Università degli Studi
di Roma "Sapienza"

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Anno accademico 2007/08

DATA EXCHANGE

Relazione relativa al corso di:
Seminari di Ingegneria del Software
Prof. Giuseppe De Giacomo

a cura di:
LUCA PORRINI
FRANCESCO MINICUCCI

Indice

RIFERIMENTI

[1] -- Maurizio Lenzerini: Data Integration: A Theoretical Perspective.
PODS 2002: 233-246

[2] -- Phokion G. Kolaitis: Schema mappings, data exchange, and metadata management.
PODS 2005: 61-75

[3] -- Phokion G. Kolaitis, Jonathan Panttaja, Wang Chiew Tan: The complexity of data exchange.
PODS 2006: 30-39

[4] -- Schema Mappings, Data Exchange, and Metadata Management
PODS 2005: Invited Talk, June 2005 (powerpoint slides - large file)

[5] -- Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, Lucian Popa: Data Exchange: Semantics and
Query Answering.
ICDT 2003: 207-224

[6] -- Data Exchange, Talk at the Database Seminar, University of Toronto, November 2003
(postscript slides)

[7] -- Ronald Fagin, Phokion G. Kolaitis, Lucian Popa: Data exchange: getting to the core.
ACM Trans. Database Syst. 30(1): 174-210 (2005)

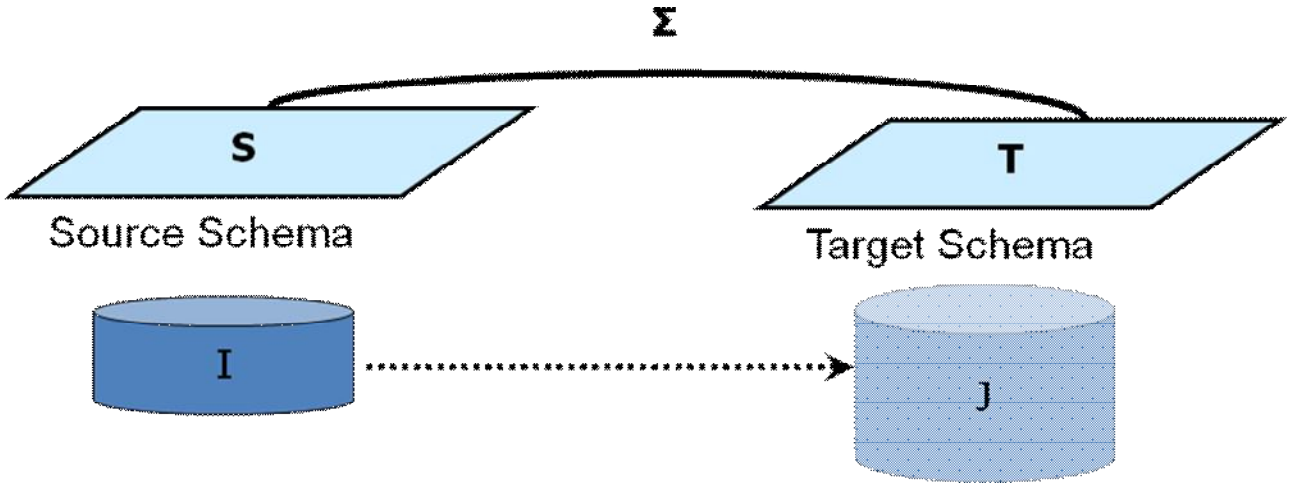
[8] -- Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, Wang Chiew Tan: Composing schema mappings:
Second-order dependencies to the rescue.
ACM Trans. Database Syst. 30(4): 994-1055 (2005)

[9] -- Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
On reconciling data exchange, data integration, and peer data management.
PODS 2007: 133-142

[10] -- Georg Gottlob, Alan Nash: Data Exchange: Computing Cores in Polynomial Time

INTRODUZIONE AL DATA EXCHANGE

Il problema del Data Exchange, anche noto come *"Data Translation"*, consiste nel prendere dati strutturati in un determinato schema, detto "Source Schema", e trasformarli in dati strutturati in un altro schema, detto "Target Schema". L'istanza del Target Schema risultante dovrà rispecchiare i dati residenti nel Source Schema nel modo più accurato possibile.



La problematica del Data Exchange è stata studiata moltissimo soprattutto negli ultimi anni, per porre rimedio alla crescente necessità di spostare in un nuovo schema dati che risiedono in diversi schemi indipendenti, che non sempre hanno lo stesso formato dei dati. L'importanza del Data Exchange è aumentata esponenzialmente con la proliferazione di *dati-Web* in formati differenti, con il rapido sviluppo di applicazioni di e-business e con l'avvento di dati *semi-strutturati*, come i DTD (*Document Type Definition*) o gli XML schema.

Ciononostante, tale problema pone le sue radici ben più nel passato, e si ripresenta in diverse situazioni con elevata ricorrenza. Phil Berstein, ricercatore principale del *"Database Group"* della Microsoft, lo considera addirittura *"Il più vecchio problema di basi di dati"*.

Uno dei primi sistemi di Data Exchange è stato **EXPRESS** (**EX**traction, **P**rocessing, and **RES**tructuring **S**ystem), sviluppato nel laboratorio di ricerca della IBM a San Jose nel 1977. Lo scopo principale di Express era di trasformare dati tra database gerarchici (nei quali i dati vengono organizzati in una serie di insiemi legati tra loro da relazioni di "possesso", in cui un insieme può *"possedere"* altri insiemi di dati, ma può *"essere posseduto"* da un solo altro insieme, dando vita ad una struttura risultante ad albero).

In una tipica situazione di Data Exchange, abbiamo un source schema **S** e un target schema **T**, ed assumiamo che **S** e **T** siano disgiunti. Dal momento che **T** può essere uno schema creato indipendentemente, può avere le sue proprie condizioni, che sono mostrate come un insieme di sentenze Σt in qualche formalismo logico su **T**. Inoltre, dobbiamo avere a disposizione un modo di modellare le relazioni tra il source schema e il target schema, e per farlo ricorriamo alle *dipendenze source-to-target* Σst, che specificano come e quali dati del source schema devono apparire nel target schema.

Uno schema mapping **M** è una terna (**S**, **T**, Σ), dove $\Sigma = \Sigma t \cup \Sigma st$, necessaria per descrivere come i dati debbano essere trasformati da una rappresentazione all'altra. Se **I** è un istanza del source schema, allora una **soluzione** per **I** è un'istanza **J** del target schema tale che **<I,J>** soddisfa Σ.

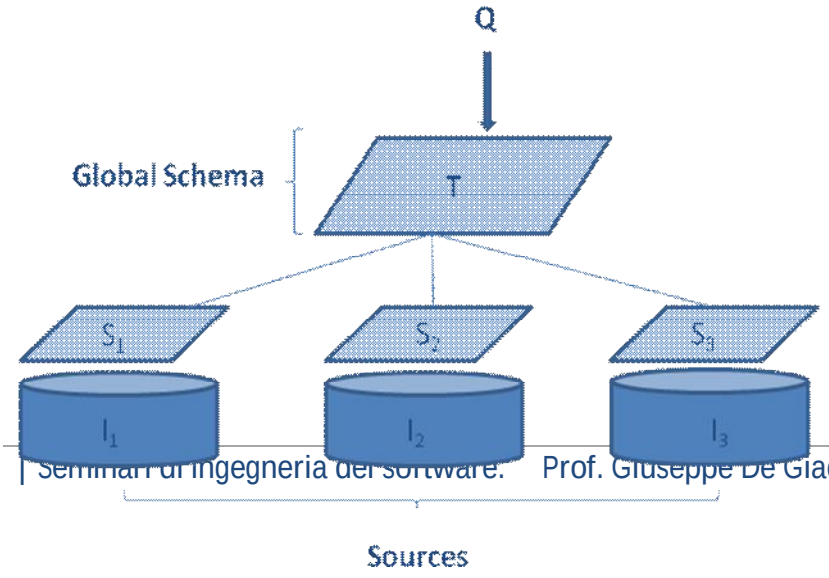
Il Data Exchange sta alla base di un gran numero di altre problematiche recenti, quali il **Data Warehousing** (creazione di archivi informatici, letteralmente *"Magazzini di dati"*, contenenti i dati di un'organizzazione, utili per produrre facilmente relazioni ed analisi), i **tasks ETL** ("*Extract-Transform-Load"*, una serie di tools responsabili dell'estrazione di dati da sorgenti, della loro "pulizia" e del successivo caricamento in un data warehouse target), l'**XML publishing** e l'**XML storage**.

Nel prossimo paragrafo saranno analizzate le caratteristiche comuni e le differenze fondamentali tra il Data Exchange e il problema non meno importante del **Data Integration**, mentre nel paragrafo 3 vedremo un approccio più dettagliato e specifico alle problematiche e alle soluzioni offerte dal Data Exchange.

DATA INTEGRATION E DATA EXCHANGE: CARATTERISTICHE E DIFFERENZE

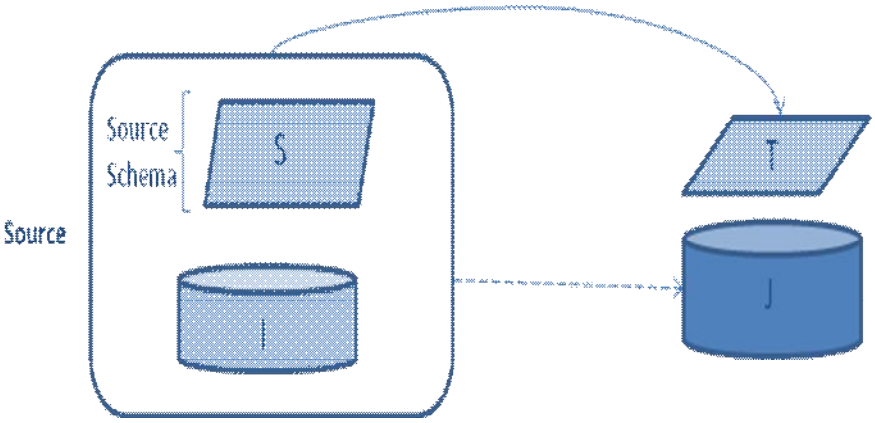
Il problema dell'interoperabilità dei dati è stato studiato profondamente ed è stato affrontato da diversi punti di vista. Ci sono due approcci differenti anche se correlati, data integration (data federation) e data exchange (data translation). Queste due tecniche hanno diverse cose in comune ma anche delle ovvie differenze. Ci sembra molto interessante approfondire questo aspetto per chiarire e definire i vari aspetti dei problemi ma prima di fare questo riteniamo opportuno richiamare alcune cose di data integration.

Data integration. Richiamiamo alcuni concetti essenziali del data integration in modo da poter apprezzare maggiormente gli aspetti diversificati rispetto al data exchange. Questo argomento è trattato in maniera molto chiara in modo da poter offrire una guida interessante su rif. [1: Len, data Integration: a theoretical perspective]. Data integration è il problema di combinare dati residenti su differenti sorgenti e fornire all'utente un vista unificata di questi dati. I sistemi di data integration che andiamo ad analizzare sono basati su un global schema ed un insiemi di sorgenti. Le sorgenti contengono dati reali mentre il global schema fornisce una vista riconciliata integrata ed unificata dei dati. Modellare la relazione tra sorgenti e global schema è un aspetto cruciale. Passando ad un aspetto formale possiamo dire che un sistema data integration è una tripla (G, S, M) dove G è il global schema, S è il source schema mentre M è il mapping tra G e S che è un insieme di asserzioni che stabiliscono una connessione tra le sorgenti e lo schema globale. Ci sono due approcci fondamentali per costruire tale mapping. Il primo chiamato LAV (Local As View) che esprime le sorgenti in termini di



global schema ed, il secondo, GAV (Global As View) che esprime il global schema in termini di sorgenti. In realtà esiste un terzo approccio, chiamato GLAV che è dato dall'unificazione dei due precedenti. Indipendentemente da quale approccio si sceglie il servizio base che il data integration deve fornire è il query answering, ossia rispondere a delle query poste in termini di global schema.

Data integration & data exchange. Data exchange è il problema di prendere dati strutturati sotto una schema, source schema, e di trasformarli in dati strutturati sotto un altro schema, target schema.



