Describing Data Sources

Vũ Tuyết Trinh

1

Motivation and Outline

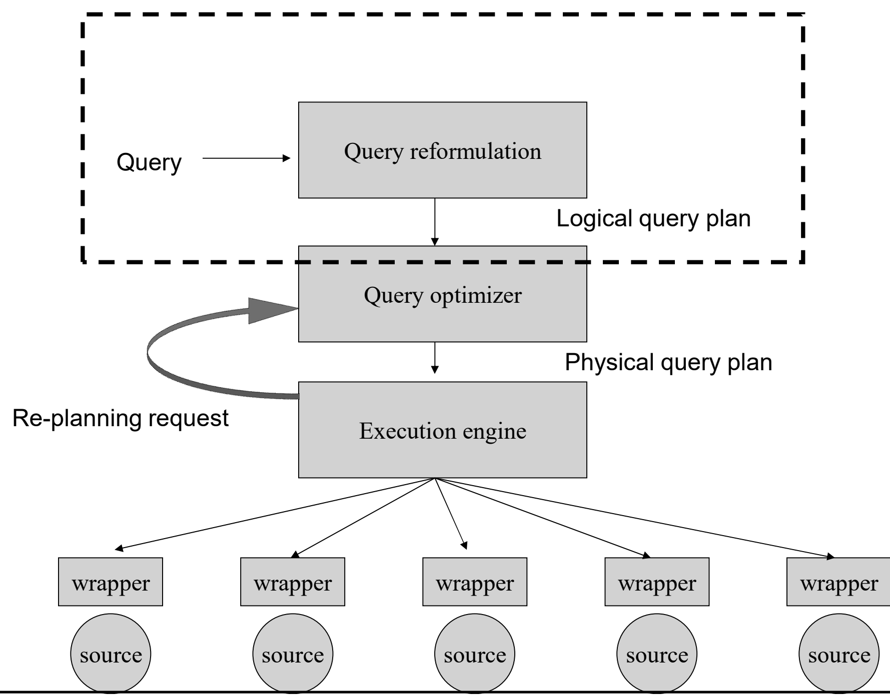
- Descriptions of data sources enable the data integration system to:
  - Determine which sources are relevant to a query
  - Access the sources appropriately
  - Combine data from multiple sources
  - Overcome limitations that specific sources may have
  - Identify opportunities for more efficient query evaluation
- Source descriptions are a formalism for specifying the important aspects of data sources.

2

Outline

- ➢Introduction to semantic heterogeneity
- Schema mapping languages
- Data-level heterogeneity
- Query unfolding

3



4

Schema Heterogeneity

- Schema heterogeneity is a fact of life.
  - Whenever schemas are designed by different people/organizations, they will be different, even if they model the *same* domain!
- The goal of schema mappings is to reconcile schema heterogeneity:
  - Mostly between the mediated schema and the schema of the data sources.

5

Schema Heterogeneity by Example

Mediated Schema

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

logic

Sources

S1
Movie(title, director, year, genre)
Actor(AID, firstName, lastName, nationality, yearofBirth)
ActorPlays(AID, MID)
MovieDetails(MID, director, genre, year)

S2
Cinemas(place, movie, start)

S3
NYCCinemas(name, title, startTime)

S4
Reviews(title, date, grade, review)

S5
MovieGenres(title, genre)

S6
MovieDirectors(title, dir)

S7
MovieYears(title, year)

6

Table and Attribute Naming

Mediated Schema

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

Table and attribute names

Sources

S1
Movie(title, director, year, genre)
Actor(AID, firstName, lastName, nationality, yearofBirth)
ActorPlays(AID, MID)
MovieDetails(MID, director, genre, year)

S2
Cinemas(place, movie, start)

S3
NYCCinemas(name, title, startTime)

S4
Reviews(title, date, grade, review)

S5
MovieGenres(title, genre)

S6
MovieDirectors(title, dir)

