

## Data Integration

Craig Knoblock  
University of Southern California

These slides are based in part on slides from José Luis Ambite and Rao Kambhampati, which are in turn based in part on slides from Alon Halevy.

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings

## Outline

- Database Theory Background
  - **Datalog**
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings

## Datalog

- Datalog Program = set of datalog rules
- Datalog rule = conjunctive query

```
big-LA-buyers(Buyer,Seller, Price) :-  
  person(Buyer, "Los Angeles"),  
  purchase(Buyer, Seller, Product, Price),  
  Price > 10000.
```

head  
body

## Datalog

- Datalog Program = set of datalog rules
- Datalog rule = conjunctive query

```
big-LA-buyers(Buyer,Seller, Price) :-  
  person(Buyer, "Los Angeles"),  
  purchase(Buyer, Seller, Product, Price),  
  Price > 10000.
```

Datalog

Buyer, Seller, Price  
[  $\exists$  Product [ person(Buyer, "Los Angeles") ^  
 purchase(Buyer, Seller, Product, Price) ^  
 Price > 10000 ] ]  
→ big-LA-buyers(Buyer,Seller, Price)

antecedent  
First-Order Logic  
consequent

## Conjunctive Queries and Views

```
CREATE VIEW Big-LA-buyers AS  
SELECT buyer, seller, price  
FROM Person, Purchase  
WHERE Person.city = "Los Angeles" AND  
Person.buyer = Purchase.buyer AND  
Purchase.price > 10000
```

```
big-LA-buyers(Buyer,Seller, Price) :-  
  person(Buyer, "Los Angeles"),  
  purchase(Buyer, Seller, Product, Price),  
  Price > 10000.
```

Datalog rule ~ view definition  
Rule body ~ select-from-where construct of SQL

## Recursion in Datalog

```

path(X, Y) :- arc(X, Y)
path(X, Y) :- path(X, Z), path(Z, Y).

```

Craig Knoblock      University of Southern California      7

## Recursion in Datalog

```

path(X, Y) :- arc(X, Y)
path(X, Y) :- path(X, Z), path(Z, Y).

```

**Semantics:** evaluate the rules bottom-up until a fixpoint:

Craig Knoblock      University of Southern California      8

## Recursion in Datalog

```

path(X, Y) :- arc(X, Y)
path(X, Y) :- path(X, Z), path(Z, Y).

```

**Semantics:** evaluate the rules bottom-up until a fixpoint:

Iteration #0: arc: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
path: {}

Craig Knoblock      University of Southern California      9

## Recursion in Datalog

```

path(X, Y) :- arc(X, Y)
path(X, Y) :- path(X, Z), path(Z, Y).

```

**Semantics:** evaluate the rules bottom-up until a fixpoint:

Iteration #0: arc: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
path: {}  
Iteration #1: path: {(a,b), (a,c), (b,d), (c,d), (d,e)}

Craig Knoblock      University of Southern California      10

## Recursion in Datalog

```

path(X, Y) :- arc(X, Y)
path(X, Y) :- path(X, Z), path(Z, Y).

```

**Semantics:** evaluate the rules bottom-up until a fixpoint:

Iteration #0: arc: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
path: {}  
Iteration #1: path: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
Iteration #2: path gets the new tuples: (a,d), (b,e), (c,e)

Craig Knoblock      University of Southern California      11

## Recursion in Datalog

```

path(X, Y) :- arc(X, Y)
path(X, Y) :- path(X, Z), path(Z, Y).

```

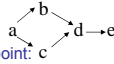
**Semantics:** evaluate the rules bottom-up until a fixpoint:

Iteration #0: arc: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
path: {}  
Iteration #1: path: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
Iteration #2: path gets the new tuples: (a,d), (b,e), (c,e)  
Iteration #3: path gets the new tuple: (a,e)