Discussioni interessanti su questo confronto sono trattate su [2: Kol, Schema mappings Data exchange and metadata management] e [5: Fag, Data exchange: semantics and query answering]. Come detto precedente data integration è una tripla (G, S, M) con G e S espresse in linguaggio che permette di esprimere vincoli. Data exchange setting invece è una quadrupla (S, T, $\sum_{st}$, $\sum_{t}$) che può essere pensato come un data integration system nel quale S è il source schema, T e $\sum_{t}$ lo schema mapping e le s-t tgds in $\sum_{st}$ solo le asserzioni del data integration system. In un sistema LAV ogni asserzione in M lega un elemento del source schema S ad una query sul global schema, quindi si assume $\sum_{t} = \emptyset$. In un GAV system ogni asserzione lega un elemento del global schema ad una query sul source schema S. Quindi le dipendenze s-t in $\sum_{st}$ legano una query sul source schema S a una query sul target schema T, ne consegue che un data exchange setting non è né un LAV né un GAV system, bensì può essere pensato come un GLAV system. Adesso andiamo ad analizzare le differenze.

Scopo principale. In data integration, l'obiettivo è rappresentare dati da diverse sorgenti eterogenee in un'unificata vista su un global schema. Questa vista è virtuale, i dati rimangono nelle sorgenti e sono acceduti dagli utenti simbolicamente tramite il global schema. In data exchange, l'obiettivo è prendere una data istanza sorgente e trasformarla in un'istanza target tale che soddisfa le specifiche dello schema mapping e riflette la sorgente nella maniera più accurata possibile. Quindi al contrario del data integration questa istanza target è un'istanza "materializzata", fisica e non una vista virtuale.

Vincoli. Come detto prima, in data exchange, il target schema viene creato con i propri vincoli. In data integration il global schema G è comunemente una riconciliata e virtuale vista di insiemi eterogenei di sorgenti e non ha vincoli.
Query answering. Considerato un problema di data exchange e data un'istanza sorgente potrebbero essere molteplici le istanze target, chiamate soluzioni, che soddisfano lo schema mapping. Questa situazione dà luogo ad una questione fondamentale del data exchange che verrà affrontata in dettaglio nel prosieguo e che qui ci limitiamo ad accennare. Data un'istanza sorgente, quale soluzione riteniamo essere la migliore? Sia data integration che data exchange utilizzano il concetto di certain answers come semantica standard delle query answering. In data integration, per calcolare le certain answers di queries su global schema vengono utilizzate le istanze sorgenti mentre in data exchange setting queries su target schema potrebbero utilizzare solo istanze target senza riferirsi alle originali istanze di sorgente. Questo perché una volta effettuato lo scambio dei dati le istanze sorgenti potrebbero non essere più disponibili.

IL PROBLEMA DEL DATA EXCHANGE

Uno schema è una collezione finita R = {R1,. . . , Rk} di simboli di relazione. Un'*istanza* I su uno schema R è una funzione che associa a ogni simbolo di relazione Ri una relazione I(Ri). Data una tupla t che occorre in una relazione R, denotiamo con R(t) l'associazione tra t e R: tale associazione si chiama *fatto*. Un'istanza può quindi chiaramente essere rappresentata come un insieme di fatti. Una *dipendenza* su R è una sentenza in qualche formalismo logico su R.

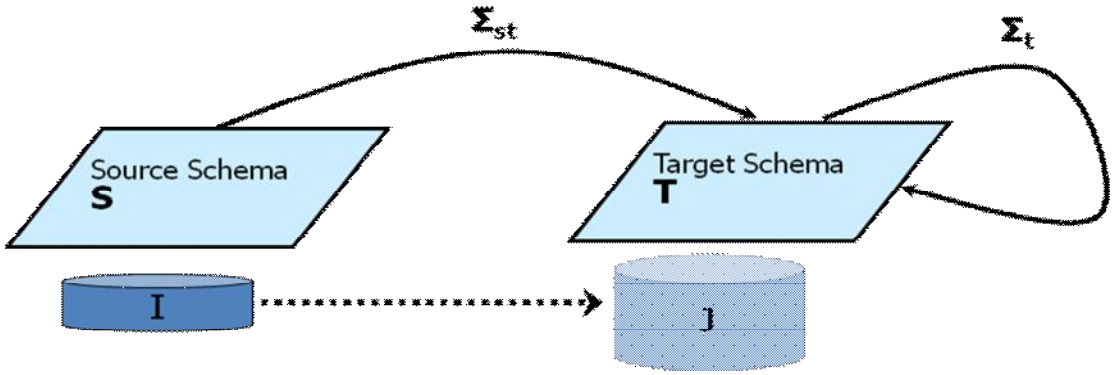
Siano S = {S1 , . . . , Sn} e T = {T1 , . . . , Tm} due schemi disgiunti. Chiameremo S "source schema" ed Si i suoi simboli di relazione; allo stesso modo chiameremo T "target schema" e Ti i suoi simboli di relazione. Sia I una istanza del source schema e J una istanza del target schema: chiameremo <I,J> l'istanza K sullo schema S ∪ T tale che K(Si) = I (Si) e K(Tj) = J (Tj), per i ≤ n e j ≤ m.

Una dipendenza *source-to-target* (il cui insieme si denota con Σst) è una dipendenza della forma $\forall x(\phi_S(x) \rightarrow \chi_T(x))$, dove $\phi_S(x)$ è una formula (con variabili libere x) in qualche formalismo logico su S, mentre $\chi_T(x)$ è una formula (con variabili libere x) in qualche formalismo logico su T.

Una dipendenza *target* (il cui insieme si denota con Σt) è una dipendenza sul target schema T, il cui formalismo logico potrebbe essere anche differente da quelli usati per le dipendenze source-to-target.

Anche il source schema può ovviamente avere dipendenze, ma risultano essere poco rilevanti per il problema del Data Exchange, dal momento che le consideremo essere soddisfatte da ogni istanza del source schema che riceviamo come input per il problema. Tuttavia giocano un ruolo importante nella derivazione delle dipendenze source-to-target.

Una configurazione (S, T, Σst , Σt) per il problema del data Exchange consiste in un source schema S, un target schema T, un insieme Σst di dipendenze source-to-target, e un insieme Σt di dipendenze target. Tale configurazione prende il nome di *Schema Mapping.*



Data un'istanza finita I del source schema, si deve trovare un'istanza finita J del target schema tale che <I,J> soddisfi Σst e J soddisfi Σt. In tal caso, J si chiama *soluzione* per I. L'insieme di tutte le soluzioni per I si denota con Sol(I). L'unico input al problema è un'istanza del source schema: la configurazione precedentemente descritta è da considerarsi fissata.
Essenzialmente ad uno schema mapping sono associati due tipi di problemi:

- PROBLEMA DECISIONALE: data un'istanza I del source schema, esiste una soluzione J per I?
- PROBLEMA FUNZIONALE: data un'istanza I del source schema, trovare una soluzione J per I, dopo aver accertato che esista

Per motivi pratici e per la garanzia della maggior parte dei risultati ottenuti, ogni dipendenza source-to-target in Σst viene considerata una *dipendenza tuple generating (tgd)* nella forma $\forall x(\phi S(x) \rightarrow \exists y\ \psi T(x, y))$, dove $\phi S(x)$ è una congiunzione di formule atomiche su S e $\psi T(x, y)$ è una congiunzione di formule atomiche su T.

- *Esempio (source-to-target tgd):*
  $(Student(s) \wedge Enrolls(s,c)) \rightarrow \exists t\ \exists g\ (Teaches(t,c) \wedge Grade(s,c,g))$

Peraltro, i source-to-target tgd generalizzano le specifiche principali usate nel Data Integration, in cui le views sono "sound" e sono definite da conjunctive queries:

- generalizzano LAV (local-as-view):
  $P(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}\ \psi(\boldsymbol{x}, \boldsymbol{y})$, *dove P è un source schema.*
- generalizzano GAV (global-as-view):
  $\varphi(\boldsymbol{x}) \rightarrow R(\boldsymbol{x})$, *dove R è un target schema*

Inoltre, ogni dipendenza target in Σt è anch'essa una dipendenza *tuple generating*

(tgd) nella forma $\forall x(\phi T(x) \rightarrow \exists y\ \psi T(x, y))$, oppure una dipendenza *equality-generating (egd)* nella forma $\forall x(\phi T(x) \rightarrow (x1 = x2))$. In entrambi i casi, $\phi T(x)$ e $\psi T(x, y)$ sono congiunzioni di formule atomiche su T.

- *Esempio (target tgd):*
  Dept (did, dname, mgr_id, mgr_name) $\rightarrow$ *Mgr (mgr_id, did)*
  (a target inclusion dependency constraint)

- *Esempio (target egd):*
  $(Mgr\ (e, d_1) \wedge Mgr\ (e, d_2)) \rightarrow (d_1 = d_2)$
  (a target key constraint)

In ogni caso l'assunzione di avere tgd o egd come dipendenze target è da considerarsi naturale, in quanto queste due classi comprendono insieme le dipendenze implicazionali, che includono essenzialmente tutte le condizioni sui database relazionali comunemente usate.

Il prossimo esempio mostra come possa esserci più di una possibile soluzione per un dato problema di Data Exchange.

- *Esempio (Problema di Data Exchange)*

Si consideri un problema di Data Exchange nel quale il source schema consiste in due simboli di relazione binari: *ImpCitta*, che associa gli impiegati alle città in cui lavorano, e *ViveIn*, che associa gli impiegati alla città in cui vivono. Assumiamo che il target schema consista in tre simboli di relazione binari così definiti: *Residenza*, che associa gli impiegati alle proprie città di residenza, *ImpDipart*, che associa gli impiegati ai dipartimenti, e *DipartCitta*, che associa i dipartimenti con le loro città. Assumiamo inoltre che Σt = Le tgd source-to-target e l'istanza del source schema sono definite come segue (dove d1, d2, d3 e d4 sono delle label di riferimento che ci saranno utili successivamente):

Σst: (d1) ImpCitta(i, c) $\rightarrow \exists R$ Residenza(i, R)
(d2) ImpCitta(i, c) $\rightarrow \exists D$(ImpDipart(i, D) $\wedge$ DipartCitta(D, c))
(d3) ViveIn(i, c) $\rightarrow$ Residenza(i, c)
(d4) ViveIn(i, c) $\rightarrow \exists D \exists C$(ImpDipart(i, D) $\wedge$ DipartCitta(D, C))

I = {ImpCitta(Luca, Roma), ImpCitta(Francesco, Cisterna),
ViveIn(Luca, Avezzano), ViveIn(Francesco, Latina) }.

Dal momento che i tgd in Σst non specificano completamente l'istanza target, ci sono molte soluzioni che sono consistenti con la specifica. Una soluzione è, ad esempio:

J0 = {Residenza(Luca, Avezzano), Residenza(Francesco, Latina),
ImpDipart(Luca, D1), ImpDipart(Francesco, D2),
DipartCitta(D1 , Roma), DipartCitta(D2 , Cisterna)}

dove D1 e D2 rappresentano valori "sconosciuti", ovvero valori che non appaiono nell'istanza del source schema. Tali valori sono detti *"labeled nulls"*, mentre i valori che appaiono nell'istanza del source sono detti *costanti*. Intuitivamente, D1 e D2 sono usati per "dare valori" al quantificatore esistenziale D che appare in (d2) per le due tuple del source ImpCitta(Luca,Roma) e ImpCitta(Francesco,Cisterna). Al contrario, due costanti (Avezzano e Latina) sono usate per "dare valori" alla variabile di quantificazione esistenziale R di (d1), così da soddisfare (d1) per le due tuple sopra citate.

Anche le seguenti sono soluzioni per il problema:

J1 = {Residenza(Luca, Avezzano), Residenza(Francesco, Latina),
Residenza(Luca, R1), Residenza(Francesco, R2),
ImpDipart(Luca, D1), ImpDipart(Francesco, D2),
DipartCitta(D1 , Roma), DipartCitta(D2 , Cisterna)}

J2 = {Residenza(Luca, Avezzano), Residenza(Francesco, Latina),
ImpDipart(Luca, D), ImpDipart(Francesco, D),
DipartCitta(D, Roma), DipartCitta(D, Cisterna)}

L'istanza J1 differisce da J0 perchè ha due tuple "Residenza" in più nelle quali le città di residenza di Luca e Francesco sono due null, R1 e R2 rispettivamente. L'istanza J2 differisce da J0 perchè usa lo stesso null (chiamato D) per rappresentare il dipartimento "sconosciuto" in cui lavorano sia Luca che Francesco (un'assunzione che, in ogni caso, non è parte della specifica).

Dal momento che abbiamo dimostrato l'esistenza di più di una soluzione per un dato problema, sorge il problema di stabilire quale sia la migliore (intuitivamente, quella che contenga nè più nè meno di quanto le specifiche richiedano). A tal fine analizzeremo nel prossimo paragrafo il concetto di *soluzioni universali*.

