Lezione 19

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
- 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

Esempio

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

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

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

$$s\ 0 + s\ s\ 0 = s\ s\ s\ 0\ \ id2$$

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,
    - e completamento di Clark

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 $\neg A$:

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

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 A è vera in (appartiene a) M
  KB $|=_{CWA} \neg$ A se A è falsa in (non appartiene a) M
- e:
  KB $|-_{CWA}$ A se esiste un albero di prova di A
  KB $|-_{CWA} \neg$ 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)

3.1.3. CWA non monotona

- L monotona: da KB1 $|-_L$ H e KB2 $\supseteq$ KB segue KB2 $|-_L$ H
- L non monotona: da KB1 $|-_L$ H e KB2 $\supseteq$ KB non segue necessariamente KB2 $|-_L$ H

  cioè nuova conoscenza può contraddire le conoscenze precedenti
- Vediamo un esempio

KB1:

```
amico(X,Y) :- frequenta(X,C), frequenta(Y,C).
frequenta(mario,ai).
frequenta(gigi,cs).
frequenta(alex,ai).
```

? ¬ amico(mario,gigi) Risposta in CWA per DCL: YES

KB2 contenente KB1:

```
amico(X,Y) :- frequenta(X,C), frequenta(Y,C).
frequenta(mario,ai).
frequenta(gigi,cs).
frequenta(gigi,ai).
frequenta(alex,ai).
```

? ¬ amico(mario,gigi) Risposta in CWA per DCL: NO

Nota

- Un fatto può essere considerato
  - vero se c'è evidenza per la sua verità
  - falso se c'è evidenza per la sua falsità
  - incerto se i casi precedenti non valgono
- CWA non ammette incertezza
- Considereremo nella seconda parte la presenza di incertezza.

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

4.2. NAFF e il completamento di Clark

- Che significato logico ha NAFF?
- Idea di Clark: 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): $$\frac{not(\ t_k = t'_k\ )}{not\ (f(t_1,..,t_k,..,t_n) = f(t'_1,..,t'_k,..,t'_n))}$$

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

ock $$\frac{}{not(\ T(X) = X\ )}$$

Esempio di prova in CET

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

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)

Passo 2.
per ogni simbolo di predicato n-ario p, introduco p(V), con V n-upla di nuove variabili; poi:

p(T) ← BODY diventa	p(V) ← V=T ∧ BODY
se BODY è vuoto:	p(V) ← V=T.
se p non è definito, inserisco	not p(V).

Passo 3. Se il passo 2 ha fornito not p(V), stop; altrimenti parto dal risultato del passo 2:

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

e costruisco la ***definizione completa di*** p:

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

Esempio

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

Passo 1.

CET per 0,s implicitamente dati.

Passo 2: normalizzo.

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: ottengo la definizione completa di sum.

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)).

- La definizione completa consente di derivare informazione negativa, ad es. per assurdo:
- assumiamo sum(s(0),s(0),s(0))
  se fosse vero avremmo uno dei due casi
  - $\exists$ x (s 0=x $\wedge$ s 0=0 $\wedge$ s 0=x) falso per CET
  - $\exists$ x,y,z (s 0=x $\wedge$ s 0=s(y) $\wedge$ s 0=s(z) $\wedge$ sum(x,y,z))
    ma allora dovremmo avere x = s 0, y=0, z=0 e quindi
    sum(s 0, 0, 0)
  - ma procedendo in modo analogo otteniamo che sum(s 0, 0, 0) deve essere falso (farlo per esercizio), ASSURDO

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) $\leftarrow$ uccello(X) $\wedge$ not(anomalo(X)).
uccello(titti).
uccello(pingu).
anomalo(pingu).

$Comp(P_{vola})$: not(titti=pingu)
vola(V) $\leftrightarrow$ uccello(V) $\wedge$ not(anomalo(V))
uccello(V) $\leftrightarrow$ V=titti $\vee$ V=pingu
anomalo(V) $\leftrightarrow$ 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) $\leftarrow$ uccello(X) $\wedge$ not(anomalo(X)).
uccello(titti).
uccello(pingu).

$\subseteq$

$P2_{vola}$: vola(X) $\leftarrow$ uccello(X) $\wedge$ not(anomalo(X)).
uccello(titti).
uccello(pingu).
anomalo(pingu).

$Comp(P1_{vola})$: not(titti=pingu)
vola(V) $\leftrightarrow$ uccello(V) $\wedge$ not(anomalo(V))
uccello(V) $\leftrightarrow$ V=titti $\vee$ V=pingu
not(anomalo(V))

$Comp(P2_{vola})$: not(titti=pingu)
vola(V) $\leftrightarrow$ uccello(V) $\wedge$ not(anomalo(V))
uccello(V) $\leftrightarrow$ V=titti $\vee$ V=pingu
anomalo(V) $\leftrightarrow$ 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 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

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