

Managing Inconsistencies in Collaborative Data Management

Eric Kao
Logic Group
Computer Science Department
Stanford University

Talk given at HP Labs on November 9, 2010

Structured Data

Public Sources

Company Directories

Product Catalogs

Airline Schedules

Weather Reports

Patient Records

Drug Studies

Enterprise Sources

Personnel Records

Equipment Databases

Room Schedules

Orders

Inventories

Relationships Among Sources

Replicated data

 Cached data

 Materialized views (as in data warehouses)

Heterogeneity (different schemas or vocabularies)

 values in euros vs values in dollars

 French instead of English

 different numbers of tables or attributes

Real World Constraints

 Physical laws

 Governmental laws

 Business rules

Update

In the face of interrelationships among sources, updates to one source may necessitate updates to other sources in order to preserve their relationships.

Update Integration is the process of updating interrelated data sources in an integrated fashion. Analogous to **Data Integration** for data consumers.

Collaboration

In large enterprises, consortia, and open information systems (e.g. the WWW), it is common for different data sources to be controlled by different people. In such cases, the individuals performing updates must collaborate (explicitly or implicitly).

Collaborative Data Management (CDM) is update integration in a multiple user setting.

Inconsistencies

- In CDM, we capture the relationships between data sources as constraints written in a logical language
 - Not all relationships captured, but many important ones
- When multiple users update data sources independently, the constraints may be violated.
- We called these violations **inconsistencies**.

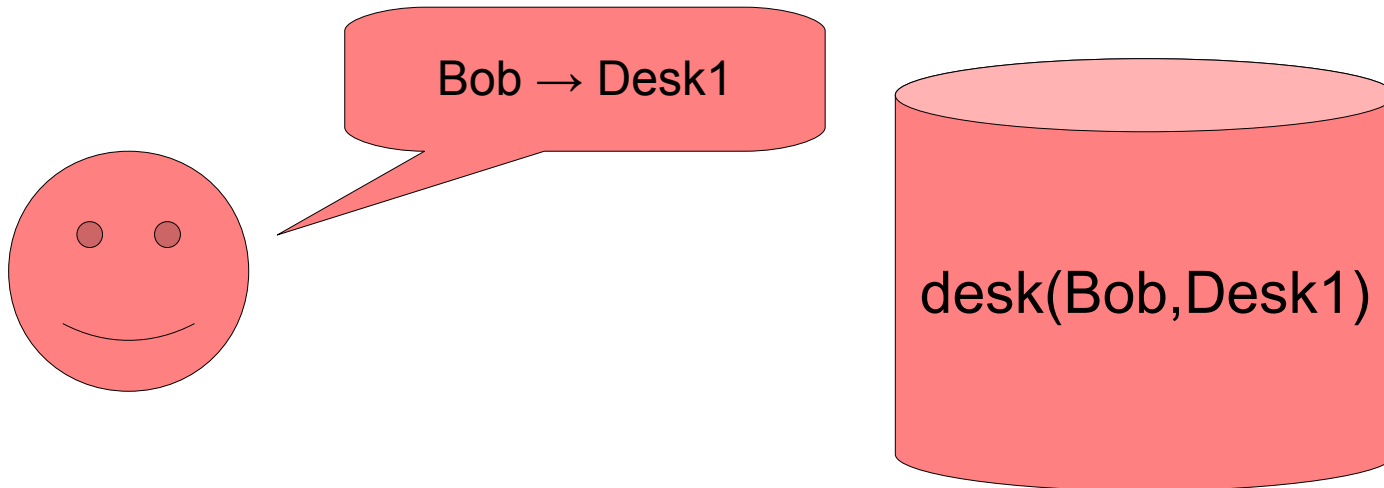
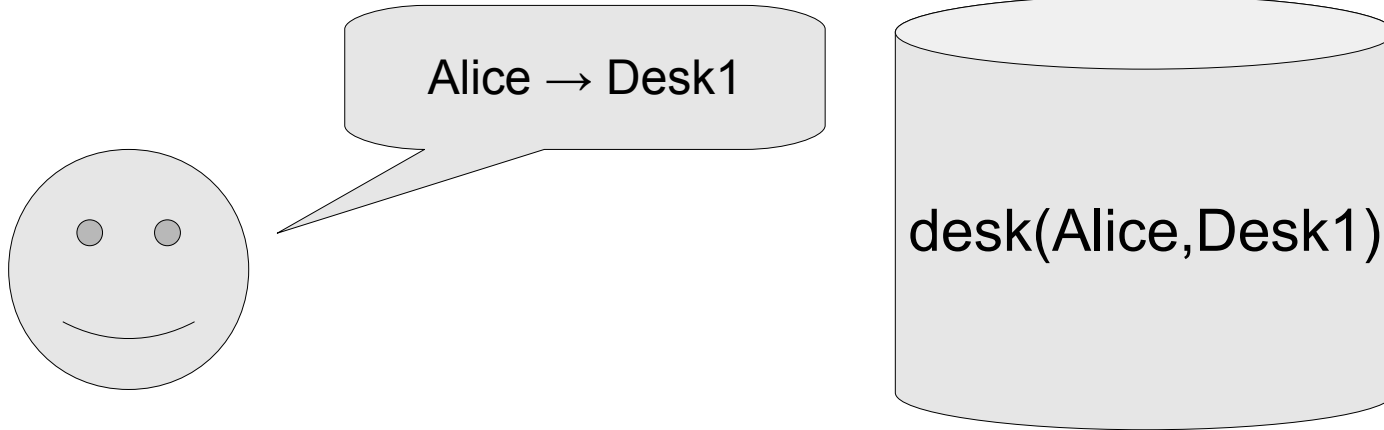
Inconsistency Management