SOLUZIONI UNIVERSALI

In questo paragrafo vedremo una speciale classe di soluzioni che sono dette soluzioni *universali*. Una soluzione universale ha una seria di "buone" proprietà che giustificano la sua scelta per la semantica del Data Exchange. In particolare, una soluzione universale non contiene né più ne meno di quanto richiesto dalle specifiche, ed è considerata la soluzione più "generale" nel Data Exchange. Iniziamo con l'introdurre alcune terminologie e notazioni.
Denotiamo con Const l'insieme di tutti i valori che compaiono nelle istanze del source, e le chiamiamo *costanti*. In aggiunta, assumiamo l'esistenza di un insieme infinito Var di valori, che chiamiamo *labeled nulls*, tali che Var ∩ Const = 0(insieme nullo). Denotiamo con I , I′ , I1 , I2 , . . . le istanze del source schema S con valori in Const; denotiamo invece con J, J′ , J1 , J2 , . . . le istanze del target schema T con valori in Const ∪ Var.

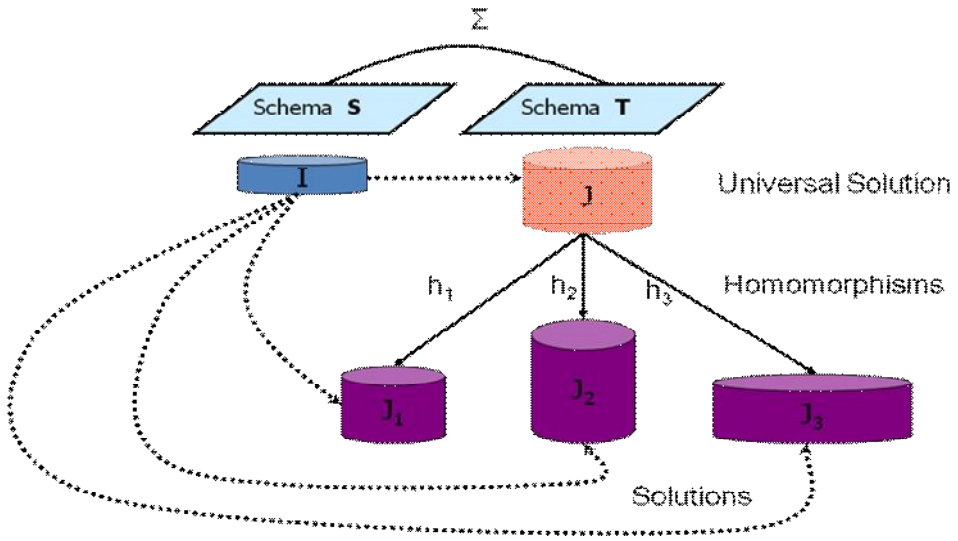
Siano K1 e K2 due istanze su R con valori in Const ∪ Var.
Un *omomorfismo* h : K1→ K2 è un mapping da Const ∪ Var(K1) a Const ∪ Var(K2) tale che:

1. h(c) = c, per ogni c che appartiene a Const;
2. per ogni fatto Ri (t) di K1 , si ha che Ri (h(t)) è un fatto di K2 (dove, se t= (a1 , . . . , as), allora h(t) = (h(a1), . . . , h(as))).

Diremo inoltre che K1 è omomorficamente equivalente a K2 se c'è un omomorfismo h : K1 →K2 e un omomorfismo h′ : K2→ K1 .

Intuitivamente un omomorfismo tra due strutture relazionali è un mapping che preserva i fatti.

Consideriamo una configurazione di data Exchange (S, T, Σst , Σt). Se I è una istanza del source schema, allora una *soluzione universale* per I è una soluzione J per I tale che per ogni soluzione J ′ per I, esiste un omomorfismo h : J→ J ′ .



- *Esempio (Soluzioni Universali)*

Prendiamo in considerazione le 3 soluzioni proposte nell'esempio precedente (problema di Data Exchange).

J0 = {Residenza(Luca, Avezzano), Residenza(Francesco, Latina),
ImpDipart(Luca, D1), ImpDipart(Francesco, D2),
DipartCitta(D1 , Roma), DipartCitta(D2 , Cisterna)}

J1 = {Residenza(Luca, Avezzano), Residenza(Francesco, Latina),
Residenza(Luca, R1), Residenza(Francesco, R2),
ImpDipart(Luca, D1), ImpDipart(Francesco, D2),
DipartCitta(D1 , Roma), DipartCitta(D2 , Cisterna)}

J2 = {Residenza(Luca, Avezzano), Residenza(Francesco, Latina),
ImpDipart(Luca, D), ImpDipart(Francesco, D),
DipartCitta(D, Roma), DipartCitta(D, Cisterna)}

L'istanza J2 non è universale. In particolare, si può notare come non esista omomorfismo da J2 a J0. Difatti, la soluzione J2 contiene informazioni "extra" che non sono richieste dalle specifiche; in particolare, assume che i dipartimenti di Luca e Francesco siano gli stessi. Può essere invece notato come J0 e J1 siano soluzioni universali, in quanto hanno omomorfismi ad ogni soluzione (e di conseguenza anche tra di loro).

Sia (S, T, Σst , Σt) una configurazione di data exchange.

1. Se I è una istanza del source schema e J,J' sono soluzioni universali per I, allora J e J' sono omomorficamente equivalenti.
2. Assumiamo che Σst sia un insieme di tgds. Siano I, I' due istanze del source schema, J una soluzione universale per I, e J' una soluzione universale per I'. Allora Sol(I) $\subseteq$ Sol(I') se e solo se c'è un omomorfismo h: J'→ J. Di conseguenza, Sol(I) = Sol(I') se e solo se J e J' sono omomorficamente equivalenti.

Di conseguenza, le soluzioni universali sono uniche (con l'equivalenza omomorfica). Ogni soluzione universale accorpa completamente lo spazio delle soluzioni.

COMPUTING

Questa sezione analizza la problematica di cercare se una soluzione universale esista, e come trovarla nel caso in cui esista. In particolare, sarà mostrato come la classica procedura *chase* può essere applicata al Data Exchange. Il chase è un tool algoritmo indispensabile per ragionare sulle dipendenze, e in particolare per testare se un insieme di dipendenze implica logicamente un'altra dipendenza data. Per ogni chase "finito", se la procedura non fallisce restituisce una soluzione universale, altrimenti non esiste soluzione. Per un insieme arbitrario di dipendenze non è assicurata l'esistenza di un chase finito, per cui sarà introdotta la classe dei "*weakly acyclic sets of tgds*", per la quale è garantito che il chase termini in tempo polinomiale. Per tale insieme di dipendenze, sarà dimostrato che: (1) l'esistenza di una soluzione universale può essere controllata in tempo polinomiale, (2) una soluzione universale esiste se e solo se una soluzione esiste, e (3) una soluzione universale può essere prodotta in tempo polinomiale.

Procedura chase: La procedura per produrre una soluzione universale è intuitivamente la seguente: si comincia con un'istanza <I, ∅> che consiste nell'istanza I del source schema e un'istanza vuota per il target schema; quindi si applica il chase a <I, ∅> applicando le dipendenze in Σ*st* e Σ*t* fintantoché sono applicabili. Questa procedura può fallire e può anche non terminare, ma se termina è garantito che l'istanza risultante soddisfa tutte le dipendenze e che, per di più, è una soluzione universale.

Ora passiamo a definire i singoli *chase steps*. Similmente all'omomorfismo tra istanze, un omomorfismo da una formula congiuntiva $\varphi(\mathbf{x})$ a un'istanza J è un

mapping dalle variabili x a Const $\cup$ Var(J) tale che per ogni atomo $R(x1, \ldots, xn)$ di φ il fatto $R(h(x1), \ldots, h(xn))$ è in J.

Chase step: sia K un'istanza.

(tgd) Sia d una tgd $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$. Sia h un omomorfismo da $\varphi(\mathbf{x})$ a K tale che non esista un'estensione di h a un omomorfismo h' da $\phi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y})$ a K. Diciamo che d può essere applicato a K con omomorfismo h.

Sia K' l'unione di K con l'insieme dei fatti ottenuti: (a) estendendo h ad h' in modo tale che ad ogni variabile in y sia assegnato un nuovo labeled null; (b) prendendo l'immagine degli atomi di ψ sotto h'. Diciamo che il risultato dell'applicazione di d a K con h è K', e scriviamo $K \;\; d,h \rightarrow K'$.

(egd) Sia d un egd $\varphi(\mathbf{x}) \rightarrow (x1 = x2)$. Sia h un omomorfismo da $\varphi(\mathbf{x})$ a K tale che $h(x1) \; != \; h(x2)$. Diciamo che d può essere applicato a K con omomorfismo h. Distinguiamo ora due casi:

- Se sia h(x1) che h(x2) sono in Const, diciamo che il risultato dell'applicazione di d a K con h è "fallimento", e scriviamo $K \;\; d,h \rightarrow \perp$.
- Altrimenti, sia K' come K in cui identifichiamo h(x1) e h(x2) come segue: se uno è una costante, allora il labeled null è rimpiazzato dovunque dalla costante; se sono entrambi labeled nulls, allora ognuno è rimpiazzato dovunque dall'altro. Diciamo che il risultato dell'applicazione di d a K con h è K', e scriviamo $K \;\; d,h \rightarrow K'$.

Analizziamo ora i concetti di *sequenza chase* e *chase finito*.

- Una *sequenza chase* di K con Σ è una sequenza (finita o infinita) di chase steps
  $Ki \;\; di,hi \rightarrow Ki+1$, con i=0,1,...., con K=K0 e d *i* una dipendenza in Σ.

- Un *chase finito* di K con Σ è una sequenza chase finita $Ki \;\; di,hi \rightarrow Ki+1$, con

  $0 \leq i < m$, con il requisito che: (a) $Km = \perp$ oppure (b) non c'è dipendenza d *i* in Σ e non c'è omomorfismo h *i* tale che d *i* possa essere applicato a K*m* con h *i* . Diciamo che K*m* è il risultato del chase finito. Nel caso (a) abbiamo un chase finito che fallisce, nel caso (b) un chase finito che ha successo.

Nel data Exchange, per la natura delle dipendenze, ogni sequenza chase che comincia con $<I, \emptyset>$ non cambia o aggiunge tuple in I. Di conseguenza, se esiste un chase finito, allora J è una soluzione. Il prossimo teorema mostra inoltre come J sia anche una soluzione universale, e dimostra che il chase può essere utilizzato

per controllare l'esistenza di una soluzione.

TEOREMA: Si assuma una configurazione di Data Exchange in cui Σst consiste in tgds e Σt consiste in tgds e egds.

1. Sia <I,J> il risultato di un qualche chase finito di $<I, \emptyset>$ con $\Sigma st \cup \Sigma t$. Allora J è una soluzione universale.
2. Se esiste un qualche chase finito di $<I, \emptyset>$ che fallisce con $\Sigma st \cup \Sigma t$ allora non esiste soluzione.

DIMOSTRAZIONE

La dimostrazione del teorema fa uso della seguente proprietà del chase step:

Lemma: sia K1 $d,h \rightarrow K2$ un chase step dove K2!= $\perp$. Sia K un'istanza tale che: (i) K soddisfa d e (ii) esiste un omomorfismo $h1 : K1 \rightarrow K$. Allora esiste un omomorfismo $h2 : K2 \rightarrow K$.

Dimostrazione del lemma:

1. caso 1: d è un tgd $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$. Dalla definizione di chase step, $h : \varphi(\mathbf{x}) \rightarrow K1$ è un omomorfismo. Dal momento che K soddisfa d esiste un omomorfismo $h' : \varphi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y}) \rightarrow K$ tale che h' è un'estensione di $h1 \circ h$, ovvero h'(**x**)=h1(h(**x**)). Per ogni variabile y in **y** si denoti con Λy il labeled null che rimpiazza y nel chase step. Si definisce h2 su Var(K2) come segue: $h2(\Lambda) = h1(\Lambda)$, se $\Lambda \in$ Var(K1), e $h2(\Lambda y) = h'(y)$ per y in **y**. Bisogna mostrare che h2 è un omomorfismo da K2 a K, ovvero h2 mappa i fatti di K2 in fatti di K preservando i simboli di relazione. Per i fatti di K2 che sono anche in K1 è vero perché h1 è un omomorfismo. Sia T(**x0**, **y0**) un atomo arbitrario nella congiunzione ψ (da qui **x0** and **y0** contengono le variabili in **x** and **y**, rispettivamente). Allora K2 contiene, oltre ai fatti di K1, un fatto T(h(**x0**),**Λy0**). L'immagine sotto h2 di questo fatto è, per definizione di h2, il fatto T($h1(h($**x0**$))$, $h'($**y0**$)$). Dal momento che $h'($**x0**$) = h1(h($**x0**$))$, questo è lo stesso di T($h'($**x0**$)$, $h'($**y0**$)$) . Ma h' mappa omomorficamente tutti gli atomi di $\varphi \wedge \psi$, in particolare T(**x0**, **y0**), in fatti di K. Quindi, h2 è un omomorfismo.
2. caso 2: d è un egd $\varphi(\mathbf{x}) \rightarrow (x1 = x2)$. Come nel caso1, $h1 \circ h : \varphi(\mathbf{x}) \rightarrow K$ è un omomorfismo. Si faccia in modo che h2 sia h1. Dobbiamo assicurare che h1 è ancora un omomorfismo quando considerato da K2 a K. L'unica condizione per cui h1 possa NON essere un omomorfismo è che h1 non sia una funzione, mappando h(x1) e h(x2) in due differenti costanti o labeled null di K. Ma questo non è il caso, poiché K soddisfa d, e quindi $h1(h(x1)) = h1(h(x2))$.

