NEGAZIONE IN DATALOG

- in alcuni casi è necessario utilizzare la negazione nel corpo delle regole

- questo può però dar luogo a problemi nel definire la semantica della base di dati

- la nozione di *negazione stratificata* permette di limitare l'uso della ricorsione a programmi la cui semantica è ben definita

Esempio

EDB = { R(0) }

IDB: $\mathtt{P(X)} \leftarrow \mathtt{R(X), NOT\ Q(X)}$
$\mathtt{Q(X)} \leftarrow \mathtt{R(X), NOT\ P(X)}$

la base di dati ha due possibili modelli:
{R(0), P(0)} e {R(0), Q(0)}
ma non esiste un modello minimo

NEGAZIONE IN DATALOG

Esempio 2 (ok)

predicato estensionale `Voli(C,X,Y,P,A)`

[esiste un volo della compagnia `C` dalla città `X` alla città `Y` con partenza all'ora `P` e arrivo all'ora `A`]

predicato intensionale `UAsolo(X,Y)`: `UA` vola da `X` a `Y` (anche attraverso altre città), ma `AA` no

$$\mathtt{UAragg}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{Voli}(\mathtt{ua},\mathtt{X},\mathtt{Y},\_,\_)$$
$$\mathtt{UAragg}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{UAragg}(\mathtt{X},\mathtt{Z}),\mathtt{UAragg}(\mathtt{Z},\mathtt{Y})$$
$$\mathtt{AAragg}(\mathtt{X},\mathtt{Y}) \leftarrow (\mathtt{Voli}(\mathtt{aa},\mathtt{X},\mathtt{Y},\_,\_)$$
$$\mathtt{AAragg}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{AAragg}(\mathtt{X},\mathtt{Z}),\mathtt{AAragg}(\mathtt{Z},\mathtt{Y})$$
$$\mathtt{UAsolo}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{UAragg}(\mathtt{X},\mathtt{Y}),\mathtt{NOT}\ \mathtt{AAragg}(\mathtt{X},\mathtt{Y})$$

differenza insiemistica di `UAragg` e `AAragg`: la semantica è ben definita

NEGAZIONE IN DATALOG

Esempio 2 (segue)

dato l'EDB

```
Voli(ua,sf,den,930,1230)
Voli(aa,sf,dal,900,1430)
Voli(ua,den,chi,1500,1800)
Voli(ua,den,dal,1400,1700)
Voli(aa,dal,chi,1530,1730)
Voli(aa,dal,ny,1500,1900)
Voli(aa,chi,ny,1900,2200)
Voli(ua,chi,ny,1830,2130)
```

`UAragg` consiste nelle seguenti coppie:
`(sf,den), (sf,dal), (sf,chi), (sf,ny),`
`(den,dal), (den,chi), (den,ny), (chi,ny)`

`AAragg`: `(sf,dal), (sf,chi), (sf,ny), (dal,chi),`
`(dal,ny), (chi, ny)`

`UASolo` è la differenza di questi insiemi di coppie: `(sf,den), (den,dal), (den,chi), (den,ny)`

STRATIFICAZIONE

- per evitare i problemi dovuti all'uso della ricorsione attraverso la negazione, ci si restringe alla ricorsione in cui la negazione stratificata

- quando la negazione è stratificata esiste un algoritmo per calcolare un particolare minimo punto fisso, che corrisponde al contenuto informativo "intuitivo"

- stratificazione:
  - grafo i cui nodi corrispondono ai predicati IDB
  - arco dal nodo A al nodo B etichettato da - se una regola con il predicato A nella testa contiene nel corpo un atomo negato con il predicato B
  - arco dal nodo A al nodo B se una regola con predicato di testa A contiene nel corpo un atomo non-negato con il predicato B

STRATIFICAZIONE

- se il grafo ha un ciclo che contiene uno o più archi negativi la ricorsione non è stratificata, altrimenti il grafo è stratificato

- i predicati IDB possono essere raggruppati in strati: lo strato di un predicato A è il più grande numero di archi negativi su un cammino che comincia da A

- se la ricorsione è stratificata è possibile valutare i predicati IDB in accordo alla semantica bottom-up nell'ordine dei loro strati, partendo dal più basso

- in questo modo si ottiene come punto fisso il contenuto informativo della base di dati

NEGAZIONE IN DATALOG

- inoltre la nozione di safety viene estesa alla negazione:
  - ogni variabile che compare nella testa della regola deve comparire in un atomo non negato nel corpo
  - ogni variabile che compare in un atomo negato nel corpo deve anche comparire in un atomo non negato nel corpo
- garantisce che la valutazione di atomi negati non debba generare legami per le variabili