Craig Knoblock      University of Southern California      12

## Recursion in Datalog

$\text{path}(X, Y) \text{ :- arc}(X, Y)$   
 $\text{path}(X, Y) \text{ :- path}(X, Z), \text{path}(Z, Y).$



**Semantics:** evaluate the rules bottom-up until a fixpoint:  
 Iteration #0: arc: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
 path: {}  
 Iteration #1: path: {(a,b), (a,c), (b,d), (c,d), (d,e)}  
 Iteration #2: path gets the new tuples: (a,d), (b,e), (c,e)  
 Iteration #3: path gets the new tuple: (a,e)  
 Iteration #4: Nothing changes => stop.

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings
  - Local Completeness Information

## Query Containment

- **Query Containment:**  $q' \subseteq q$ 
  - $\forall D q'(D) \subseteq q(D)$
  - $q' \models q$
- **Query Equivalence:**  $q' = q \leftrightarrow q' \subseteq q \wedge q \subseteq q'$
- **Complexity of Query Containment**
  - Conjunctive Queries (CQ), Union of CQs: NP-complete
  - CQ with comparisons ( $=, <, \neq$ ):  $\Pi_p^2$ -complete
  - FOL, recursive queries: Undecidable

## Query Containment for Conjunctive Queries and Datalog

### Method of Canonical Databases

1. Create a canonical database  $D$  that is the "frozen" body of  $q_1$
2. Compute  $q_2(D)$
3. If  $q_2(D)$  contains the "frozen" head of  $q_1$ , then  $q_1 \subseteq q_2$ , otherwise not.

## Query Containment Example

$q_1$  is the CQ:  $\text{path}(X, Y) \text{ :- arc}(X, Z) \ \& \ \text{arc}(Z, W) \ \& \ \text{arc}(W, Y)$

$q_2$  is the value of path in the following recursive Datalog program:

$\text{path}(X, Y) \text{ :- arc}(X, Y)$   
 $\text{path}(X, Y) \text{ :- path}(X, Z) \ \& \ \text{path}(Z, Y)$

Intuitively,  $q_1 =$  paths of length 3;  $q_2 =$  paths of length 1 or more,  $q_1 \subseteq q_2$

1. Freeze  $q_1$ , say with 0, 1, 2, 3 as constants for X, Z, W, Y, respectively.  
 $D = \{\text{arc}(0, 1), \text{arc}(1, 2), \text{arc}(2, 3)\}$   
 Frozen head of  $q_1$  is  $\text{path}(0, 3)$ .
2. Compute  $q_2(D)$      $\text{Ext}(\text{path}) = \{(0,1), (1,2), (2,3), (0,2), (1,3), (0,3)\}$
3. Since frozen head of  $q_1$ ,  $\text{path}(0, 3)$ , is in  $q_2(D)$  then  $q_1 \subseteq q_2$

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- **Dimensions of Data Integration**
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings

## Principal Dimensions of Data Integration

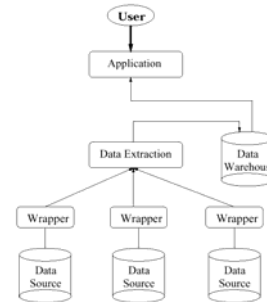
- Virtual vs. materialized architecture
- Access: query only or query & update?
- Mediated schema and query reformulation
  - Content Descriptions
    - Global-as-view
    - Local-as-view
  - Language for descriptions and queries: conjunctive queries (CQs), union of CQs, Datalog (recursion), first-order logic ( $\wedge, \vee, \neg$ ), description logics, ...
- Types of Sources
  - Structured (DB's) vs. semi-structured (Web)
  - Source capabilities: positive and negative