S7
MovieYears(title, year)

7

Tabular Organization of Schema

Mediated Schema

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

Different tabular organization

Sources

S1
Movie(title, director, year, genre)
Actor(AID, firstName, lastName, nationality, yearofBirth)
ActorPlays(AID, MID)
MovieDetails(MID, director, genre, year)

S2
Cinemas(place, movie, start)

S3
NYCCinemas(name, title, startTime)

S4
Reviews(title, date, grade, review)

S5
MovieGenres(title, genre)

S6
MovieDirectors(title, dir)

S7
MovieYears(title, year)

8

Schema Coverage

Mediated Schema

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

Sources

Different coverage and detail

S1
Movie(title, director, year, genre)
Actor(AID, firstName, lastName, nationality, yearofBirth)
ActorPlays(AID, MID)
MovieDetails(MID, director, genre, year)

S2
Cinemas(place, movie, start)

S3
NYCCinemas(name, title, startTime)

S4
Reviews(title, date, grade, review)

S5
MovieGenres(title, genre)

S6
MovieDirectors(title, dir)

S7
MovieYears(title, year)

9

Semantic Heterogeneity Summary

- Differences in:
  - Naming of schema elements
  - Organization of tables
  - Coverage and detail of schema
  - Data-level representation (IBM vs. International Business Machines)
- Reason:
  - Schemas probably designed for different applications/contexts.

10

More in Source Descriptions

- Possible access patterns to the data source:
  - Are there required inputs? (e.g., web forms, web services)
  - Can the source process complex database queries?
- Source completeness:
  - Is the source complete, or partially complete?
- Reliability, load restrictions, mirror sites, …

11

Outline

✓Introduction to semantic heterogeneity

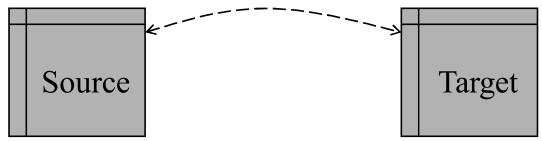
➢Schema mapping languages

- Data-level heterogeneity
- Query unfolding

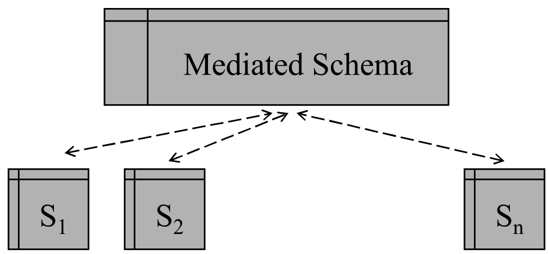
12

Principles of Schema Mappings

Schema mappings describe the relationship between:



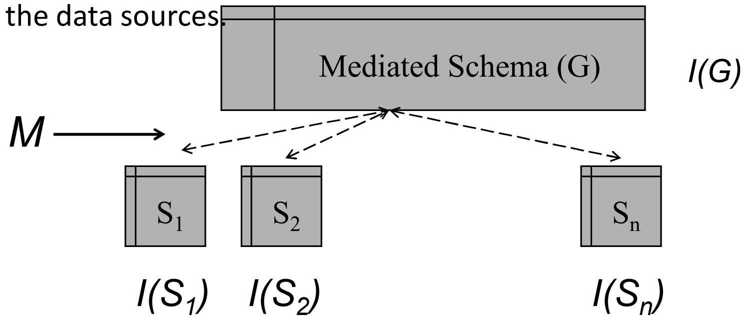
Or between:



13

Semantics of Schema Mappings

Formally, schema mappings describe a relation: which instances of the mediated schema are consistent with the current instances of the data sources.



$I(G)$ $(I(S_i))$: the set of possible instances of the schema G (S_i).

$$M_R \subseteq I(G) \times I(S_1) \times \ldots I(S_n)$$

14

Relations, explained

- A relation is a subset of the Cartesian product of its columns' domains:

1	1
2	4
3	9
4	16
5	25

The table on the left is a subset of the Cartesian product of

Int x Int.

