Lezione del 28-5

Estensioni di DCL

Datalog, DCL e NCL

- **Abbiamo visto**
  - Datalog, DCL
- Vediamo
  - Uguaglianza e disuguaglianza
  - Negazione
    - CWA
    - NAFF, NCL

1. Identità e disuguaglianza

- L'unificazione corrisponde ad UNA (Unique Name Assumption):
  - termini ground distinti denotano individui distinti
- L'unificazione lavora anche su termini aperti e possiamo usarla nel corpo delle clausole:

```
anteriore(bici(X,Y),X).
        si puo' scrivere anche
anteriote(Bici,X) :- Bici = bici(X,_).
```


1. Identità e disuguaglianza

- Considerando i modelli di Herbrandt, l'unificazione porta a validità e completezza anche usando = nel corpo delle clausole:
  - validità: se σ è una sostituzione di risposta di un goal A, allora KB |= Aσ
  - completezza:
    - se KB |= $\exists$ x A , allora vi è una sostituzione di risposta
    - se il goal A fallisce, KB |= $\neg \exists$ x A

Esempio

KBsum:
sum(X,0,Z) :- Z=X.
sum(X,s(Y),Z) :- sum(X,Y,A), Z=s(A).

? sum(s(0),X,Y)
una risposta : Y = s(X)

? sum(s(0),X,0)
NO ➔ KBsum |= $\neg \exists$ x sum(s(0),x,0)

Disuguaglianza

- Le cose cambiano se usiamo la disuguaglianza con termini aperti; come risolvere disuguaglianze?

```
freq(mario,cl(1,a)).
freq(gigi,cl(1,a)).
freq(ugo,cl(2,a)).
freq(lea,cl(3,a)).

amico(X,Y) :- X ≠ Y, freq(X,A), freq(Y,A).
```

X=Y ha successo, ma non significa che non vi siano soluzioni per X ≠ Y;
Ad esempio:
X=mario, Y=gigi

Soluzioni parziali

- Dare solutori di disequazioni che lavorino nel contesto considerato
- Oppure ritardare le disuguaglianze aperte, come avviene in certe implementazioni di Prolog.

ESEMPIO SOLUZIONE 1.

```
freq(mario,cl(1,a)).
freq(gigi,cl(1,a)).

diverso(mario,gigi).
diverso(gigi,mario).

amico(X,Y) :- diverso(X, Y), freq(X,A), freq(Y,A).
```

NB: Per problemi di terminazione, non possiamo dare div(X,Y) :- div(Y,X).

1.1. Estensioni: CLP, Eqlog

- Fatti del tipo
  2 + 2 = 4
  sono esclusi dal nome unico.
- In tali casi si dovrebbe avere
  - Una teoria equazionale che assiomatizza l'uguaglianza
  - Un algoritmo di decisione per tale teoria

Teorie equazionali

- Assiomi dell'identità, "logici", cioè validi in ogni interpretazione di =
- Assiomi specifici: assiomatizzano l'identità per la struttura algebrica o la classe di strutture algebriche di interesse

$$\frac{}{T = T}\ id1$$

Assiomi dell'identità
come regole di inferenza

$$\frac{H(T) \quad T=T'}{H(T')}\ id2$$

ESEMPIO

Assiomi specifici per la somma

1. X+0=0
2. X+s(Y)=s(X+Y)

Una prova

$$\dfrac{\dfrac{}{s\ 0 + s\ 0 = s(s\ 0 + 0)}\ 2 \qquad \dfrac{}{s\ 0 + 0 = s\ 0}\ 1}{s\ 0 + s\ 0 = s\ s\ 0}\ id2$$

$$\dfrac{\dfrac{}{s\ 0 + s\ s\ 0 = s(s\ 0 + s\ 0)}\ 2 \qquad s\ 0 + s\ 0 = s\ s\ 0}{s\ 0 + s\ s\ 0 = s\ s\ s\ 0}\ id2$$

CLP, Eqlog

- In prolog **is** risolve in parte il problema per l'aritmetica.
- Vi sono estensioni di Prolog che consentono l'uso di operatori quale la somma, che portano a non avere più nome unico
  - CLP (Constraint Logic Programming)
  - Eqlog (implementazioni?)

2. Negazione

- Vediamo prima i legami fra negazione e assunzione del mondo chiuso.
- Poi consideriamo la negazione come
  - Negation as Failure e CWA
  - NAFF, Negation as Finite Failure

3.1. CWA e i suoi problemi

CWA è esprimibile in termini di negazione come fallimento, con la meta-regola:

se A non è dimostrabile da KB (dove A è chiusa), inferisco $\neg A$:

$$\frac{? A \text{ fail}}{\neg A}\ \text{cwa}$$

fail finito o infinito, cioè non successo

3.1.1. Possibile inconsistenza di CWA in NCL

Consideriamo il programma NCL proposizionale:

KB = {a ← ¬b, b ← ¬a}

i suoi modelli (di Herbrandt) sono {a}, {b}, {a,b}

Il programma non termina:

? a non ha prove finite, quindi fallisce infinitamente

? b non ha prove finite, quindi fallisce infinitamente

Applicando CWA

$$\frac{?a \text{ fail}}{\neg a}\ \text{cwa} \qquad \frac{? b \text{ fail}}{\neg b}\ \text{cwa}$$

$$\neg(a \leftarrow \neg b)$$

Con le tavole di verità, contradddice KB

3.1.2. CWA consistente, valida e completa in DCL

- CWA è usabile in modo consistente per le ***atomiche ground*** quando esiste modello minimo di Herbrandt M: siccome le formule di M coincidono con le atomiche ground dimostrabili,
  - se A è ground e ?A fail, allora
  - possiamo inferire ¬A, intendendo con ciò che ¬A è vera nel modello minimo
- In DCL l'esistenza del modello minimo è garantita; ciò consente un uso consistente di CWA, come "regola di inferenza" valida e completa rispetto alla verità nel modello minimo:

- Logica CWA per DCL. Poniamo, ***con A ground***
  KB $|=_{CWA}$ A se è vera in (appartiene a) M
  KB $|=_{CWA}$ ¬ A se A è falsa in (non appartiene a) M
- e:
  KB $|-_{CWA}$ A se esiste un albero di prova di A
  KB $|-_{CWA}$ ¬ A se non esiste un tale albero
- Letterali L:
  - positivi = atomiche A
  - negativi = atomiche negate ¬ A
- ***DCL+CWA***
  - ***valida*** KB $|-_{CWA}$ L ==> KB $|=_{CWA}$ L (L letterale ground)
  - ***completa*** KB $|=_{CWA}$ L ==> KB $|-_{CWA}$ L (L letterale ground)
  - ma CWA è un ***principio non calcolabile*** (terminazione indecidibile)

4. NAFF e NCL (Prolog)

- CWA non è calcolabile
- Si può usare NAFF (Negation as Finite Failure)
  - se non esiste alcun albero di prova finito di A ***ground***, allora inferisco not A

  NAFF è implementabile. In Prolog:

```
not(A) :- A, !, fail.
not(A).
```

- Se si ammette l'uso di not(A) nel body si ottiene NCL (Normal Clause Logic), di cui Prolog è un'implementazione

4.1. Perchè not(A) ground, floundering

- frequenta(ugo, ai).
- frequenta(gigi, alg).
- non_frequenta(X,Y) :- not(frequenta(X,Y)).

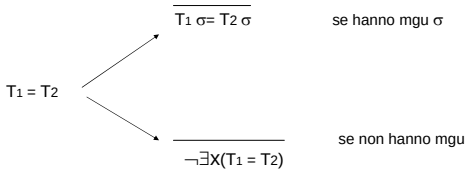
- Query: non_frequenta(ugo,C)
  mi aspetto C = alg
  ma ottengo NO

- NAFF come implementata in Prolog è NON valida, se si ha floundering: cioè vengono valutati goals negativi non ground

Il completamento di Clark

- Ovvero: che significato logico ha NAFF?
  - che significato logico ha il fallimento dell'unificazione?
  - che significato logico ha il fallimento

Circa la teoria di Clark: la diamo come schema di regola top down:



Circa il fallimento finito: la definizione completa Comp(r,P) di un predicato r in un programma P

Si ottiene come segue:

- normalizza ogni clausola con testa r:
  - $r(t_k) \leftarrow Body_k$ diventa
  - $r(v) \leftarrow \exists y_k\ (v=t_k \wedge Body_k)$ dove v sono nuove variabili
- Passa alla def. completa
  Comp(r,P) =
  $\forall v(\ r(v) \leftrightarrow \exists y_1\ (v=t_1 \wedge Body_1) \vee \ldots \vee \exists y_n\ (v=t_n \wedge Body_n)\ )$
- Nel caso non ci siano in P clausole con testa r,
  Comp(r,P) = $\forall v(\ r(v) \leftrightarrow$ false)

Circa il fallimento finito: la definizione completa Comp(P) di un programma P

Si ottiene come la congiunzione:

- Comp(P) = $\bigwedge_{r \in Predicati(P)}$ Comp(r,P)

Un esempio

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

$sum(A,B,C) \leftrightarrow \exists X(A=X \wedge B=0 \wedge C=X) \vee$
$\exists X,Y,Z(A=X \wedge B=s(Y) \wedge C=s(Z) \wedge sum(X,Y,Z))$
$odd(A) \leftrightarrow \exists X,Z\ (\ A=s(X) \wedge sum(X,X,Z)\ )$

Validita' di NAFF rispetto a Comp(P)

- Sia P un programma normale e si consideri una goal normale, G contenente atomi positivi aperti o chiusi e atomi negati ground; le risposte ottenute mediante computazioni prive di floundering sono cinsegyenze logiche di Comp(P).
- Non vale invece la completezza, ovvero ci sono conseguenze logiche che non vengono calcolate. La completezza si ottiene assumendo che il programma abbia particolari proprieta'.
- Non entreremo in ulteriori dettagli.

NOTA. E' utile usare Comp(P) per capire meglio i propri programmi.

APPENDICE. Sintassi e semantica FOL (Firts Order Logic)

- Comp(P) e' una formula 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.

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

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

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

X quantificata

X libera

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

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

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

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