Craig Knoblock

University of Southern California

19

## Materialized Architecture: Data Warehouse

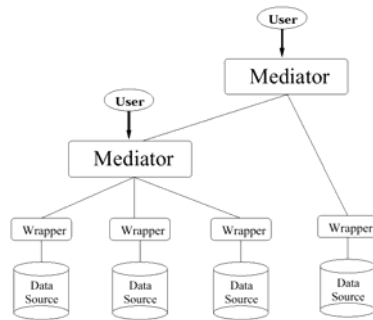


Craig Knoblock

University of Southern California

20

## Virtual Architecture: Mediator

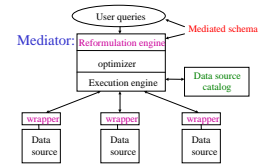


Craig Knoblock

21

## Virtual Integration Architecture

- Leave the data in the sources
- When a query comes in:
  - Determine the relevant sources to the query
  - Break down the query into sub-queries for the sources
  - Get the answers from the sources, and combine them appropriately
- Data is fresh. Approach scalable
- Issues:
  - Relating Sources & Mediator
  - Reformulating the query
  - Efficient planning & execution



Garlic [IBM], Hermes[UMD];Tsimmis, InfoMaster[Stanford]; DISCO[INRIA]; Information Manifold [AT&T]; SIMS/Ariadne[USC];Emerac/Havasu[ASU]

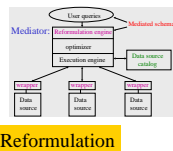
Craig Knoblock

University of Southern California

22

## Desiderata for Relating Source-Mediator Schemas

- **Expressive power:** distinguish between sources with closely related data. Hence, be able to prune access to irrelevant sources.
- **Easy addition:** make it easy to add new data sources.
- **Reformulation:** be able to reformulate a user query into a query on the sources efficiently and effectively.
- **Nonlossy:** be able to handle all queries that can be answered by directly accessing the sources



### Reformulation

- **Given:**
  - A query Q posed over the mediated schema
  - Descriptions of the data sources
- **Find:**
  - A query Q' over the data source relations, such that:
    - Q' provides only correct answers to Q, and
    - Q' provides all possible answers to Q given the sources.

Craig Knoblock

University of Southern California

23

## Source Descriptions

Elements of source descriptions:

- Contents: source contains movies, directors, cast.
- Constraints: only movies produced after 1965.
- Completeness: contains all American movies.
- Capabilities:
  - Negative: source requires movie title or director as input
  - Positive: source can perform selections, joins, ...

Craig Knoblock

University of Southern California

24

## Approaches to Specification of Source Descriptions

- **Global-as-View (GAV):**  
Mediator relation defined as a view over source relations  
Ex: TSIMMIS (Stanford), HERMES (Maryland)
- **Local-as-View (LAV):**  
Source relation defined as view over mediator relations  
Ex: Information Manifold (AT&T), Tukwila(UW), InfoMaster (Stanford), Ariadne (USC)

View ~ named query ~ logical formula

## Query Reformulation

Problem: rewrite the user query expressed in the mediated schema into a query expressed in the source schemas

Given a query  $Q$  in terms of the mediated-schema relations, and descriptions of the information sources,

Find a query  $Q'$  that uses only the source relations, such that

- $Q' \models Q$  (i.e., answers are correct; i.e.,  $Q' \subseteq Q$ ) and
- $Q'$  provides all possible answers to  $Q$  given the sources

## Answering queries using views

Given query  $q$  and view definitions  $V=\{V1...Vn\}$

- $q'$  is an *Equivalent Rewriting* of  $q$  using  $V$  if:
  - $q'$  refers only to views in  $V$ , and
  - $q' = q$
- $q'$  is a *Maximally-Contained Rewriting* of  $q$  using  $V$  if:
  - $q'$  refers only to views in  $V$ , and
  - $q' \subseteq q$ , and
  - there is no rewriting  $q_1$ , such that  $q' \subseteq q_1 \subseteq q$  and  $q_1 \neq q$

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings

## Global-as-View (GAV)

Each mediator relation is defined as a view over source relations.