The table describes the Squared relation.

Simialry, M_r specifies the possible instances of the mediated schema, given instances of the sources.

15

Possible Instances of Mediated Schema: Simple Example

- Source 1: (Director, Title, Year) with tuples
  - {(Allen, Manhattan, 1979),
  - (Coppola, GodFather, 1972)}
- Mediated schema: (Title, Year)
  - Simple projection of Source 1
  - Only one possible instance:
  - {(Manhattan, 1979), (GodFather, 1972)}

16

Possible Instances of Mediated Schema: Second Example

- Source 1: (Title, Year) with tuples
  - {(Manhattan, 1979), (GodFather, 1972)}
- Mediated schema: (Director, Title, Year)
  - Possible instance 1: {(Allen, Manhattan, 1979), (Coppola, GodFather, 1972)}
  - Another possible instance 2: {(Halevy, Manhattan, 1979), (Stonebraker, GodFather, 1972)}.
- This matters when we answer queries:
  - See next slide.

17

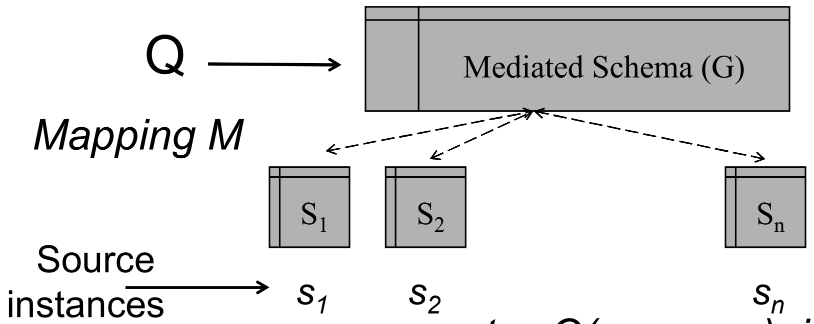
Answering Queries over Possible Instances of Mediated Schema

- Mediated schema: (Director, Title, Year)
  - Possible instance 1: {(Allen, Manhattan, 1979), (Coppola, GodFather, 1972)}
  - Another possible instance 2: {(Halevy, Manhattan, 1979), (Stonebraker, GodFather, 1972)}.
- Query Q1: return all years of movies
  - Answer: (1979, 1972) are certain answers.
- Query Q2: return all directors
  - No certain answers because no directors appear in all possible instances of the mediated schema.

18

Certain Answers Makes this Formal

An answer is certain if it is true in every instance of the mediated schema that is consistent with: (1) the instances of the sources and (2) the mapping *M*.



$$t \in Q(s_1, \ldots, s_n) \text{ iff}$$
$$t \in Q(g) \text{ for } \forall g, \text{ s.t. } (g, s_1, \ldots s_n) \in M_R$$

