y\ \text{not}\ (T=T1 \wedge \text{Body})}$$

Se T, T1 non unificano

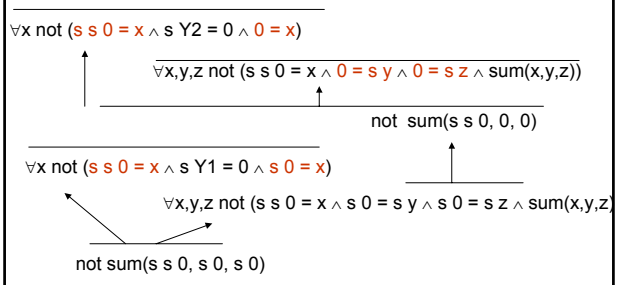
$$\dfrac{\text{not}(\text{Body}\sigma\ )}{\forall y\ \text{not}\ (T=T1 \wedge \text{Body})}$$

Se T ground e
T = T1σ

$$\dfrac{\text{not}(Ak\ )}{\text{not}\ (A1 \wedge\ ..\ \wedge\ An)}$$

Le regole qui date sono si possono dimostrare a partire dal completamento e corrispondono sostanzialmente alla negazione come fallimento finito senza floundering

not sum(V1,V2,V3) ← ∀ x not (V1=x ∧ V2=0 ∧ V3=x) ∧
∀ x,y,z not (V1=x ∧ V2=s(y) ∧ V3=s(z) ∧ sum(x,y,z)).



L'assunzione di conoscenza completa e il completamento di Clark

- In presenza di negazione, CWA può risultare inconsistente
- La situazione è migliore con il completamento di Clark, per cui si può ragionevolmente formalizzare l'assunzione di conoscenza completa come segue.
- Sia KB una base di conoscenza con clausole normali, e sia Comp(KB) il suo completamento di Clark; sia L un letterale:

$KB \models_{CK} L$ sse $Comp(KB) \models L$

Esempio

P_{vola} : vola(X) ← uccello(X) ∧ not(anomalo(X)).
uccello(titti).
uccello(pingu).
anomalo(pingu).

$Comp(P_{vola})$: not(titti=pingu)
vola(V) ↔ uccello(V) ∧ not(anomalo(V))
uccello(V) ↔ V=titti ∨ V=pingu
anomalo(V) ↔ V=pingu

Esercizio:
$P \models_{CK}$ not(anomalo(titti)) perché not(titti=pingu)
$P \models_{CK}$ vola(titti) ?
$P \models_{CK}$ vola(pingu)?

Non monotonicità

- CK è non monotona perché:
  - $P1 \subseteq P2$ non implica $Comp(P1) \subseteq Comp(P2)$
    - quindi non implica $(P1 \models_{CK} L \rightarrow P2 \models_{CK} L)$

$P1_{vola}$: vola(X) ← uccello(X) ∧ not(anomalo(X)).
uccello(titti).
uccello(pingu).

⊆

$P2_{vola}$: vola(X) ← uccello(X) ∧ not(anomalo(X)).
uccello(titti).
uccello(pingu).
anomalo(pingu).

$Comp(P1_{vola})$: not(titti=pingu)
vola(V) ↔ uccello(V) ∧ not(anomalo(V))
uccello(V) ↔ V=titti ∨ V=pingu
not(anomalo(V))

$Comp(P2_{vola})$: not(titti=pingu)
vola(V) ↔ uccello(V) ∧ not(anomalo(V))
uccello(V) ↔ V=titti ∨ V=pingu
anomalo(V) ↔ V=pingu

$P1_{vola} \models_{CK}$ vola(pingu)

$P2_{vola} \models_{CK}$ not(vola(pingu))

Validità ma non completezza di NAFF rispetto a CK

- NAFF (Negation as Finite Failure) è la regola riportata sotto.
- È valida ma non sempre completa rispetto a CK, ovvero:
  - $KB \vdash_{NAFF} L$ implica $KB \models_{CK} L$
  - Se KB è in DCL, allora si ha la completezza
    $KB \models_{CK} L$ implica $KB \vdash_{NAFF} L$
  - ma esistono KB ed L tali che $KB \models_{CK} L$ ma non $KB \vdash_{NAFF} L$

$$\frac{\text{A ground fallisce in tempo finito}}{\text{not(A)}}\ \text{NAFF}$$

Esempio di incompletezza

P: $q \leftarrow not(q)$
$q \leftarrow q$

Comp(P): $q \leftrightarrow (q \vee not(q))$

? q origina una computazione infinita
ma Comp(P) |= q (perché $q \vee not(q) = true$)