La dimostrazione del teorema si basa su questo lemma e sull'osservazione che l' "identity mapping" è un omomorfismo da <I, ∅> a <I, J'> per ogni soluzione J'.

Dimostrazione parte 1: dalla definizione di chase è noto che <I,J> soddisfa $\Sigma st \cup \Sigma t$. Dal momento che Σt usa solo simboli di relazione target, segue che J soddisfa Σt . Sia J' una soluzione arbitraria. Allora, <I,J'> soddisfa $\Sigma st \cup \Sigma t$. In più, l'identity mapping id: <I, ∅>→ <I, J'> è un omomorfismo. Applicando il lemma ad ogni chase step, si ottiene un omomorfismo h:<I,J'>→ <I,J'>. In particolare, h è anche un omomorfismo da J a J'. Quindi, J è universale.

Dimostrazione parte 2: Sia <I,J> $d,h \rightarrow \perp$ l'ultimo chase step di un chase che fallisce. Allora d deve essere un egd di Σt, detto $\varphi(\mathbf{x}) \rightarrow (x1 = x2)$, e $h : \varphi(\mathbf{x}) \rightarrow J$ è un omomorfismo tale che h(x1) e h(x2) sono due costanti distinte c1 e c2. Si supponga che esista una soluzione J'. Secondo le stesse argomentazioni della part1, si sa che l'omomorfismo identità id:<I, ∅>→ <I, J'> implica, dal lemma, l'esistenza dell'omomorfismo g: :<I, J>→ <I, J'>. Quindi, $g \circ h : \varphi(\mathbf{x}) \rightarrow J'$ è un omomorfismo. Dal momento che si assume che J' soddisfi d, deve essere il caso in cui $g(h(x1)) = g(h(x2))$ e quindi $g(c1) = g(c2)$. Gli omomorfismi sono identità su Const, quindi c1=c2, che è una contraddizione. ■

Il prossimo è un esempio di un insieme ciclico di dipendenze di inclusione per il quale non esiste un chase finito, per il quale quindi non si può produrre una soluzione universale. Tuttavia, una soluzione esiste; ciò mostra la necessità di introdurre restrizioni nella classe delle dipendenze permesse nel target schema.

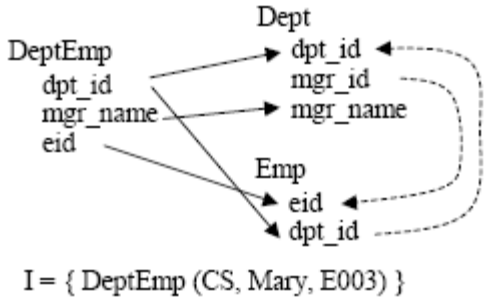
- *Esempio (chase infinito)*

Si consideri la configurazione di Data Exchange $(\mathbf{S},\mathbf{T},\Sigma st,\Sigma t)$ definita come segue. Il source schema S ha una relazione DeptEmp(dpt_id, mgr_name, eid) che mostra i dipartimenti con i loro manager e i loro impiegati. Il target schema T ha una relazione Dept(dpt_id, mgr_id, mgr_name) per i dipartimenti e i loro manager, e una relazione seprata per gli impiegati Emp(eid, dpt_id). Le dipendenze source-to-target sono:

$\Sigma st = \{ \text{DeptEmp}(d, n, e) \rightarrow \exists M.\text{Dept}(d, M, n) \wedge \text{Emp}(e, d) \}$

Mentre le dipendenze sul target sono:

$\Sigma t = \{ \text{Dept}(d, m, n) \rightarrow \exists D.\text{Emp}(m, D),$
$\text{Emp}(e, d) \rightarrow \exists M \exists N.\text{Dept}(d, M, N) \}$



Si assuma che l'istanza del source abbia una tupla in DeptEmp, per il dipartimento CS con manager Mary ed impiegato E003. Applicando il chase $< I, \emptyset >$ con Σst si ottiene l'istanza target:

$J1 = \{\text{Dept}(CS, M, Mary), \text{Emp}(E003, CS)\}$

dove M è un labeled null che istanzia la variabile quantificata esistenzialmente del tgd, e rappresenta il "manager id" sconosciuto di Mary. J1, però, non soddisfa Σt, per cui il chase non si ferma a J1. Il primo tgd in Σt richiede che M appaia in Emp come un id impiegato. Per cui, il chase aggiunge Emp(M,D) dove D è un labeled null che rappresenta il dipartimento sconosciuto in cui Mary è impiegata. In tal modo il secondo tgd diviene applicabile, ma si può notare come non esista chase finito. Soddisfare tutte le dipendenze richiederebbe costruire l'istanza infinita :

$J = \{\text{Dept}(CS, M, Mary), \text{Emp}(E003, CS), \text{Emp}(M, D), \text{Dept}(D, M', N'), \ldots\}$

D'altro canto, una soluzione finite esiste. Due soluzioni sono ad esempio:

$J' = \{\text{Dept}(CS, E003, Mary), \text{Emp}(E003, CS)\}$
$J'' = \{\text{Dept}(CS, M, Mary), \text{Emp}(E003, CS), \text{Emp}(M, CS)\}$

Tuttavia nè J' nè J'' sono universali: infatti non esiste omomorfismo tra di loro. Per cui ci si augura che nessuna delle due sia usata nel Data Exchange. J' infatti assume che il mgr_id di Mary sia uguale a E003, mentre J'' assume che il dipartimento in cui Mary è impiegata si lo stesso in cui Mary è manager. Tali assunzioni non sono conseguenza delle dipendenze e dell'istanza del source. Può essere mostrato che nessuna soluzione universale esiste per tale esempio.

In seguito saranno considerati insiemi di dipendenze per i quali ogni sequenza chase è garantita raggiungere la sua fine dopo un numero polinomiale di step (nella lunghezza dell'istanza di input). Per questi insieme cercare l'esistenza di una

soluzione, così come generare una soluzione universale, richiede tempo polinomiale.

I full tgds sono dei tgds senza variabili esistenzialmente quantificate ($\varphi_T(\mathbf{x}) \rightarrow \psi_T(\mathbf{x})$, dove $\varphi_T(\mathbf{x})$ e $\psi_T(\mathbf{x})$ sono congiunzioni di atomi target).

- *Esempio (full tgd)*
  $H(x,z) \wedge H(z,y) \rightarrow H(x,y) \wedge C(z)$

Si può notare come i full tgds esprimano contenimento tra join relazionali. E' stato provato che ogni sequenza chase con un insieme Σ di full tgds ha lunghezza finita. In più ogni chase ha lo stesso risultato. Inoltre, ogni insieme di egds può essere aggiunto a Σ senza influenzare questo risultato. Tuttavia, nonostante queste buone proprietà, i full tgds non sono molto utili in pratica, poiché la maggior parte delle dipendenze che occorrono negli schemi reali sono "non-full", come ad esempio le foreign key o le dipendenze di inclusione.

E' ben noto che, in generale, il chase con le dipendenze di inclusione può non terminare. Gli *insiemi aciclici di dipendenze di inclusione* sono un caso speciale per cui ogni sequenza chase ha una lunghezza polinomiale nella lunghezza dell'istanza di input. Queste dipendenze possono essere descritte definendo un grafo diretto nel quale i nodi sono i simboli di relazione, e tale che esista un arco da R a S ogni volta che c'è una dipendenza di inclusione da R ad S. L'insieme è detto aciclico se non esistono cicli nel grafo. La nozione di *weakly acyclic sets of tgds* include sia gli insiemi di full tgds che gli insiemi aciclici di dipendenze di inclusione. Informalmente, un insieme di tgds è "debolmente aciclico" se non permette la creazione a cascata di labeled nulls durante il chase.

Weakly acyclic set of tgds: Sia Σ be un insieme di tgds su uno schema fissato. Si costruisce un grafo diretto, chiamato *grafo delle dipendenze*, come segue:

(1) c'è un nodo per ogni coppia (R,A) in cui R è un simbolo di relazione e A un attributo di R; la coppia (R,A) è detta *posizione*;

(2) si aggiungono gli archi come segue: per ogni tgd $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ in Σ e per ogni x che occorre in ψ:

- Per ogni occorrenza di x in φ in posizione (R,Ai):
  - (a) per ogni occorrenza di x in ψ in posizione (S,Bj), si aggiunge un arco $(R,Ai) \rightarrow (S,Bj)$ (se non esiste già)
  - (b) in aggiunta, per ogni variabile y esistenzialmente quantificata e per ogni occorrenza di y in ψ in posizione (T,Ck), si aggiunga un arco speciale $(R,Ai) \ast\rightarrow (T,Ck)$ (se non esiste già)

Da notare che possono esserci due archi nella stessa direzione tra due nodi, se esattamente uno dei due archi è speciale. Quindi Σ è debolmente aciclico se il grafo delle dipendenze non ha cicli passando per un arco speciale. Intuitivamente, la parte (a) tiene conto del fatto che un valore può essere propagato dalla posizione (R,Ai) alla posizione (S,Bj) durante il chase. In più la parte (b) tiene conto del fatto che la propagazione di un valore in (S,Bj) crea anche un labeled null in ogni posizione che ha una variabile esistenzialmente qualificata. Se un ciclo passa per un arco speciale, un labeled null che appare in una certa posizione durante il chase può comportare la creazione di un altro labeled null, nella stessa posizione, in un chase step successivo. Questo processo potrebbe continuare per sempre. Sono invece possibili i cicli fintantoché non includono archi speciali.

- *Esempio (weakly acyclic set of tgds)*

Riprendiamo l'esempio precedentemente illustrato, e così definito:

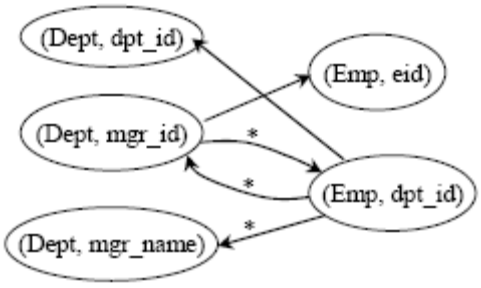
S : DeptEmp(dpt_id,mgr_name,eid)

T : Dept(dpt_id, mgr_id, mgr_name) ; Emp(eid, dpt_id)

$\Sigma st = \{$ DeptEmp(d, n, e) $\rightarrow \exists M.$Dept(d,M, n) $\wedge$ Emp(e, d) $\}$

$\Sigma t = \{$ Dept(d, m, n) $\rightarrow \exists D.$Emp(m,D),
Emp(e, d) $\rightarrow \exists M \exists N.$Dept($d,M,N$) $\}$

Il grafo delle dipendenze di Σt è il seguente:

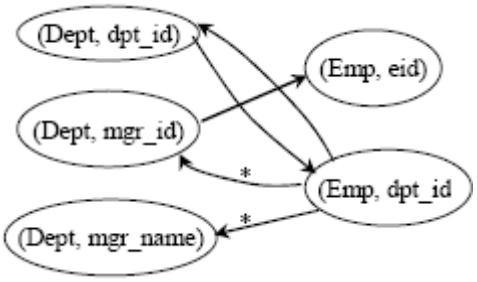


Il grafo contiene un ciclo con due archi speciali. Per cui Σt non è debolmente aciclico e quindi un chase finito può non esistere.

D'altro canto, si supponga che ogni manager di un dipartimento sia impiegato nello stesso dipartimento. In tal caso consideriamo il nuovo insieme $\Sigma' t$ così definito:

$\Sigma't = \{ \text{Dept}(d, m, n) \rightarrow \text{Emp}(m, d),$
$\text{Emp}(e, d) \rightarrow \exists M \exists N.\text{Dept}(d, M, N) \}$

Per questo nuovo insieme, il grafo delle dipendenze è il seguente:



Questo grafo non ha cicli che passano per archi speciali. Di conseguenza, $\Sigma't$ è debolmente aciclico, perciò ogni sequenza chase è finita. Ad esempio, il chase di J1 con $\Sigma't$ termina con il risultato J'' ={Dept(*CS*,*M*,*Mary*), Emp(*E*003, *CS*), Emp(*M*,*CS*)}. Questo perché si è imposto che i manager sono impiegati nel dipartimento dove sono manager.