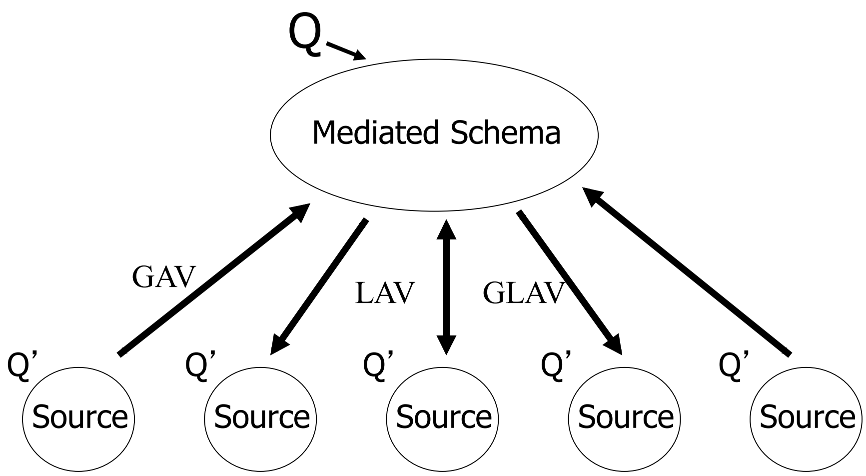
19

Desiderata from Source Description Languages

- Flexibility:
  - Should be able to express relationships between real schemata
- Efficient reformulation:
  - Computational complexity of reformulation and finding answers
- Easy update:
  - Should be easy to add and delete sources

20

Languages for Schema Mapping



21

Global-as-View (GAV)

- Mediated schema defined as a set of views over the data sources

Movie: title, director, year, genre

S1
Movie(MID,title)
Actor(AID, firstName, lastName, nationality, yearofBirth)
ActorPlays(AID, MID)
MovieDetails(MID, director, genre, year)

$$
\begin{aligned}
&Movie(title, director, year, genre) \supseteq \\
&\quad S1.Movie(MID, title), \\
&\quad S1.MovieDetail(MID, director, genre, year)
\end{aligned}
$$

22

GAV: Formal Definition

A set of expressions of the form:

$G_i(\overline{X}) \supseteq Q(\overline{S})$ or $G_i(\overline{X}) = Q(\overline{S})$

open-world assumption

closed-world assumption

- G_i: relation in mediated schema
- $Q(\overline{S})$: query over source relations

23

GAV Example

Movie: title, director, year, genre

$$Movie(title, director, year, genre) \supseteq$$
$$S1.Movie(MID, title), S1.MovieDetail(MID, director, genre, year)$$

$$Movie(title, director, year, genre) \supseteq$$
$$S5.MovieGenres(title, genre),$$
$$S6.MovieDirectors(title, director),$$
$$S7.MovieYears(title, year)$$

24

GAV Example (cont.)

Plays: movie, location, startTime

$$Plays(movie, location, startTime) \supseteq$$
$$S2.Cinemas(location, movie, startTime)$$

$$Plays(movie, location, startTime) \supseteq$$
$$S3.NYCCinemas(location, movie, startTime)$$

S2
Cinemas(place, movie, start)

S3
NYCCinemas(name, title, startTime)

25

Reformulation in GAV

- Given a query Q on the mediated schema:
  - Return the best query possible on the data sources.

26

Reformulation in GAV = Query/View Unfolding

Q(title, location, startTime) :-
Movie(title, director, year, "comedy"),
Plays(title, location, st), st ≥ 8pm

$Movie(title, director, year, genre) \supseteq$

$\boxed{S1.Movie(MID, title), S1.MovieDetail(MID, director, genre, year)}$

$Plays(movie, location, startTime) \supseteq$

$\boxed{S2.Cinemas(location, movie, startTime)}$

27

First Reformulation

Q(title, location, startTime) :-
Movie(title, director, year, "comedy"),
Plays(title, location, st), st ≥ 8pm

↓

Q'(title, location, startTime) :-
S1.Movie(MID, title),
S1.MovieDetail(MID, director, "comedy", year)
S2.Cinemas(location,title, st), st ≥ 8pm

28

Another Reformulation

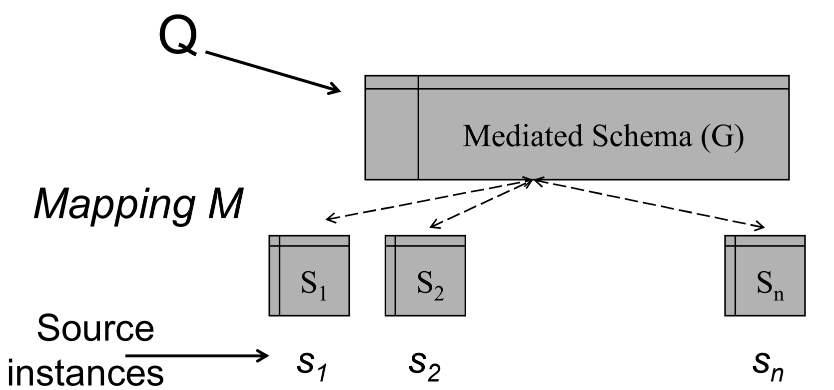
Q(title, location, startTime) :-
Movie(title, director, year, "comedy"),
Plays(title, location, st), st ≥ 8pm