MovieActor(title,actor)  $\leftarrow$

DB1(id,title,actor,year)

MovieActor(title,actor)  $\leftarrow$

DB2(title,director,actor,year)

MovieReview(title, review)  $\leftarrow$

DB1(id,title,actor,year)  $\wedge$  DB3(id,review)

## Query Reformulation in GAV

Query reformulation = rule unfolding+simplification

Query: Find reviews for 'DeNiro' movies

$q(\text{title,review}) :- \text{MovieActor}(\text{title},\text{'DeNiro'}),$   
 $\text{MovieReview}(\text{title,review})$

1.  $q'(\text{title,review}) :- \text{DB1}(\text{id,title},\text{'DeNiro'},\text{year}),$  Redundant  
 $\text{DB1}(\text{id,title,actor,year}), \text{DB3}(\text{id,review})$

2.  $q'(\text{title,review}) :-$  Redundant wrt 1  
 $\text{DB2}(\text{title,director},\text{'DeNiro'},\text{year}),$   
 $\text{DB1}(\text{id,title,actor, year}), \text{DB3}(\text{id,review})$

## Local-as-View (LAV)

- Each source relation is defined as a view over mediator relations

$$V1(\text{title, year, director}) \stackrel{\subseteq}{=} \text{Movie}(\text{title, year, director, genre}) \wedge \text{American}(\text{director}) \wedge \text{year} \geq 1960 \wedge \text{genre} = \text{'Comedy'}$$

$$V2(\text{title, review}) \stackrel{\subseteq}{=} \text{Movie}(\text{title, year, director, genre}) \wedge \text{year} \geq 1990 \wedge \text{MovieReview}(\text{title, review})$$

Craig Knoblock University of Southern California 31

## Query Reformulation in LAV

Query: *Reviews for comedies produced after 1950*  
 $q(\text{title, review}) :- \text{Movie}(\text{title, year, director, 'Comedy'}), \text{year} \geq 1950, \text{MovieReview}(\text{title, review})$

Reformulated query:  
 $q'(\text{title, review}) :- V1(\text{title, year, director}), V2(\text{title, review})$   $q' \subseteq q$

$V1(\text{title, year, director}) \rightarrow \text{Movie}(\text{title, year, director, genre}) \wedge \text{American}(\text{director}) \wedge \text{year} \geq 1960 \wedge \text{genre} = \text{'Comedy'}$   
 $V2(\text{title, review}) \rightarrow \text{Movie}(\text{title, year, director, genre}) \wedge \text{year} \geq 1990 \wedge \text{MovieReview}(\text{title, review})$

Craig Knoblock University of Southern California 32

## GAV vs. LAV

<ul style="list-style-type: none"> <li>Not modular           <ul style="list-style-type: none"> <li>Addition of new sources changes the mediated schema</li> </ul> </li> <li>Can be awkward to write mediated schema without loss of information</li> <li>Query reformulation easy           <ul style="list-style-type: none"> <li>reduces to view unfolding (polynomial)</li> <li>Can build hierarchies of mediated schemas</li> </ul> </li> <li>Best when           <ul style="list-style-type: none"> <li>Few, stable, data sources</li> <li>well-known to the mediator (e.g. corporate integration)               <ul style="list-style-type: none"> <li>Garlic, TSIMMIS, HERMES</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Modular--adding new sources is easy</li> <li>Very flexible--power of the entire query language available to describe sources</li> <li>Reformulation is hard           <ul style="list-style-type: none"> <li>Involves answering queries only using views (can be intractable)</li> </ul> </li> <li>Best when           <ul style="list-style-type: none"> <li>Many, relatively unknown data sources</li> <li>possibility of addition/deletion of sources               <ul style="list-style-type: none"> <li>Information Manifold, InfoMaster, Emerac</li> </ul> </li> </ul> </li> </ul>
--	--