RICORSIONE IN SQL3

- solo ricorsione *lineare*: solo un sottogol ricorsivo in ogni regola

- stratificazione applicata a negazione ed estesa agli aggregati

- comando `WITH` per definire relazioni IDB

  `WITH R AS` <*definizione di R*>
  <*interrogazione che coinvolge R*>

  si definisce cioè una relazione temporanea chiamata `R` e si usa `R` in un'interrogazione

  - si possono definire più relazioni (separate da virgola)

  - ognuna di queste definizioni può essere ricorsiva e le relazioni possono essere mutuamente ricorsive

  - ogni relazione coinvolta in una ricorsione deve essere preceduta dalla parola chiave `RECURSIVE`

RICORSIONE IN SQL3

1. parola chiave `WITH`

2. una o più definizioni, separate da virgole

   ogni definizione consiste in

   (a) parola chiave opzionale `RECURSIVE` (deve essere specificata se la relazione che si sta definendo è ricorsiva)

   (b) il nome della relazione che si sta definendo

   (c) la parola chiave `AS`

   (d) l'interrogazione che definisce la relazione

3. interrogazione che può far riferimento alle precedenti definizioni e forma il risultato del comando `WITH`

RICORSIONE IN SQL3

Esempio `Voli(lineaAerea,da,a,parte,arriva)`

predicato `Raggiunge` (non lineare)

```
WITH RECURSIVE Raggiunge(da,a) AS
       (SELECT da,a FROM Voli)
     UNION
       (SELECT R1.da, R2.a
        FROM Raggiunge AS R1, Raggiunge AS R2
        WHERE R1.a = R2.a)
SELECT * FROM Raggiunge;
```

predicato `Raggiunge` (lineare)

```
WITH Coppie AS SELECT da,a FROM Voli,
     RECURSIVE Raggiunge(da,a) AS
       Coppie
     UNION
       (SELECT Coppie.da, Raggiunge.a
        FROM Coppie, Raggiunge
        WHERE Coppie.a = Raggiunge.da)
SELECT * FROM Raggiunge;
```

RICORSIONE IN SQL3

- le definizioni all'interno del comando `WITH` sono disponibili solo all'interno del comando e non possono essere usate al di fuori di questo

- nel comando `WITH` si possono definire viste invece che tabelle

  la differenza sintattica è che si usa la parola chiave `VIEW` nella definizione della relazione

- esempio:
  `VIEW Coppie AS SELECT da, a FROM Voli`

- se si definisce `Coppie` come vista, tale relazione non viene effettivamente costruita, ma il suo uso nella costruzione di `Raggiunge` è sostituito dall'uso delle componenti dalle tuple di `Voli`

RICORSIONE IN SQL3

- restrizione a negazione stratificata

esempio ok:

```
WITH
 Triple AS SELECT lineaAerea, da, a FROM Voli,
 RECURSIVE Raggiunge(lineaAerea,da,a) AS
   Triple
    UNION
   (SELECT Triple.lineaAerea, Triple.da,
           Raggiunge.a
    FROM Triple, Raggiunge
    WHERE Triple.a = Raggiunge.da AND
          Triple.lineaAerea = Raggiunge.lineaAerea)
(SELECT da,a FROM Raggiunge
 WHERE lineaAerea = ´UA´)
EXCEPT
(SELECT da,a FROM Raggiunge
 WHERE lineaAerea = ´AA´);
```

RICORSIONE IN SQL3

- esempio non ok:

```
WITH
   RECURSIVE P(X) AS
     (SELECT * FROM R)
     EXCEPT
     (SELECT * FROM Q),
   RECURSIVE Q(X) AS
     (SELECT * FROM R)
     EXCEPT
     (SELECT * FROM P)
SELECT * FROM P;
```

RICORSIONE IN SQL3

- viene richiesta la stratificazione anche rispetto ad altri costrutti non monotoni, ad esempio aggregati

```
WITH
   RECURSIVE P(X) AS
     (SELECT * FROM R)
     UNION
     (SELECT * FROM Q),
   RECURSIVE Q(X) AS
     SELECT SUM(X) FROM P
SELECT * FROM P;
```

es. R contiene 12 e 34 Q passa da $\emptyset$ a $\{(46)\}$ a $\{92\}$ $\Rightarrow$ non monotono