↓

Q'(title, location, startTime) :-
S1.Movie(MID, title),
S1.MovieDetail(MID, director, "comedy", year)
S3.NYCCinemas(location,title, st), st ≥ 8pm

29

Certain Answers (recall definition)



$$t \in Q(s_1, \ldots, s_n) \text{ iff}$$
$$t \in Q(g) \text{ for } \forall g, \text{ s.t. } (g, s_1, \ldots s_n) \in M_R$$

30

Semantics of GAV

(g, $s_1, \ldots, s_n$) $\in M_R$ if:

$$\begin{array}{ccc} G & S_1, \ldots, S_n \\ \downarrow & \downarrow \quad\quad \downarrow \\ g & s_1, \ldots, s_n \end{array}$$

$$G_i(\overline{X}) \supseteq Q(\overline{S}) \quad \Rightarrow$$

The extension of G_i in *g* is a super-set of evaluating Q_i on the sources.

$$G_i(\overline{X}) = Q(\overline{S}) \quad \Rightarrow$$

The extension of G_i in *g* is equal to evaluating Q_i on the sources.

31

Tricky Example for GAV

S8: stores pairs of (actor, director)

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

$$Actors(NULL, actor) \supseteq S8(actor, director)$$

$$Movie(NULL, director, NULL, NULL) \supseteq S8(actor, director)$$

32

Tricky Example for GAV

$Actors(NULL, actor) \supseteq S8(actor, director)$

$Movie(NULL, director, NULL, NULL) \supseteq S8(actor, director)$

Given the S8 tuples:
({Keaton, Allen}, {Pacino, Coppola})

We'd get tuples for the mediated schema:

Actors: ({NULL, Keaton}, {NULL, Pacino})
Movie: ({NULL, Allen, NULL, NULL},
{NULL, Coppola, NULL, NULL})

33

Tricky Example (2)

Actors: ({NULL, Keaton}, {NULL, Pacino})
Movie: ({NULL, Allen, NULL, NULL},
{NULL, Coppola, NULL, NULL})

Can't answer the query:
Q(actor, director) :-
Actors(title, actor),
Movie(title, director, genre, year)

LAV (Local as View) will solve this problem

34

GAV Summary

- Mediated schema is defined as views over the sources.
- Reformulation is conceptually easy
  - Polynomial-time reformulation and query answering.
- GAV forces everything into the mediated schema's perspective:
  - Cannot capture a variety of tabular organizations.

35

Local-as-View (LAV)

- Data sources defined as views over mediated schema!

S5
MovieGenres(title, genre)

S6
MovieDirectors(title, dir)

S7
MovieYears(title, year)

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

$$S5.MovieGenres(title, genre) \subseteq Movie(title, dir, year, genre)$$

$$S6.MovieDirectors(title, dir) \subseteq Movie(title, dir, year, genre)$$

36

Local-as-View (LAV)

- Data sources defined as views over mediated schema!