Craig Knoblock University of Southern California 33

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings

Craig Knoblock University of Southern California 34

## Query Reformulation in LAV

### The Bucket Algorithm

[Levy+1996]

Given: user query  $q$ , source descriptions  $\{V_i\}$

- Find relevant sources (fill buckets)
 

For each relation  $g$  in query  $q$

  - Find  $V_j$  that contains relation  $g$
  - Check that constraints in  $V_j$  are compatible with  $q$
- Combine source relations  $\{V_j\}$  from each bucket into a conjunctive query  $q'$  and check for containment ( $q' \subseteq q$ )

Craig Knoblock University of Southern California 35

## The Bucket Algorithm: Example

$V1(\text{student, number, year}) \rightarrow \text{Registered}(\text{student, course, year}), \text{Course}(\text{course, number}), \text{number} \geq 500, \text{year} \geq 1992$   
 $V2(\text{student, dept, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{Enrolled}(\text{student, dept})$   
 $V3(\text{student, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{year} \leq 1990$   
 $V4(\text{student, course, number}) \rightarrow \text{Registered}(\text{student, course, year}), \text{Course}(\text{course, number}), \text{Enrolled}(\text{student, dept}), \text{number} \leq 100$

User Query (using mediator relations):  
 $q(S,D) :- \text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995.$

Craig Knoblock University of Southern California 36

# 1. Filling the Buckets

$V1(\text{student, number, year}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Course}(\text{course, number}), \text{number} \geq 500, \text{year} \geq 1992$   
 $V2(\text{student, dept, course}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Enrolled}(\text{student, dept})$   
 $V3(\text{student, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{year} \leq 1990$   
 $V4(\text{student, course, number}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Course}(\text{course, number}), \text{Enrolled}(\text{student, dept}), \text{number} \leq 100$   
 $q(S,D) :-$   
 $\text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   
 $V2(S,D,C')$   
 $V4(S,C',N')$

# 1. Filling the Buckets

$V1(\text{student, number, year}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Course}(\text{course, number}), \text{number} \geq 500, \text{year} \geq 1992$   
 $V2(\text{student, dept, course}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Enrolled}(\text{student, dept})$   
 $V3(\text{student, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{year} \leq 1990$   
 $V4(\text{student, course, number}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Course}(\text{course, number}), \text{Enrolled}(\text{student, dept}), \text{number} \leq 100$   
 $q(S,D) :-$   
 $\text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   
 $V2(S,D,C')$   $V1(S,N',Y)$   
 $V4(S,C',N')$   $V2(S,D',C)$   
 $V4(S,C,N')$

# 1. Filling the Buckets

$V1(\text{student, number, year}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Course}(\text{course, number}), \text{number} \geq 500, \text{year} \geq 1992$   
 $V2(\text{student, dept, course}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Enrolled}(\text{student, dept})$   
 $V3(\text{student, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{year} \leq 1990$   
 $V4(\text{student, course, number}) \rightarrow \text{Registered}(\text{student, course, year}),$   
 $\text{Course}(\text{course, number}), \text{Enrolled}(\text{student, dept}), \text{number} \leq 100$   
 $q(S,D) :-$   
 $\text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   
 $V2(S,D,C')$   $V1(S,N',Y)$   $V1(S',N,Y')$   
 $V4(S,C',N')$   $V2(S,D',C)$   
 $V4(S,C,N')$

# 2. Checking Containment

$q(S,D) :- \text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   
 $V2(S,D,C')$   $V1(S,N',Y)$   $V1(S',N,Y')$   
 $V4(S,C',N')$   $V2(S,D',C)$   
 $V4(S,C,N')$

# 2. Checking Containment

$q(S,D) :- \text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   
 $V2(S,D,C')$   $V1(S,N',Y)$   $V1(S',N,Y')$   
 $V4(S,C',N')$   $V2(S,D',C)$   
 $V4(S,C,N')$