Uso di CK e Comp(P) per ragionare sulla correttezza

- Nonostante l'inconveniente dell'incompletezza, CK è utile per ragionare sulla correttezza di programmi normali e definiti
  - Un programma P è corretto sotto l'assunzione CK sse Comp(P) è vera nell'interpretazione esterna

Esempio

Interpretazione esterna: half(X,Y) : Y è la metà intera di X, cioè la metà troncata

half(0,0).
half(s s X, s Y) $\leftarrow$ half(X,Y).

half(A,B) $\leftrightarrow$ A=0 $\wedge$ B=0 $\vee$ $\exists$x,y (A=s s x $\wedge$ B=s y $\wedge$ half(x,y))

Il completamento non è vero in particolare per A=s 0, perché non esiste x tale che s 0 = s s x. Dunque dobbiamo aggiungere:

half(s 0,0).

ESERCIZIO. Costruire ora il completamento e mostrare che è vero nell'interpretazione esterna

APPENDICE. Sintassi e semantica FOL (Firts Order Logic)

- Il completamento di un programma è un insieme di formule del primo ordine pieno. Richiamiamo sintassi e semantica per chi non avesse seguito logica. Ci limitiamo alle interpretazioni di Herbrandt.

LA SINTASSI di FOL:

- Base: Gli atomi (o formule atomiche) sono definiti come in DCL; sono le formule base.
- Passo: Le formule non atomiche sono della forma (not A), (A $\wedge$ B), (A $\vee$ B), (A $\rightarrow$ B), (A $\leftarrow$ B), (A $\leftrightarrow$ B), ($\forall$x A), ($\exists$x A)
  dove A, B sono (ricorsivamente) formule e x è una variabile

Esempio

Nei cortili ci sono solo cani e gatti, cioè se X è in un cortile è un cane o un gatto.
Nel cortile corte3 ci sono un gatto e un cane.

$\forall$X,C (incortile(X,C) $\rightarrow$ cane(X) $\vee$ gatto(X)) $\wedge$
$\exists$A,B (cane(A) $\wedge$ gatto(B) $\wedge$
incortile(A,corte3) $\wedge$ incortile(B,corte3))

Solite precedenze; i quantificatori hanno la precedenza di not

Variabili libere

- $\forall$ è detto quantificatore universale ed $\exists$ è detto quantificatore esistenziale
- In (Qx A), dove Q è un quantificatore, A è detta scopo di Qx
- Una variabile x occorre vincolata in una formula se occorre nello scopo di un quantificatore Qx; occorre libera se non occorre vincolata; può occorrere sia libera, sia vincolata.
- Una formula è chiusa se nessuna variabile occorre libera
- Le sostituzioni sostituiscono SOLO le occorrenze libere
- Le istanze chiuse o ground si ottengono sostituendo con termini ground tutte le occorrenze libere di variabili

ESEMPIO

$\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(X,Y)$

X quantificata

X libera

$(\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(X,Y))\{X/fido\}$
$= (\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(fido,Y)$

Le istanze ground nell'universo {fido,felix} sono 4:

$(\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(fido,fido)$
$(\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(fido,felix)$
$(\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(felix,fido)$
$(\forall X,C\ (incortile(X,C) \rightarrow cane(X) \vee gatto(X)) \wedge vede(felix,felix)$

Verità in un modello di Herbrandt

Base. A atomica chiusa:

$H \models A$ sse $A \in H$

Passo.

- $H \models \neg A$ sse non $H \models A$
- $H \models A \wedge B$ sse $H \models A$ e $H \models B$
- $H \models A \vee B$ sse $H \models A$ o $H \models B$
- $H \models \forall x\ A$ sse $H \models A\sigma$ per ogni istanza chiusa $A\sigma$
- $H \models \exists x\ A$ sse $H \models A\sigma$ per almeno un'istanza chiusa $A\sigma$

Nota. $A \rightarrow B$ equivale a $\neg A \vee B$
$A \leftarrow B$ equivale a $B \rightarrow A$
$A \leftrightarrow B$ equivale a $(A \rightarrow B) \wedge (A \leftarrow B)$

Esempio

$\exists x \exists y\ (\ cane(x) \wedge incortile(x,y)) \wedge \forall x\ (\neg cane(x) \vee abbaia(x))$

È vera nella seguente interpretazione?

cane(pluto). Universo = {pluto,felix,c1}
abbaia(pluto).
gatto(felix).
incortile(pluto,c1).

cane(pluto). Universo = {pluto,felix,c1}
abbaia(pluto).
gatto(felix).
incortile(pluto,c1).