Sarà illustrato ora il maggior risultato che riguarda i weakly acyclic sets of tgds

Teorema: *Sia* Σ *l'unione di un weakly acyclic set of tgds con un insieme di egds, e sia K un'istanza. Allora esiste un valore polinomiale sulla taglia di K che limita la lunghezza di ogni sequenza chase di K con* Σ**.**

Dimostrazione

Mettiamoci nel caso in cui $\sum$ non contenga egds, in realtà considerando anche le egds non cambia in sostanza la dimostrazione. Per ogni nodo (R, A) nel grafo di dipendenza di $\sum$ definiamo:

- ***incoming path:*** ogni percorso che finisce in (R, A);

- ***rank:*** il massimo numero di archi speciali su ogni incoming path e lo denoteremo con rank(R, A).

Da ipotesi $\sum$ è weakly acyclic, quindi non contiene cicli sugli archi speciali e di conseguenze rank(R, A) è finito.

Sia ***r*** il massimo di rank(R, A) per ogni nodo (R, A) e sia ***p*** il numero di nodi nello schema. Chiaramente p è una costante e r limitato superiormente da p.

Ora partizioniamo i nodi del grafo di dipendenza in sottoinsieme ***N0, N1, …., Nr*** dove Ni è l'insieme di tutti i nodi con rank = i.

Sia ***n*** il numero totale di valori distinti che appartengono all'istanza K. Sia K' qualsiasi istanza ottenuta da K dopo qualche arbitraria sequenza di chase.

Proviamo per induzione il seguente:

per ogni i esiste un polinomio Qi, tale che il numero di valori distinti che occorrono in tutti i nodi (R,A) di Ni, in Ki, è al più Qi(n).

Passo Base: se (R, A) è un nodo di N0, significa che non ci sono incoming paths con archi speciali. Nessun valore viene creato al nodo (R, A) durante il chase e quindi i valori appartenenti al nodo (R, A) in K' sono tra gli n valori dell'istanza originale K. Questo è vero per tutti i nodi che sono in N0 e quindi consideriamo ***Q0(n) = n***;

Passo Induttivo: i primi valori che sono in un nodo appartenente a Ni, in K', sono quelli che già occorrono nei stessi nodi in K. Il numero di tali valori è al più n.

Un valore può occorrere in un nodo appartenente a Ni, in K' per due motivi:

1. È stato copiato da qualche nodo in Nj con $j \neq i$, durante un chase step.
2. È stato generato come nuovo valore (labeled null) durante un chase step.

Cominciamo a determinare quanti valori possono essere generati (2).

Sia (R, A) un nodo appartenente a Ni. Un nuovo valore può essere generato durante un chase step solo a causa di archi speciali ma ogni arco speciale che entra in (R, A) deve iniziare in un nodo appartenente a N0 U …. U N_{i-1}.

Applicando le ipotesi induttive, il numero di valori distinti che possono esistere in tutti i nodi in N0 U …. U N_{i-1} è limitato da $P(n) = Q0(n) + \ldots. + Q_{i-1}(n)$.

Sia ***d*** il numero massimo di archi speciali che entrano in un nodo, su tutti i nodi dello schema. Per ogni distinta d-tupla di valori di N0 U …. U N_{i-1} e per ogni dipendenza in ∑ c'è al più un valore nuovo che può essere generato al nodo (R, A).

Quindi il numero totale di nuovi valori che possono essere generati in (R, A) è al più $(P(n))^d$ x D, dove D è il numero di dipendenze in ∑. Se consideriamo tutti i nodi in Ni il numero totale di valori che possono essere generati è al più ***G(n) = pi x $(P(n))^d$ x D***, dove pi è il numero di nodi in Ni. Considerando che lo schema e ∑ sono fissi, G è polinomiale.

Ora andiamo a considerare il numero di valori distinti che possono essere copiati da un nodo di Nj a un nodo di Ni, con i ≠ j (2). Tale copia può accedere solo se ci sono archi non speciali dai nodi in Nj ai nodi in Ni con j ≠ i. Osserviamo che tali archi non speciali possono derivare solo da nodi in N0 U …. U $N_{i-1,}$ quindi per tutte le j < i, altrimenti si cadrebbe in una contraddizione. Dunque il numero di valori distinti che possono essere copiati in nodi di Ni è limitato dal numero totale di valori in N0 U …. U N_{i-1} che è ***P(n)***.

Ricapitolando ***Qi(n) = n + G(n) + P(n)*** che è polinomiale e quindi la proposizione è dimostrata.

Osserviamo che i è limitato da massimo rank r, che è costante. Quindi esiste un polinomio fissato Q tale che il numero di valori distinti che può esistere, un tutti i nodi, in K', è limitato da Q(n). In particolare il numero di valori distinti che può esistere, in un singolo nodo, in K' è anche limitato da Q(n). Il numero totale di tuple che possono esistere in K' è al più $(Q(n))^p$ che è polinomiale dato che p è una costante. Dato che ogni chase step con un tgds aggiunge almeno qualche tupla a K', segue che la lunghezza di ogni chase sequence è al più $\boldsymbol{(Q(n))^p}$. ■

Corollario: Si assuma una configurazione di Data Exchange in cui Σst sia un insieme di tgds e Σt l'unione di un weakly acyclic set of tgds con un insieme di egds. L'esistenza di una soluzione può essere controllata in tempo polinomiale. Se la soluzione esiste, allora una soluzione universale può essere trovata in tempo polinomiale.

IL CORE NEL DATA EXCHANGE

Nel paragrafo precedente si è approfondito il discorso relativo alle universal solutions e abbiamo visto come queste siano le soluzioni preferite al problema di data exchange. Allo stesso tempo bisogna precisare che tali soluzioni potrebbero essere diverse, in particolare sappiamo che le soluzioni universali sono tra loro omomorficamente equivalenti una all'altra ma non necessariamente isomorfe. Facciamo un esempio. Consideriamo un data exchange setting nel quale lo schema sorgente consiste di una relazione binaria Padre che associa ad ogni genitore di sesso maschile il proprio figlio e lo schema target di una relazione binaria Parente che indica che il rapporto di parentela. Sia il seguente schema mapping $\sum_{st}$: E(x,y) → ∃ z (H(x,z) ∧ H(z,y)) mentre $\sum_{t}$ = Ø. Sia l'istanza sorgente

I = { E(a,b) }. Soluzioni universali a questo problema possono essere molteplici, elenchiamone alcune:

J1 = {H(a,X), H(X,b)}

J2 = { H(a,X), H(X,b), H(a,Y), H(Y,b)}

J3 = {H(a,X), H(X,b), H(Y,Y)}.

Come si vede queste sono non isomorfe tra loro. A questo punto ci poniamo un'altra domanda importante: quale soluzione universale prendere in considerazione? Esiste la soluzione migliore anche tra le soluzioni universali? Prima abbiamo considerato il problema delle soluzioni universali, facendoci guidare dall'approccio *"small is beautiful"* ora vogliamo scegliere la più piccola soluzione universale. In riferimento all'esempio potremmo scegliere la soluzione J1 che è la più piccola e la più economica soluzione da realizzare. Questa soluzione è "unique up to isomorphism" ed è proprio il *core* di tutte le soluzioni universali. Nel seguito daremo la definizione di core, e tratteremo le sue caratteristiche in termini computazionali e di complessità trattando in dettaglio i problemi di core recognition e core identification. Questo argomento viene largamente trattato nei rif [2: Kol, Schema mappings, data exchange and metadata management], [7: Fag, Data ecxhange: getting to the core] e [10: Got, Data Exchange: Computing Cores in Polynomial Time]. In particolare in quest'ultimo documento si introduce un nuovo risultato importantissimo riguardo la trattabilità del problema del calcolo del Core.

Core. Il concetto di core ha origine dalla teoria dei grafi e successivamente espanso alla teoria dei database dimostrando che poteva essere efficacemente utilizzato nel processamento delle conjunctive queries. Ora diamo la definizione di core:

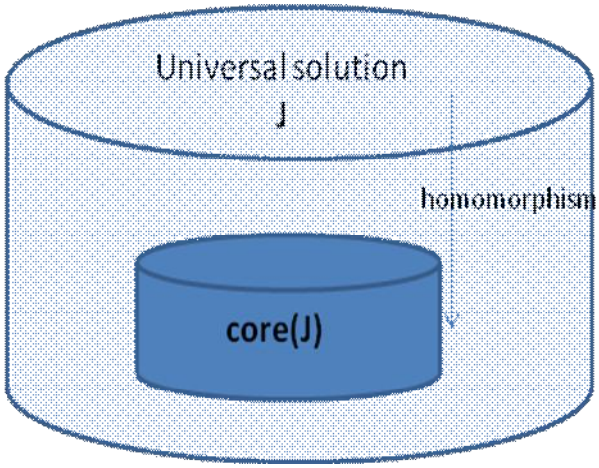
Def. Sia G = (N, A) un grafo. Un sottografo G' = (N', A') è il core di G se:

1. ∃ un omomorfismo da G a G'.
2. ∄ un omomorfismo da G' a qualche altro sottografo proprio di G'.

Ne deduciamo che G è un core se e solo se è il core di se stesso. Ora diamo un po' di risultati ottenuti da diverse ricerche.

Prop. Sia G un grafo finito:

1. G ha un core.
2. Tutti i core di G sono isomorfi, per questo quando ci riferiamo al core di G possiamo denotarlo con core(G).
3. ∃ un omomorfismo h da core(G) a G tale che h(d) = d ∀ d ∈ G.
4. G è omomorficamente equivalente al suo core. Quindi due grafi sono omomorficamente equivalenti se e solo se i loro core sono isomorfi.



E' da notare che stiamo parlando di grafi finiti. La definizione vista e le proprietà elencate possono essere utilizzate interamente nel nostro contesto, quindi su core di istanze target, con una piccola aggiunta, ossia quando andiamo a considerare omomorfismi tra istanze target tali che h(c)=c ∀ c ∈ Const.

Le soluzioni universali Ji per un'istanza sorgente I sono omomorficamente equivalenti ma, come abbiamo visto, non necessariamente sono isomorfe. Dalle proprietà di cui sopra si evince che i core delle varie Ji sono isomorfi e quindi tutte le soluzioni universali per I hanno core isomorfi. Quindi se J è una soluzione universale per I e core(J) è una soluzione per I, abbiamo che core(J) è anche una

soluzione universale per I, quindi J e core(J) sono omomorficamente equivalenti. Diamo la seguente proposizione:

Prop. Sia $(S, T, \sum_{st}, \sum_t)$ un data exchange setting con $\sum_{st}$ un insieme di s-t tgds e $\sum_t$ un insieme di target egds e tgds:

1. Se I è un'istanza sorgente e J una soluzione per I, core(J) è una soluzione per I.
2. Se I è un'istanza sorgente e J una soluzione universale per I, anche core(J) è una soluzione universale per I.
3. Se I è un'istanza sorgente per la quale esiste una soluzione universale, c'è una soluzione universale J_0 tale che:
    a. J_0 è il core ed è isomorfo al core di ogni soluzione universale per I.
    b. Se J è una soluzione universale per I, c'è un omomorfismo one-to-one da J_0 a J. Ne consegue $|J_0| <= |J|$ con $|J_0|$ e $|J|$ le dimensioni di J_0 e J; quindi J_0 è la soluzione universale più piccola.

Il core di una soluzione universale è la soluzione universale preferita per essere materializzata in data exchange. Questo fa sorgere la questione relativa al calcolo di tale soluzione. Nella precedente sezione abbiamo studiato in determinate condizioni una soluzione universale può essere calcolata tramite chase in tempo polinomiale. Il risultato dell'algoritmo mostrato potrebbe comunque non essere un core di una soluzione universale. Nella prossima sezione illustreremo la complessità di calcolare core di arbitrarie istanze e successivamente il core di soluzioni universali in data exchange.

***Complexity*.** Il problema di identificare un core dato un grafo, o più specificatamente una struttura finita relazionale, nella sua generalità è intrattabile. In data exchange il nostro obiettivo è calcolare il core di una soluzione universale piuttosto che di un'istanza arbitraria. Quindi l'intrattabilità di trattare il core di un'istanza arbitraria non significa l'intrattabilità di calcolare il core di una soluzione universale. Riferiamo la nostra attenzione quindi a due problemi correlati. Il primo si chiama CORE IDENTIFICATION, sia dato un grafo G ed un suo sottografo G' verificare che core(G)=G'. Questo problema è DP-complete (nel seguito spiegheremo questa classe di problemi). Il secondo problema, CORE RECOGNITION, tratta del seguente problema di decisione: dato un grafo G, è G un core? Questo problema è in coNP-complete. Questo risultato è stato stabilito mostrando una riduzione tempo polinomiale da Non-3-Colorability su grafi di cardinalità almeno 7.