# 2. Checking Containment

$q(S,D) :- \text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   
 $V2(S,D,C')$   $V1(S,N',Y)$   $V1(S',N,Y')$   
 $V4(S,C',N')$   $V2(S,D',C)$   
 $V4(S,C,N')$

$q'(S,D) :- V2(S,D,C'), V1(S,N',Y), V1(S',N,Y'), N \geq 300, Y \geq 1995$



## 2. Checking Containment

$q(S,D) :- \text{Enrolled}(S,D), \text{Registered}(S,C,Y), \text{Course}(C,N), N \geq 300, Y \geq 1995$   

 $V2(S,D,C')$     $V1(S,N',Y)$     $V1(S',N',Y')$   
 $V4(S,C',N')$     $V2(S,D',C)$   
 $V4(S,C,N')$

$q'(S,D) :- V2(S,D,C), V1(S,N',Y), V1(S',N',Y'), N \geq 300, Y \geq 1995$   
 $q'(S,D) :- V2(S,D,C), V1(S,N',Y), Y \geq 1995$

$\text{Registered}(S,C',Y) \wedge \text{Enrolled}(S,D) \wedge$   
 $\text{Registered}(S,C'',Y) \wedge \text{Course}(C'',N) \wedge N \geq 500 \wedge Y \geq 1992 \wedge$   
 $N \geq 300 \wedge Y \geq 1995 \rightarrow$   
 $\text{Enrolled}(S,D) \wedge \text{Registered}(S,C',Y) \wedge \text{Course}(C'',N) \wedge N \geq 500 \wedge Y \geq 1995 \rightarrow$   
 $\text{Enrolled}(S,D) \wedge \text{Registered}(S,C,Y) \wedge \text{Course}(C,N) \wedge N \geq 300 \wedge Y \geq 1995$

$\Rightarrow q' \subseteq q$  (and  $q'$  is a maximally-contained rewriting of  $q$ )

$V1(\text{student, number, year}) \rightarrow \text{Registered}(\text{student, course, year}), \text{Course}(\text{course, number}), \text{number} > 500, \text{year} > 1992$   
 $V2(\text{student, dept, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{Enrolled}(\text{student, dept})$   
 $V3(\text{student, course}) \rightarrow \text{Registered}(\text{student, course, year}), \text{year} < 1990$   
 $V4(\text{student, course, number}) \rightarrow \text{Registered}(\text{student, course, year}), \text{Course}(\text{course, number}), \text{Enrolled}(\text{student, dept}), \text{number} < 100$

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings
  - Local Completeness Information

## Inverse-Rules Algorithm

[Duschka+1997]

Idea: Construct an equivalent logic program whose evaluation yields the answer to the query

- The antecedent of the query and views is in term of mediator predicates
- Would like to have source predicates in antecedent so that program can be evaluated

$\Rightarrow$  Invert the rules

(simply by using standard logical manipulations)

## The Inverse-Rules Algorithm: Example

$V1(\text{dept, course}) \rightarrow \text{Enrolled}(\text{student, dept}) \wedge \text{Registered}(\text{student, course})$

$\forall D,C [V1(D,C) \rightarrow \exists S [e(S,D) \wedge r(S,C)]]$   
 $\equiv \neg \forall 1(D,C) \vee [e(f(D,C),D) \wedge r(f(D,C),C)]$   
 $\equiv [\neg \forall 1(D,C) \vee e(f(D,C),D)] \wedge [\neg \forall 1(D,C) \vee r(f(D,C),C)]$   
 $\equiv [V1(D,C) \rightarrow e(f(D,C),D)] \wedge [V1(D,C) \rightarrow r(f(D,C),C)]$   
 $\equiv$   
 $e(f(D,C),D) \leftarrow V1(D,C)$   
 $r(f(D,C),C) \leftarrow V1(D,C)$