1. $\exists x \exists y\ (\ cane(x) \wedge incortile(x,y))$ vero?
2. $\forall x\ (\neg cane(x) \vee abbaia(x))$ vero?

Risolviamo 1. Scegliamo l'istanza x=pluto

$\exists y\ (\ cane(pluto) \wedge incortile(pluto,y))$ vero?

Scegliamo l'istanza y = c1

$cane(pluto) \wedge incortile(pluto,c1)$ vero?

ora come nel proposizionale

cane(pluto). Universo = {pluto,felix,c1}
abbaia(pluto).
gatto(felix).
incortile(pluto,c1).

1. $\exists x \exists y\ (\ cane(x) \wedge incortile(x,y))$ vero?
2. $\forall x\ (\neg cane(x) \vee abbaia(x))$ vero?

Risolviamo 2. Le istanze sono x=pluto, x=felix, x=c1

$\neg cane(pluto) \vee abbaia(pluto)$ vero?
$\neg cane(felix) \vee abbaia(felix)$ vero?
$\neg cane(c1) \vee abbaia(c1)$ vero?
ora come nel proposizionale

Preparazione del II Compitino
Programma dettagliato

- STRUTTURA
  - 4 domande + 1 riserva
  - 1 esercizio a scelta fra 2

- Su DCL e Prolog, principalmente **esercizio**; si chiede di documentare le scelte; bene rivedere:
  - Come verificare correttezza positiva e copertura
  - Nozione di modo In e Out, modi In e ricorsione primitiva
  - Notazione Prolog per le liste
  - Difference lists
  - Not e floundering

- Su ricerca, da sapere bene:
  - Spazi di ricerca e loro rappresentazione
    - Spazio stati e spazio problemi, esempi
    - Problema di Ricerca R=(D,S,G)
    - Rappresentazione dei nodi e neighbors
  - L'algoritmo generico e depth-first, breadth-first
  - L'algoritmo generico completo
    - Costi e cammini
    - Predicati aperti (da cosa dipendono)
  - Strategie di ricerca: per ciascuna
    - Come si specializzano select e add_to_frontier
    - Completezza, ottimalità, complessità
    - Le strategie sono quelle viste a lezione:

- Blind (perché dette blind?)
  - Depth-first, breadth-first, lowest-cost first
- Euristiche o informate (perché dette così?)
  - Funzioni f, g, h
  - Best First
  - A* (sapere le condizioni di ammissibilità)
- Varianti: iterative deepening (perché utile?)

- Rappresentazione della conoscenza
  - Aspetti tipici
    - Problemi e soluzioni (qualità)
    - Aspetti epistemologici (dal problema alla rappresentazione)
    - Aspetti euristici (RRS, strategie)
    - Ingegneria conoscenza
  - Reti semantiche
    - Query e ordine logica usata
    - Reificazione
    - Reti semantiche
      - prop, isa, frame
  - Una classificazione delle logiche

- Ingegneria della conoscenza
  - Struttura di un sistema esperto e figure (ruoli) persone
  - Uso di prolog, meta-interpreti
    - Vanilla, pove con profonità, prove con ritardo, prove con askable, how
  - Problemi di interazione con l'utente
    - Cosa, come e quando chiedere
    - Spiegare
      - How, why, whynot
  - Debugging
    - Uso di how a fronte di risposta positiva errata
    - Uso di whynot a fronte di risposta mancante
    - Come trattare la terminazione
    - Tecniche di trasformazione per debugging

- Ultima parte (1 domanda, imprecisioni ammesse)
- Identità.
  - Le regole dell'identità
  - UNA
  - Problemi con la diversità e cenni su CLP
- NCL: i programmi normali (sintassi: not nel body)
  - Problemi con CWA
    - Non monotonia di CWA
    - Problemi di inconsistenza fuori da DCL
  - CET (teoria identità di Clark, esempi di prove di diversità)
  - Comp(P): come si costruisce, significato della parte only-if mediante la contrapposizione
  - L'assunzione CK
  - Non monotonia di CK
  - NAFF e CK (validità e problemi di incompletezza)
  - Uso di CK nella correttezza