Tornando al primo problema, la classe DP è la classe dei problemi di decisione che possono essere scritti come un'intersezione dei problemi NP e coNP. Questa classe è stata introdotta da Papadimitriou e Yannakakis nel 1982 che scoprirono diversi problemi DP-complete (SAT/UNSAT). Tale classe quindi contiene sia NP che coNP come sottoclassi, inoltre ogni problema DP-complete è sia NP-hard che coNP-hard. In altre parole stabilire che un problema è DP-complete indica che questo è intrattabile e 'più intrattabile' rispetto un NP-complete.

Consideriamo ora l'intrattabilità del problema di CORE RECOGNITION. Chandra e Merlin [1977] studiarono che un grafo è 3-colorabile se e solo se core(G $\times$ K3) =

K3. Ne segue che non c'è un algoritmo polinomiale per calcolare il core.

Resta ora il problema di trovare determinati classi di data exchange setting per i quali calcolare il core è un problema trattabile. Nel seguito illustreremo dei risultati importanti che definiscono degli algoritmi polinomiali per calcolare il core per i seguenti data exchange setting $(S, T, \sum_{st}, \sum_{t})$:

- $\sum_{st}$ un insieme di s-t tgds e $\sum_{t}$ un insieme di egds;

- $\sum_{st}$ un insieme di s-t tgds e $\sum_{t}$ un insieme di weakly acyclic tgds con arbitrarie egds.

Computing core. Fino ad ora abbiamo detto che calcolare il core di istanze arbitrarie non è trattabile. Questo non significa che non esistono algoritmi che, sotto determinate condizioni, possano calcolare il core di soluzioni universale in tempo polinomiale. Dapprima illustreremo un risultato recente che interessa setting nei quali il target schema è un insieme di weakly acyclic tgds con aribitrarie egds poi descriveremo un data exchange settings $(S, T, \sum_{st}, \sum_{t})$ tali che $\sum_{st}$ sia un insieme di s-t tgds e $\sum_{t}$ sia un insieme di egds. Mostriamo quindi ora un problema aperto a cui da poco si è dato una risposta circa la trattabilità del problema.

Dopo aver raggiunto dei risultati concreti definendo degli algoritmi tempo polinomiale circa la trattabilità di classi di data exchange setting con target schema composti solamente da egds è rimasto aperto per diversi anni il problema di individuare classi di dipendenze più larghe e significative per le quali il problema dell'individuazione del core fosse trattabile. A questo proposito è stato definito un particolare problema che è rimasto aperto fino a poco tempo fa. Dato un problema di data exchange con $\sum_{st}$ un insieme di s-t tgds e $\sum_{t}$ un insieme di weakly acyclic tgds con arbitrarie egds può il core di una soluzione universale essere calcolato in tempo polinomiale? Il principale risultato mostrato in [10]

afferma che il core può essere calcolato in tempo polinomiale. Nel riferimento [10] Gottlob e Nash evidenziamo questo risultato e definiscono un algoritmo per computarlo.

Illustriamo ora due algoritmi tempo polinomiale che calcolano il core in data exchange setting tali che $\sum_{st}$ sia un insieme di s-t tgds e $\sum_{t}$ sia un insieme di egds: un algoritmo concettualmente più semplice, Greedy ed un altro più complicato ma che offre qualche vantaggio di cui parleremo, Blocks. In questo sezione introdurremo l'algoritmo Greedy e successivamente studieremo l'algoritmo Blocks prima con $\sum_{t} = \emptyset$ e poi per $\sum_{t}$ con egds.

Algoritmo ***Greedy***. Intuitivamente questo algoritmo controlla prima se una soluzione esiste ed in caso affermativo calcola il core di una soluzione universale eliminando successivamente tuple da una soluzione universale che a ogni passo deve soddisfare le s-t tgds in $\sum_{st}$.

Input: sia un'istanza sorgente I, sia un data exchange setting $(S, T, \sum_{st}, \sum_{t})$ con S schema sorgente, T schema target, $\sum_{st}$ un insieme di s-t tgds e $\sum_{t}$ un insieme di target egds.

Output: il core di una soluzione universale per I se esiste, altrimenti 'failure'.

Begin

1. Chase I con $\sum_{st}$ in maniera tale da produrre un'istanza J in un data exchange setting $(S, T, \sum_{st})$.
2. Chase J con $\sum_{t}$ in maniera tale da produrre una soluzione universale J'; se tale passo fallisce restituisci 'failure'.
3. Poni J* = J'.
4. While ($\exists$ R(t) $\in$ J* tale che (I, J*- { R(t) }) soddisfa $\sum_{st}$))

   J* = J* \ {R(t)}

5. Return J*.

Diamo il seguente teorema:

Th. Sia un data exchange setting $(S, T, \sum_{st}, \sum_{t})$ con S schema sorgente, T schema target, $\sum_{st}$ un insieme di s-t tgds e $\sum_{t}$ un insieme di target egds. Allora l'algoritmo definito precedentemente è corretto e impiega tempo polinomiale per verificare

l'esistenza della soluzione e (se esiste la soluzione) calcolare il core di una soluzione universale.

Questo è un risultato molto importante, facciamo alcune osservazioni.

Questo algoritmo e la sua correttezza dipendono chiaramente dal rispetto delle sue ipotesi, quindi le assunzioni in $\sum_t$ devono essere solo egds.

Inoltre il passo 4 può essere costruito in tempo polinomiale per produrre il core delle soluzioni universali, data un'istanza sorgente e un'arbitraria soluzione universale. I primi due passi determinano in tempo polinomiale nella dimensione dell'istanza sorgente I una soluzione universale oppure 'failure', ossia la non esistenza delle soluzioni. Quindi l'algoritmo impiega tempo polinomiale nella dimensione di I.

Sebbene questo algoritmo è meno complicato del seguente, richiede che l'istanza sorgente sia disponibile durante l'intera esecuzione. Come detto agli inizi del documento, ci sono situazioni in cui I potrebbe non essere più disponibile dopo aver prodotto una soluzione universale. Nasce quindi di l'idea di utilizzare un algoritmo che operi in tempo polinomiale utilizzando solo una soluzione universale. Quello che descriviamo di seguito affronta e risolve tale situazione.

Algoritmo ***BLOCKS*** *con* $\sum_t = \emptyset$. Prima di descrivere il funzionamento, introduciamo dei concetti essenziali per comprendere al meglio l'algoritmo. Sia K un'istanza arbitraria i quali elementi sono costanti $\in$ Const e nulli $\in$ Var. Il grafo di

Gaifman dei nulli di K è un grafo indiretto nel quale: i nodi sono i nulli di K ed esiste un arco tra due nulli ogni volta che i nulli sono adiacenti in K. Due elementi di K sono adiacenti se esiste qualche tupla in qualche relazione di K nella quale occorrono i due elementi. Diamo ora la definizione di endomorfismo che risulterà essenziale poi.

Sia h un omomorfismo di K. Se h(K) è una sottoistanza di K, chiameremo h endomorfismo di K. Un endomorfismo di K è utilizzabile se h(K) $\neq$ K, ossia h(K) è

una sottoistanza proprio di K. Come detto consideriamo data exchange setting senza vincoli sul target. Inoltre considereremo un'istanza di sorgente I e una soluzione universale J ottenuta applicando chase con $\sum_{st}$. Il nostro scopo è calcolare il core(J) che è una sottoistanza C di J tale che C = h(J) per qualche endomorfismo h e non esiste una sottoistanza propria di di C con la stessa proprietà. L'idea è mostrare come tali endomorfismi h possono essere trovati dalla composizioni di sequenze di endomorfismi locali i quali possono essere

trovati in tempo polinomiale. Ma cosa rappresenta il concetto di locale? Spieghiamolo con la seguente definizione.

Siano K e K' due istanze tali che i nulli di K sono un sottoinsieme dei nulli di K', ossia Var(K) $\subseteq$ Var(K'). Sia h un endomorfismo di K' e B un blocco di nulli di K.

Diciamo che h è K-local per B se h(x) = x per x $\notin$ B. Diremo che h è K-local se è K-

local per qualche blocco B di K.

Lem. Infine, sia J' una sottoistanza della soluzione universale J, allora se esiste un endomorfismo utilizzabile di J', esiste un J-local endomorfismo utilizzabile di J'.

Ora possiamo descrivere l'algoritmo.

Input: istanza sorgente I e data exchange setting (S, T, $\sum_t$).

Output: core di una soluzione universale.

Begin:

1. Calcola la soluzione universale J da (I, $\emptyset$) ottenuta dal chase con $\sum_t$.
2. Calcola il blocco di J e poni J' = J.
3. Controlla se esiste un J-local endomorfismo utilizzabile h di J'. Se non esiste ritorna J'.
4. Poni J' = h(J') e torna al passo 3. End.

Sia dato il seguente teorema:

Sia un data exchange setting (S, T, $\sum_{st}$), allora l'algoritmo descritto sopra è corretto e computa in tempo polinomiale il core delle soluzioni universale.

Ora analizziamo l'algoritmo Blocks per setting con target schema contenenti solo egds.

*Algoritmo **Blocks** con $\sum_t$ di egds*. Assumiamo di avere un data exchange setting (S, T, $\sum_{st}$, $\sum_t$) con $\sum_t$ un insieme di egds. Assumiamo di avere anche un'istanza sorgente I. L'idea di tale algoritmo chiaramente è la stessa di quello precedente con piccoli accorgimenti. Si calcola tramite chase prima la soluzione preuniversale (ossia quella ottenuta tramite chase con I con $\sum_{st}$) e poi la soluzione unversale J.

Successivamente andiamo a determinare il core(J)=C, ossia una sottoistanza di J tale che C=h(J) per un endomorfismo h e tale che non esistono sottoistanze non esiste un sottoistanza propria di C con la stessa proprietà. Ora diamo un'estensione del lemma cha abbiamo descritto in precedenza. Nelle stesse ipotesi di cui sopra, sia J una soluzione preuniversale e J'' un'immagine endomorfica di una soluzione universale J' allora se esiste un endomorfismo di J'' utilizzabile (abbiamo visto sopra come significa), esiste un endomorfismo J-local utilizzabile di J''. Descriviamo di seguito l'algoritmo che ritorna il core di una soluzione universale o 'failure' se non esistono soluzione data un'istanza sorgente I.

Input: siano un data exchange setting (S, T, $\sum_{st}$, $\sum_{t}$) come descritto sopra e un'istanza sorgente I.

Output: core di una soluzione universale o 'failure' se non esistono soluzioni.

Begin

1. Calcola la soluzione preuniversale J da (I,) tramite chase con $\sum_{st}$.
2. Calcola il block di J e produci una soluzione universale J' tramite chase J con $\sum_{t}$. Se chase fallisce torna 'failure', altrimenti poni J'' = J'.
3. Verifica se esiste un endomorfismo J-local utilizzabile h di J''. Se non esiste ritorna J''.
4. Poni J''=h(J'') e torna al passo 3.

Diamo ora il seguente teorema.

L'algoritmo descritto sopra è corretto e computa il core di una soluzione universale in tempo polinomiale.

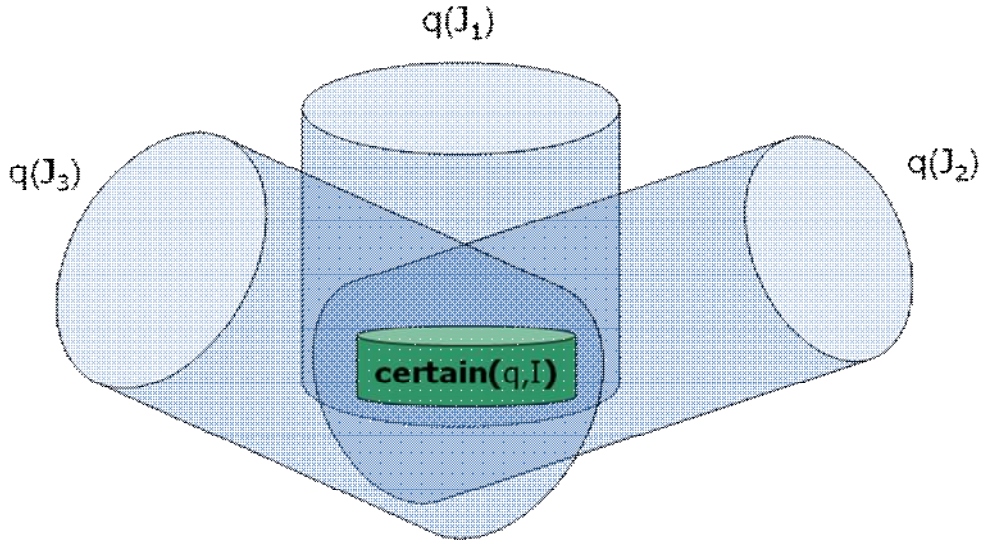
Estensioni. Finiamo di trattare di trattare questo argomento ponendoci due interrogativi: esistono classi più estesi di asserzioni nel target schema per le quali il problema di calcolare il core impiega tempo polinomiale? Infine possiamo ottenere il core tramite chase?