S8
ActorDirectors(actor,dir)

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

$$S8.ActorDirectors(actor, dir) \subseteq$$
$$Movie(title, dir, year, genre), Actor(title, actor)$$
$$year \geq 1980$$

37

LAV: Formal Definition

A set of expressions of the form:

$S_i(\overline{X}) \subseteq Q_i(G)$ or $S_i(\overline{X}) = Q_i(G)$

open-world assumption

closed-world assumption

- $S_i(\overline{X})$: source relation
- $Q_i(G)$: query over mediated schema

38

Semantics of LAV

$(g, s_1, \ldots, s_n) \in M_R$ if:

$$\begin{array}{ccc} G & S_1, \ldots, S_n \\ \downarrow & \downarrow \quad\quad \downarrow \\ g & s_1, \ldots, s_n \end{array}$$

$S_i(\overline{X}) \subseteq Q_i(G) \quad \Rightarrow$

The result of Q_i over *g* is a superset of s_i.

$S_i(\overline{X}) = Q_i(G) \quad \Rightarrow$

The result of Q_i over *g* equals s_i.

Possible Databases

Unlike GAV, LAV definitions imply a *set* of possible databases for the mediated schema.

$S8.ActorDirectors(actor, dir) \subseteq$

$Movie(title, dir, year, genre), Actor(title, actor)$

$S8 : \{(Keaton, Allen)\}$

Two possible databases for the mediate schema are:

Movie: {("manhattan", allen, 1979, comedy)}
Actor:{("manhattan", keaton)}

Movie: {("foobar", allen, 1979, comedy)}
Actor:{("foobar", keaton)}

Possible Databases

Since the source may be incomplete, other tuples may be in the instance of the mediated schema:

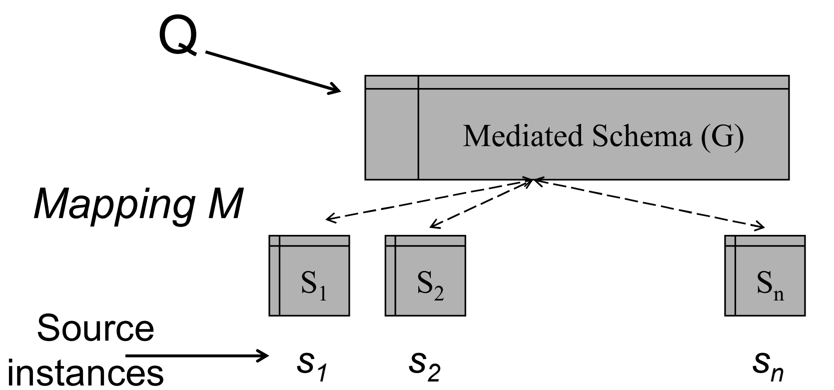
$$S8.ActorDirectors(actor, dir) \subseteq$$
$$Movie(title, dir, year, genre), Actor(title, actor)$$

$$S8 : \{(Keaton, Allen)\}$$

Movie: {(manhattan, allen, 1981, comedy),
(leatherheads,clooney,2008, comedy)}
Actor:{(manhattan, keaton),
(the godfather, keaton)}

41

Certain Answers: by now you know this slide by heart



$$t \in Q(s_1, \ldots, s_n) \text{ iff}$$
$$t \in Q(g) \text{ for } \forall g, \text{ s.t. } (g, s_1, \ldots s_n) \in M_R$$

42

Certain Answers Example 1

$S8.ActorDirectors(actor, dir) \subseteq$
$Movie(title, dir, year, genre), Actor(title, actor)$
$year \geq 1980$

$$S8 : \{(Keaton, Allen)\}$$

$Q(actor, dir) :-$
$Movie(title, dir, year, genre), Actor(title, actor)$

Only one certain answer: *(Keaton, Allen)*

43

Certain Answers Example 2

$V_8(dir) = DirectorActor(ID, dir, actor)$ {Allen}
$V_9(actor) = DirectorActor(ID, dir, actor)$ {Keaton}

$Q(dir, actor) :- Director(ID, dir), Actor(ID, actor)$

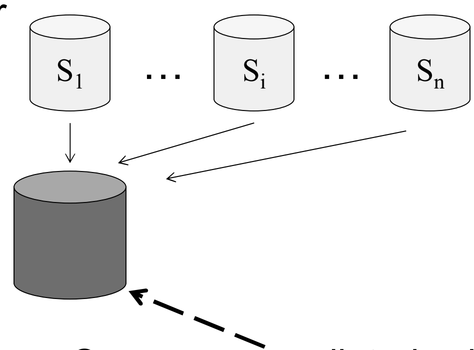
Under closed-world assumption:
single DB possible $\Rightarrow$ (Allen,Keaton)

Under open-world assumption:
no certain answers.