$a \rightarrow b \equiv \neg a \vee b$

## The Inverse-Rules Algorithm: Example

$q(D) \leftarrow \text{Enrolled}(S,D) \wedge \text{Registered}(S, \text{"DB"})$   
 $v1(D,C) \rightarrow \text{Enrolled}(S,D) \wedge \text{Registered}(S,C)$

$q(D) \leftarrow \text{Enrolled}(S,D) \wedge \text{Registered}(S, \text{"DB"})$   
 $\text{Enrolled}(f(D,C),D) \leftarrow v1(D,C)$   
 $\text{Registered}(f(D,C),C) \leftarrow v1(D,C)$   
 $q(D) \leftarrow v1(D, \text{"DB"})$

$\text{Ext}(v1) = \{(\text{"CS"}, \text{"DB"}), (\text{"EE"}, \text{"DB"}), (\text{"CS"}, \text{"AI"})\}$   
 $\text{Ext}(q) = \{(\text{"CS"}), (\text{"EE"})\}$

## Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings

## Modeling Source Capabilities

Negative capabilities:

- A web-site may require certain inputs (in an HTML form) to answer a query
- Need to consider only valid query execution plans

Positive capabilities:

- A source may be database (understands SQL)
- Need to decide the placement of operations according to capabilities

Problem: how to describe and exploit source capabilities

## Negative Capabilities: Binding Patterns

Sources:

$AAAIdb^f(X) \rightarrow AAAIPapers(X)$

$CitationDB^{bf}(X,Y) \rightarrow Cites(X,Y)$

$AwardDB^b(X) \rightarrow AwardPaper(X)$

Query: find all the award winning papers:

$q(X) \leftarrow AwardPaper(X)$

## Recursive Rewritings

$q(X) \leftarrow AwardPaper(X)$

- Problem: *Unbounded* union of conjunctive queries

$q1(X) \leftarrow AAAIdb(X), AwardDB(X)$

$q1(X) \leftarrow AAAIdb(X1), CitationDB(X1,X), AwardDB(X)$

...

$q1(X) \leftarrow AAAIdb(X1), CitationDB(X1,X2), \dots, CitationDB(Xn,X), AwardDB(X)$

- Solution: Recursive Rewriting

$papers(X) \leftarrow AAAIdb(X)$

$papers(X) \leftarrow papers(Y), CitationDB(Y,X)$

$q(X) \leftarrow papers(X), AwardDB(X)$

$AAAIdb^f(X) \rightarrow AAAIPapers(X)$ $CitationDB^{bf}(X,Y) \rightarrow Cites(X,Y)$ $AwardDB^b(X) \rightarrow AwardPaper(X)$
--

## Inverse-Rules Algorithm Binding Patterns

Sources:

$AAAIdb^f(X) \rightarrow AAAIPapers(X)$

$CitationDB^{bf}(X,Y) \rightarrow Cites(X,Y)$

$AwardDB^b(X) \rightarrow AwardPaper(X)$

Query: find all the award winning papers:

$q(X) \leftarrow AwardPaper(X)$

## Inverse-Rules Algorithm Inverse + Domain Rules (1)

Inverted Rules:

$AAAIPapers(X) \leftarrow AAAIdb(X)$

$Cites(X,Y) \leftarrow dom(X) \wedge CitationDB(X,Y)$

$AwardPaper(X) \leftarrow dom(X) \wedge AwardDB(X)$

Domain Rules:

$dom(Y) \leftarrow dom(X) \wedge CitationDB(X,Y)$

$dom(X) \leftarrow AAAIdb(X)$

Query:

$q(X) \leftarrow AwardPaper(X)$

## Inverse-Rules Algorithm Inverse + Domain Rules (2)

Simplifying the program:

$q(X) \leftarrow dom(X) \wedge AwardDB(X)$

$dom(Y) \leftarrow dom(X) \wedge CitationDB(X,Y)$

$dom(X) \leftarrow AAAIdb(X)$



## Summary

- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
  - Source Capabilities: Recursive Rewritings