Target schema più ricchi. In questo senso un risultato importante fu ottenuto da Gottlob che ottenne alcuni risultati sullo studio del calcolo del core in data exchange. In particolare stabilì che usando un'estensione dell'algoritmo Blocks si può ottenere il core in tempo polinomiale per data exchange setting (S, T, $\sum_{st,}\sum_{t}$) con $\sum_{t}$ un insieme di egds e full-tgds. Rimane un problema aperto la trattabilità di setting in cui il target schema è un'unione di egds con weakly acyclic tgds.

Core tramite chase. Tramite di ottengono soluzioni universali ma non è detto che tali soluzioni siano anche il core. Si può vedere tramite esempi che in alcuni casi si può ottenere il core a seconda dell'ordine in cui si considerano le dipendenze e quindi in che ordine si applicano i passi del metodo chase. In altri casi invece applicando tale metodo non si può ottenere un core. Due esempi che esplicitano tali due situazioni sono riportate in [7: Fag., Data exchange: getting to the core].

QUERY ANSWERING

Dal momento che possono esistere infinite soluzioni J per una data istanza source in un problema di Data Exchange, nel caso in cui una query q fosse valutata su differenti soluzioni potrebbe dare risposte differenti. Un concetto simile viene incontrato nello studio dei database incompleti, affrontato introducendo il concetto di *risposte certe (certain answers)*, adottate come semantica anche nel query answering nell'information integration. Le query non sono poste su un singolo database, ma sull'insieme di tutti i possibili database in certi contesti. Per definizione, le risposte certe di una query q sono le tuple ottenute come intersezione di tutte le q(J), per tutti gli J che soddisfano la specifica iniziale. Nel Data Exchange, il concetto di "possibili database" corrisponde alle soluzioni per I.



Se I è una istanza source, allora le *risposte certe* di q su I rispetto a M = (**S**,**T**, Σ) è l'insieme $certain_M(q, I) = \bigcap \{q(J) : J \in SOL(M, I)\}$.

Computare le risposte certe è il seguente problema decisionale: data un'istanza source I e una tupla t di constanti presa da I, t appartiene a $certain_M(q, I)$? Si nota come valutare le risposte certe può prevedere il calcolo dell'intersezione di infiniti insiemi. Nell'information integration l'approccio prevede di riscrivere le query sul target in query sulle sorgenti; nel data Exchange, sarebbe più opportuno avvantaggiarsi della soluzione materializzata ed usarla per ottenere le risposte certe delle query sul target. Sarà mostrato come le risposte certe a *unioni di query congiuntive* possono essere ottenute usando una soluzione universale. Si ricorda che una query congiuntiva è una formula del primo ordine nella forma $\exists \mathbf{w}\chi(\mathbf{x},\mathbf{w})$, dove $\chi(\mathbf{x},\mathbf{w})$ è una congiunzione di atomi.

TEOREMA: Si assuma che $M = (\mathbf{S}, \mathbf{T}, \Sigma st \cup \Sigma t)$ sia uno schema mapping tale che Σst sia un insieme di tgds source-to-target e Σt sia un insieme di target egds e target tgds. Sia q un unione di query congiuntive sullo scherma target T.

1. Se I è un'istanza source e J è una soluzione universale per I, allora $certain_M(q, I) = q(J)\downarrow$, dove $q(J)\downarrow$ è l'insieme di tutte le tuple di q(J) senza null, ovvero tutte le tuple t in q(J) tali che ogni valore in t sia una costante di Const.
2. Sia I un'istanza source e J una soluzione tale che per ogni query congiuntiva q su T, si ha che certain(q, I) = q(J)↓. Allora J è universale.

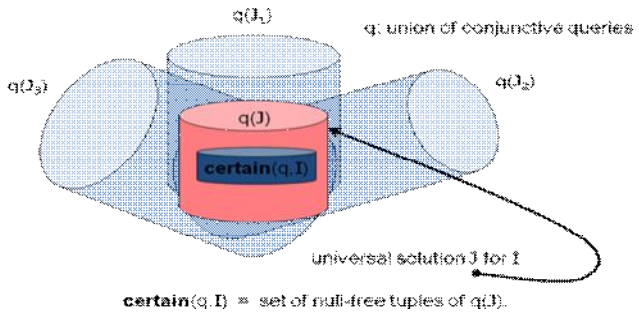
Si assuma inoltre che Σt sia l'unione di un insieme di target egds con un weakly acyclic set di target tgds. Allora esiste un algoritmo polinomiale per computare le risposte certe di q.

DIMOSTRAZIONE

1. Sia q una query di arità k che è l'unione di query congiuntive e sia t una k-tupla di costanti dall'istanza source I. Se t appartiene a certain(q,I), allora t appartiene a q(J), fintantoché J è una soluzione.
   Assumiamo ora che t appartenga a $q(J)\downarrow$. Allora t consiste solamente di costanti. Inoltre esistono un termine $\varphi(\mathbf{x})$ in q e un omomorfismo $g : \varphi(\mathbf{x}) \rightarrow J$ tale che $g(\mathbf{x}) = t$. Sia J' una soluzione arbitraria. Fintantochè J è una soluzione universale, c'è un omomorfismo $h : J \rightarrow J'$. Allora $h \circ g$ è un omomorfismo da $\varphi(\mathbf{x})$ a J'. Gli omomorfismi sono identità sulle costanti, per cui $h(g(\mathbf{x})) = h(t) = t$. Quindi t appartiene a q(J').
2. Sia q^j la query congiuntiva canonica associata a J (ad esempio, la query congiuntiva booleana ottenuta prendendo la congiunzione di tutti i fatti di J nei quali i labeled null sono sostituiti da variabili esistenzialmente quantificate). Si sa che certain(q^j, I) = q^j(J) ↓ = q^j(J), dove la prima uguaglianza deriva dall'assunzione su J, e la seconda deriva dal fatto che q^j è una query booleana. Quindi finchè q^j(J) = true, si ha che certain(q^j,I) = true. Inoltre, se J' è una soluzione arbitraria, si ha che q^j(J') = true. Questo implica l'esistenza di un omomorfismo $h : J \rightarrow J'$. Quindi J è universale.

Quindi, l'intero processo di computazione delle risposte certe richiede tempo polinomiale:

1. Trovare una soluzione universale J in tempo polinomiale
2. Valutare q(J) e rimuovere le tuple con dei null



- *Esempio (certain answers – query congiuntive)*

Sia M uno schema mapping tale che:

$\Sigma st = \{E(x, y) \rightarrow (\exists z)(H(x, z) \wedge H(z, y))\}$
$\Sigma t = \emptyset.$

Sia I l'istanza source che consiste semplicemente nel fatto E(1,2).
Sia q(x) la query congiuntiva $\exists w H(x,w)$. E' facile verificare che certain(q,I) = {1}. Prendiamo in considerazione la seguente soluzione universale per I:

$J = \{H(1, u), H(u, 2)\}$

Si può notare che q(J) = {1,u}, quindi eliminando i valori nulli si ottiene $q(J)\downarrow = \{1\}$ = certain(q,I), come ipotizzato dal teorema.

Le *query congiuntive con disuguaglianze* sono una delle estensioni più studiate delle query congiuntive. Una query congiuntiva con disuguaglianze è una formula del primo ordine nella forma $\exists \mathbf{w} \chi(\mathbf{x},\mathbf{w})$, dove $\chi(\mathbf{x},\mathbf{w})$ è una congiunzione di atomi e disuguaglianze u!=v. Un'unione di query congiuntive con disuguaglianze è una disgiunzione finita di query congiuntive con disuguaglianze.

Le query congiuntive con disuguaglianze sono più espressive delle query congiuntive, ma a tal maggior espressività corrisponde un prezzo da pagare: in particolare, il teorema precedentemente enunciato non è valido nel caso di congiuntive con disuguaglianze.

- *Esempio (certain answers – query congiuntive con disuguaglianze)*

Sia M lo stesso schema mapping dell'esempio precedente:

$\Sigma st = \{E(x, y) \rightarrow (\exists z)(H(x, z) \wedge H(z, y))\}$
$\Sigma t = \emptyset.$

Sia I l'istanza source che consiste semplicemente nel fatto E(1,1).
Sia q(x) la query congiuntiva $\exists w\ (H(x,w) \wedge w!=x)$. E' facile verificare che certain(q,I) = {}. Prendiamo in considerazione la seguente soluzione universale per I:

$J = \{H(1, u), H(u, 1)\}$

Si può notare che q(J) = {1,u}, quindi eliminando i valori nulli si ottiene $q(J)\downarrow = \{1\}$!= certain(q,I), per cui il teorema non è soddisfatto.

TEOREMA: Si assuma che $M = (\mathbf{S}, \mathbf{T}, \Sigma st \cup \Sigma t)$ sia uno schema mapping tale che Σst sia un insieme di tgds source-to-target e Σt sia un insieme di target egds e un weakly acyclic set di target tgds.

- Sia q un'unione di query congiuntive con al più una disuguaglianza per query congiuntiva. Allora le risposte certe di q sono computabili in tempo polinomiale, usando una procedura chiamata "*chase disgiuntivo*".
- Sia q un'unione di query congiuntive con disuguaglianze. Il problema delle risposte certe per q è un problema in coNP.
- Computare le risposte certe di unioni di query congiuntive con disuguaglianze può essere un problema coNP-completo anche se l'unione consiste di due query congiuntive ognuna della quali abbia al massimo due disuguaglianze e il cui schema mapping non abbia condizioni target.

Alla luce dei problemi computazionali di cui sopra, si può verificare come il core dia la migliore approssimazione alle risposte certe di unioni di query congiuntive.

Si assuma che $M = (\mathbf{S}, \mathbf{T}, \Sigma st \cup \Sigma t)$ sia uno schema mapping tale che Σst sia un insieme di tgds source-to-target e Σt sia un insieme di target egds e target tgds. Si assuma anche che I sia un'istanza source per la quale esistano soluzioni universali. Sia J0 il core delle soluzioni universali per I. Sia q un unione di query congiuntive con disuguaglianze.

- $q(J0) \subseteq q(J)$, *per ogni soluzione universale* J *per I;*
- $q(J0)\downarrow = \cap\{q(J) : J$ *è universale per I*$\} \subseteq certain(q, I)$.

Tutto quanto visto finora prevedeva l'assunzione che le soluzioni di un problema

di Data Exchange corrispondessero al concetto di "possibili database" espresso nelle risposte certe. Tuttavia si è visto come le soluzioni universali siano le soluzioni preferite nel Data Exchange; alla luce di questa considerazione, si può cambiare approccio, e far corrispondere ai "possibili database" l'insieme delle soluzioni universali per una data istanza.

- *Risposte certe:*
  "Possibili mondi" = Soluzioni
- *Risposte certe universali:*
  "Possibili mondi" = Soluzioni universali

Se I è una istanza source, allora le *risposte certe universali* di q su I rispetto a M = (**S**,**T**, Σ) è l'insieme *u-certain*$_M$(*q*, *I*) = ∩ {*q*(*J*) : *J* ∈ è universale per I*}.*

Dalle definizioni segue che *certain*(*q*, *I*) ⊆ *u-certain*(*q*, *I*) . Inoltre, se q è un'unione di query congiuntive e I è un'istanza source per la quale esiste soluzione universale, si ha che *certain*(*q*, *I*) = *u-certain*(*q*, *I*).

Le risposte certe universali possono essere computate in tempo polinomiale se q è un'unione di query congiuntive con disuguaglianze; la semantica delle risposte certe universali è identica a quella delle risposte certe per le unioni di query congiuntive, ma le due semantiche possono divergere nel caso delle unioni di query congiuntive con disuguaglianze.