44

Reformulation in LAV

We're given tuples for sources (expressed as views)



Mediated schema (but no tuples)

Query over mediated schema

This is exactly the problem of:…
Answering queries using views!

45

Local-as-View Summary

- Reformulation = answering queries using views
- Algorithms work well in practice:
  - Reformulation is not the bottleneck
- Under some conditions, guaranteed to find all certain answers
  - In practice, they typically do.
- LAV expresses incomplete information
  - GAV does not. Only a single instance of the mediated schema is consistent with sources.

46

LAV Limitation

Movie: title, director, year, genre

S1
Movie(MID,title)
Actor(AID, firstName, lastName,nationality,yearofBirth)
ActorPlays(AID, MID)
MovieDetails(MID, director, genre, year)

If a key is internal to a data source, LAV cannot use it.

47

GLAV – a better solution ?

A set of expressions of the form:

$Q^S(\overline{X}) \subseteq Q^G(\overline{X})$ or $Q^S(\overline{X}) = Q^G(\overline{X})$

- Q^G*:* query over mediated schema
  - Q^S: query over data sources

48

GLAV Example

$$S1.Movie(MID, title), S1.MovieDetail(MID, dir, genre, year)$$
$$\subseteq$$
$$Movie(title, dir, "comedy", year), year \geq 1970$$

49

Reformulation in GLAV

$$\boxed{Q^S(\overline{X}) \subseteq Q^G(\overline{X})}$$

- Given a query Q
- Find a rewriting Q' using the views

$$Q_1^G, \ldots, Q_n^G$$

- Create Q'' by replacing: $Q_i^G \rightarrow Q_i^S$
- Unfold $Q_1^S, \ldots, Q_n^S$

50

An Alternative Notation for GLAV: Tuple Generating Dependencies

Tuple generating dependencies (Chapter 2.1.2) can be used to specify GLAV expressions.

The TGD

$$(\forall \overline{X}) s_1(\overline{X}_1), \ldots, s_m(\overline{X}_m) \rightarrow (\exists \overline{Y})\, t_1(\overline{Y}_1), \ldots, t_k(\overline{Y}_k)$$

is equivalent to the GLAV expression: $Q^S(\overline{X}) \subseteq Q^G(\overline{X})$

where:

$$Q^S(\overline{X}) :- s_1(\overline{X}_1), \ldots, s_m(\overline{X}_m)$$
$$Q^G(\overline{Y}) :- t_1(\overline{Y}_1), \ldots, t_k(\overline{Y}_k)$$

51

GLAV -> TGD Example

$$S1.Movie(MID, title), S1.MovieDetail(MID, dir, genre, year)$$
$$\subseteq$$
$$Movie(title, dir, "comedy", year), year \geq 1970$$

➔

$$S1.Movie(MID, title) \wedge S1.MovieDetail(MID, dir, genre, year)$$
$$\rightarrow Movie(title, dir, "comedy", year) \wedge year \geq 1970$$

Reformulation with TGD descriptions can be done relatively directly with the Inverse Rules Algorithm

52

Outline

✓Introduction to semantic heterogeneity

✓Schema mapping languages

➢Data-level heterogeneity

- Query unfolding

53

Data-Level Heterogeneity

- Huge problem in practice:
  - Data coming from different sources rarely joins perfectly.
- Differences of scale:
  - Celsius vs. Fahrenheit
  - Numeric vs. letter grades
  - First Name, Last Name vs. FullName
  - Prices with taxes or without
  - …