Schema mapping ***M*** *= (**S**, **T**,* Σ_{st}, Σ_t*) tale che:*

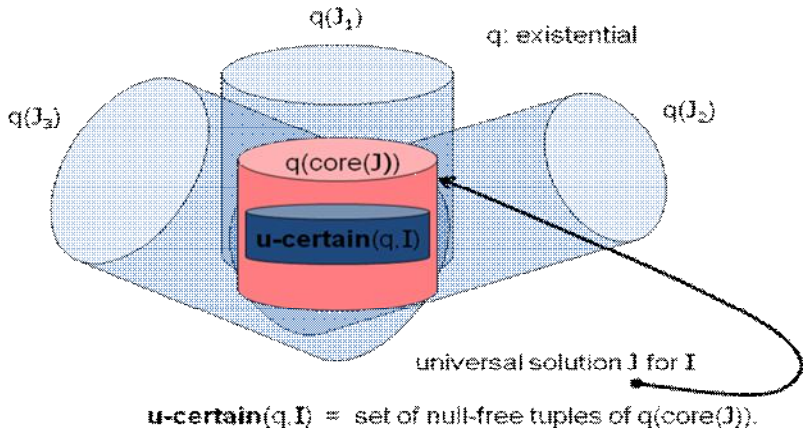
- Σ_{st} *è un insieme di tgds source-to-target*
- Σ_t *è un insieme di target egds e target tgds.*

*Sia q una query esistenziale su **T**.*

- *Se I è un'istanza source e J è una soluzione universale per I, allora*

u- certain*(q,I) = l'insieme di tutte le tuple "senza null" in q(core(J)).*

Si noti che l'unione di query congiuntive con disuaglianze è un caso speciale di query esistenziali.



u-certain(q,I) = set of null-free tuples of q(core(J)).

COMPOSING SCHEMA MAPPINGS

Supponiamo di avere due schema mappings tali che lo schema target del primo è lo schema sorgente del secondo. In questa sezione introduciamo il problema del composition schema mappings dandone una definizione formale ed elencando una serie di risultati ottenuti. Il problema informalmente è quello di, dati due schema mappings, produrne un terzo tra lo schema sorgente del primo e lo schema target del secondo che ha lo stesso effetto dell'applicazione successivi dei due schema mappings. Diamo ora una definizione formale.

DEF. Scriviamo Inst(M) l'insieme di tutte le istanze $\langle I, J \rangle$ di M. Siano $M_{12} = (S_1, S_2, \sum_{12})$ e $M_{23} = (S_2, S_3, \sum_{23})$ due schema mappings tali che S_1, S_2, S_3 non abbiano simboli relazionali in comune. Uno schema mapping $M_{13} = (S_1, S_3, \sum_{13})$ è una composizione di M_{12} e M_{23} se $Inst(M_{13}) = Inst(M_{12}) \circ Inst(M_{23})$, ossia:

$Inst(M_{13}) = \{ \langle I_1, I_3 \rangle \mid \exists I_2$ tale che $\langle I_1, I_2 \rangle \in Inst(M_{12})$ e $\langle I_2, I_3 \rangle \in Inst(M_{23}) \}$. ■

Siano $M = (S_1, S_3, \sum)$ e $M' = (S_1, S_3, \sum')$ composizioni di M_{12} e M_{23} allora gli insiemi $\sum$ e $\sum'$ sono logicamente equivalenti.

$Inst(M_{12})$ e $Inst(M_{23})$ sono chiuse sotto l'isomorfismo, quindi possiamo dire che la loro composizione è anch'essa chiusa sotto l'isomorfismo. Conseguentemente $Inst(M_{12}) \circ Inst(M_{23})$ può essere identificata con la seguente query che chiameremo composition query di M_{12} e M_{23} : date due istanze I_1 e I_3 è $\langle I_1, I_3 \rangle \in$

$Inst(M_{12}) \circ Inst(M_{23})$?

Data una formale definizione del problema osserviamo ora alcuni sviluppi tecnici che ne derivano. Il punto chiave è la chiusura dei linguaggi di schema mapping sotto l'operatore di composizione, quindi dati due schema mapping nei quali le tgds vengono espresse in una logica L, è loro composizione esprimibile nella stessa logica L? Se si può essere efficientemente calcolata? Vediamo ora un po' di risultati.

Composing s-t tgds. Query composition è sempre esprimibile in infinite intersezioni di formule del primo ordine e da insiemi infiniti di formule del primo

ordine. Andiamo quindi ad indagare quando la composizione di due schema mapping in meno espressive ma più trattabili formalismi logici. Andiamo a studiare anche se la composizione di due schema mappings è definibile nello stesso formalismo logico usato per definire i due schema mappings. Cominciamo ad elencare i risultati positivi sul composing s-t tgds per poi passare a risultati tutt'altro che positivi.

Positive result.

Siano S_1, S_2 e S_3 tre schemi senza simboli di relazione in comune e $M_{12} = (S_1 , S_2 , \sum_{12})$ e $M_{23} = (S_2 , S_3 , \sum_{23})$ due schema mappings:

1. Siano $\sum_{12}$ e $\sum_{23}$ due insiemi finiti di full s-t tgds, la composizione $M_{12} \circ M_{23}$ è definibile da un insieme finito di full s-t tgds. Conseguentemente, composition query di M_{12} e M_{23} è una query tempo polinomiale.
2. Se $\sum_{12}$ è un insieme finito di full s-t tgds e $\sum_{23}$ è un insieme finito di s-t tgds, la composizione $M_{12} \circ M_{23}$ è definibile da un insieme di s-t tgds. Conseguentemente, composition query di M_{12} e M_{23} è una query tempo polinomiale.

Negative result.

3. Se $\sum_{12}$ è un insieme finito di s-t tgds e $\sum_{23}$ è un insieme finito di full s-t tgds, la composizione $M_{12} \circ M_{23}$ ha le seguenti proprietà:
   a. $M_{12} \circ M_{23}$ non è definibile da ogni insieme finito di s-t tgds ma è definibile da un insieme infinito di s-t tgds.
   b. $M_{12} \circ M_{23}$ è definibile da una formula del primo ordine, quindi è calcolabile tempo polinomiale.
4. Se $\sum_{12}$ è un insieme di una singola s-t tgds e $\sum_{23}$ è un insieme finito di full s-t tgds, la composizione $M_{12} \circ M_{23}$ non è definibile da un insieme finito o infinito di s-t tgds.
5. Se $\sum_{12}$ è un insieme finito di s-t tgds e $\sum_{23}$ è un insieme di una singola full s-t tgds, allora la composizione $M_{12} \circ M_{23}$ è NP-complete.
6. Se $\sum_{12}$ e $\sum_{23}$ sono un insieme finito di s-t tgds, la composizione $M_{12} \circ M_{23}$ è in NP. Conseguentemente, composition query di M_{12} e M_{23} è definibile da una formula esistenziale del secondo ordine.

Come si è visto la composizione di due schema mappings specificati da s-t tgds può non essere definibile da un insieme di s-t tgds. Dall'ultimo risultato che abbiamo elencato sappiamo però che è sempre definibile da una formula esistenziale del secondo ordine.

Second-Order tgds. In questa sezione vediamo come la composizione di due schema mappings specificati da s-t tgds è sempre esprimibile da una forma ristretta di formule del secondo ordine, le quali chiameremo second-order tuple generating dependencies, in breve (SO tgds). Intuitivamente una SO tgds è una s-t tgd estesa con funzioni ed uguaglianze quantificate esistenzialmente. Una SO tgd gode della proprietà di chiusura sotto la composizione, ossia che la composizione di due schema mapping specificati da SO tgds è uno schema mapping esprimibile in SO tgds. Inoltre gode di altre buone proprietà che vedremo nel seguito.

Diamo la definizione di SO tgd.

DEF. Siano S uno schema sorgente e T uno schema target. Una SO tgd è una formula della forma:

$\exists f_1 \dots \exists f_m((\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)) \wedge \dots \wedge (\forall \mathbf{x}_n(\phi_n \rightarrow \psi_n)))$, dove

- Ogni f_i è un simbolo di funzione,

- ogni ϕ_i è un'unione di atomi da S e uguaglianze di termini,

- ogni ψ_i è un'unione di atomi da T.

Un esempio è la seguente formula:

$\exists f (\forall e(Emp(e) \rightarrow Mgr(e,f(e)) \wedge \forall e(Emp(e) \wedge (e=f(e)) \rightarrow SelfMgr(e)))$. ■

Vediamo un po' di teoremi e risultati importanti:

1. La composizione di SO tgds è definibile in un SO tgds
2. Esiste un algoritmo per comporre SO-tgds.
3. La procedura chase può essere estesa a schema mappings specificati da SO tgds in modo da produrre soluzioni universali in tempo polinomiale.
4. Per gli schema mapping specificati da SO tgds, le certain answers di query congiuntive target sono calcolabili in tempo polinomiale

Computing. Diamo ora un algoritmo di composizione e vediamolo subito applicato ad un esempio per comprenderlo meglio.

Consideriamo tre schema mappings tali che S_1, S_2, S_3 . S_1 consiste di un simbolo di relazione unario Emp di impiegati, S_2 di un simbolo di relazione binario Mgr1 che associa ogni impiegato con il suo manager e S_3 che ha due simboli di relazione, uno binario Mgr ed uno unario SelfMgr. Siano $\sum_{12}$ e $\sum_{23}$:

$\sum_{12} = \exists f (\forall e (Emp(e) \rightarrow Mgr1(e, f(e))))$

$\sum_{23} = \forall e \forall m (Mgr1(e, m) \rightarrow Mgr(e, m)) \wedge \forall e (Mgr1(e, e) \rightarrow SelfMgr(e))$

Algoritmo Compose (M_{12}, M_{23})

Input: due schema mappings $M_{12} = (S_1 , S_2 , \sum_{12})$ e $M_{23} = (S_2 , S_3 , \sum_{23})$ con $\sum_{12}$ e $\sum_{23}$ SO tgds.

Output: una composizione $M_{13} = (S_1 , S_3 , \sum_{13})$ dove $\sum_{13}$ è un insieme di SO tgds.

1. Dividere le SO tgds in $\sum_{12}$ e $\sum_{23}$.

   Es: $C_{12} = Emp(e) \rightarrow (Mgr1(e, f(e))$

   $C_{23} = Mgr1(e,m) \rightarrow Mgr(e,m)$

   $Mgr1(e,e) \rightarrow SelfMgr(e)$

2. Componi C_{12} con C_{23}

   Es: $P_1 : Emp(e_0) \wedge (e=e_0) \wedge (m = f(e_0)) \rightarrow Mgr1(e,m)$

   $P_2 : Emp(e_0) \wedge (e=e_0) \wedge (e = f(e_0)) \rightarrow SelfMgr(e)$

3. Costruisce M_{13}

   Es: $\sum_{13} = \exists f (\exists e_0 \exists e \exists m \ P_1 \wedge \exists e_0 \exists e \ P_2)$

4. Return $M_{13} = (S_1, S_3, \sum_{13})$

CONCLUSIONI

Il problema del Data Exchange è stato di grande attualità per diversi anni ed in diversi periodi della storia dell'informatica, ma forse proprio ai giorni d'oggi ha raggiunto il suo picco d'interesse, dal momento che è alla base di problematiche recenti quali il Data Warehousing, i tasks ETL, l'XML publishing e l'XML storage.

Nel corso della presente relazione si è visto come sia il caso di concentrarsi su una particolare classe di soluzioni al problema, denominata "classe delle soluzioni universali", le quali godono di una serie di proprietà interessanti: innanzitutto non contengono né più né meno di quanto richiesto dalle specifiche del problema, ed inoltre sono particolarmente adatte per il problema del query answering, in quanto le query congiuntive valutate su di esse forniscono l'insieme delle *risposte certe.*

Una problematica di attualità riguarda la necessità di investigare quanto ancora le soluzioni universali possano essere utili nel query answering: in particolare si devono valutare le condizioni per cui una certa query q possa essere ricondotta ad una query del primo ordine q* tale che le risposte certe di q possano essere computate valutando q* su una soluzione universale.

Per un'istanza del source si è comunque visto che possono esistere molte soluzioni universali, per cui ci si pone il problema di capire quale sia la migliore tra di esse. Gli studi attuali portano a considerare il *core* la migliore soluzione universale, nonché la più "piccola" tra di esse, seguendo un approccio *"small is beautiful".* Tuttavia ad oggi non è ancora perfettamente chiaro quanto sia necessario compiere uno sforzo in più per ottenere il core, soprattutto se tale sforzo è messo in relazione ai reali benefici da esso offerti.

I risultati presentati nella relazione riguardano il data Exchange tra schemi relazionali; un possibile sviluppo futuro concerne l'applicabilità di molte delle soluzioni qui descritte nel caso degli *schemi XML*: è interessante in particolar modo sapere quanto la nozione di soluzione universale può essere estesa per coprire gli schemi XML.