54

Mappings with Transformations

$$S(city, temp - 32 * 5/9, month) \subseteq$$
$$Weather(city, temp, humidity, month)$$

$$CDStore(cd, price) \subseteq CD\mathrm{Pr}ices(cd, state, price * (1 + rate)),$$
$$LocalTaxes(state, rate)$$

55

Reference Reconciliation

- Multiple ways to refer to the same real-world entity:
  - John Smith vs. J. R. Smith
  - IBM vs. International Business Machines
  - Alon Halevy vs. Alon Levy
  - South Korea vs. Republic of Korea
- Create concordance tables:
  - Pairs of corresponding values
  - How? See the next chapters!

56

Outline

✓Introduction to semantic heterogeneity
✓Schema mapping languages
✓Data-level heterogeneity
➢ Query unfolding

57

Query Unfolding

- Query composition is an important mechanism for writing complex queries.
  - Build query from views in a bottom up fashion.
- Query unfolding "unwinds" query composition.
- Important for:
  - Comparing between queries expressed with views
  - Query optimization (to examine all possible join orders)
  - Unfolding may even discover that the composition of two satisfiable queries is unsatisfiable! (exercise: find such an example).

58

Query Unfolding Example

$$Q_1(X,Y) :- Flight(X,Z), Hub(Z), Flight(Z,Y)$$
$$Q_2(X,Y) :- Hub(Z), Flight(Z,X), Flight(X,Y)$$
$$Q_3(X,Z) :- Q_1(X,Y), Q_2(Y,Z)$$

The unfolding of Q_3 is:

$$Q'_3(X,Z) :- Flight(X,U), Hub(U), Flight(U,Y),$$
$$Hub(W), Flight(W,Y), Flight(X,Z)$$

59

Query Unfolding Algorithm

- Find a subgoal $p(X_1, \ldots, X_n)$ such that p is defined by a rule r.
- Unify $p(X_1, \ldots, X_n)$ with the head of r.
- Replace $p(X_1, \ldots, X_n)$ with the result of applying the unifier to the subgoals of r (use fresh variables for the existential variables of r).
- Iterate until no unifications can be found.
- If p is defined by a union of $r_1, \ldots, r_n$, create n rules, for each of the r's.

60

Query Unfolding Summary

- Unfolding does not necessarily create a more efficient query!
  - Just lets the optimizer explore more evaluation strategies.
  - Unfolding is the opposite of rewriting queries using views (see later).
- The size of the resulting query can grow exponentially (exercise: show how).

61

Summary

- Source descriptions include:
  - Schema mappings
  - Completeness, access-pattern limitations
  - Data transformations
  - Additional query-processing capabilities
- Schema mapping languages
  - GAV: reformulation by unfolding
  - LAV/GLAV: reformulation by answering queries using views
- Binding patterns and integrity constraints can lead to tricky cases:
  - Recursive rewritings can often address these.

62

References

- Ravi Krishnamurthy, Witold Litwin, William Kent: Language Features for Interoperability of Databases with Schematic Discrepancies. SIGMOD 1991, 40-49.
- Laks V. S. Lakshmanan, Fereidoon Sadri, Iyer N. Subramanian: SchemaSQL: An extension to SQL for multidatabase interoperability. ACM TODS, 26(4), December 2001.
- Jeffrey D. Ullman: Information Integration Using Logical Views. ICDT 1997: 19-40.
- M. Lenzerini: Data Integration; A Theoretical Perspective, Proc. ACM Symposium on Principles of Database Systems, 2002.
- Alon Y. Halevy: Theory of Answering Queries Using Views, SIGMOD Record 29(4), December 2000.
- Alon Y. Levy: Answering Queries Using Views: A Survey,VLDB Journal, 2001.
- R. J. Miller, L. M. Haas, M. A. Hernandez. Schema Mapping as Query Discovery. VLDB'00.

63