- Identifying and indicating inconsistencies.
- Automatically updating some data sources to avoid inconsistencies.
- Answering queries in the presence of inconsistencies.

Decentralized Office assignment

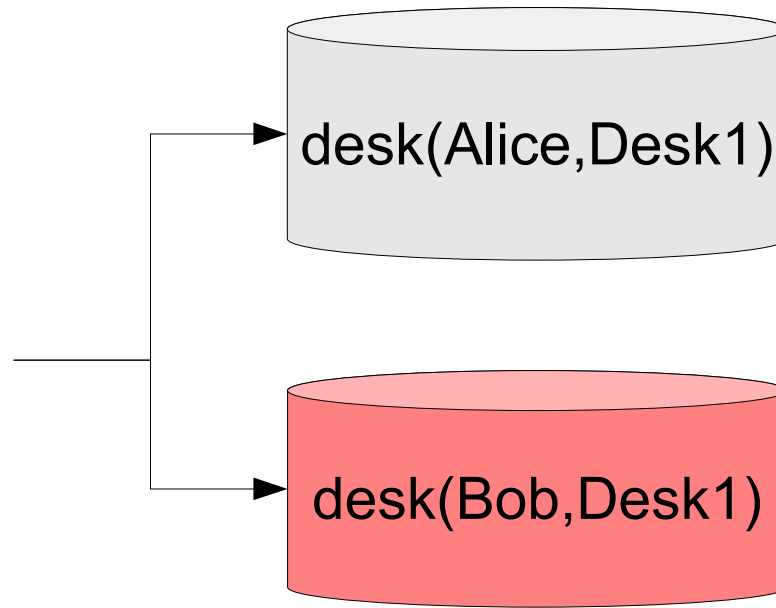
- A building is divided into spaces.
- Each space as a “space czar” who is tasked with assigning people to offices within that space.
- The building manager, department manager, and department chair also exercise authority over the assignment of offices.
- Individual professors request specific assignments for his or her affiliates.

Desk Assignment Example



Desk Assignment Example

Conflict:
Data is inconsistent
with the following
constraint



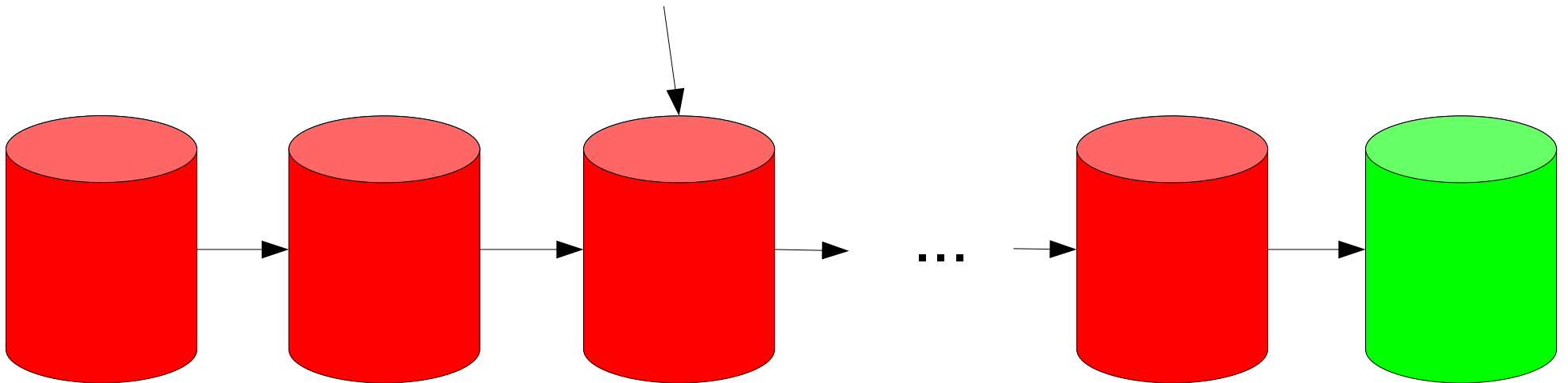
$$\begin{aligned} &\forall x1, x2, y \\ &\quad (\text{person.desk}(x1,y) \wedge \text{person.desk}(x2,y) \\ &\quad \rightarrow x1 = x2) \end{aligned}$$

Alert the users to the conflict

- The users can work to resolve the conflict

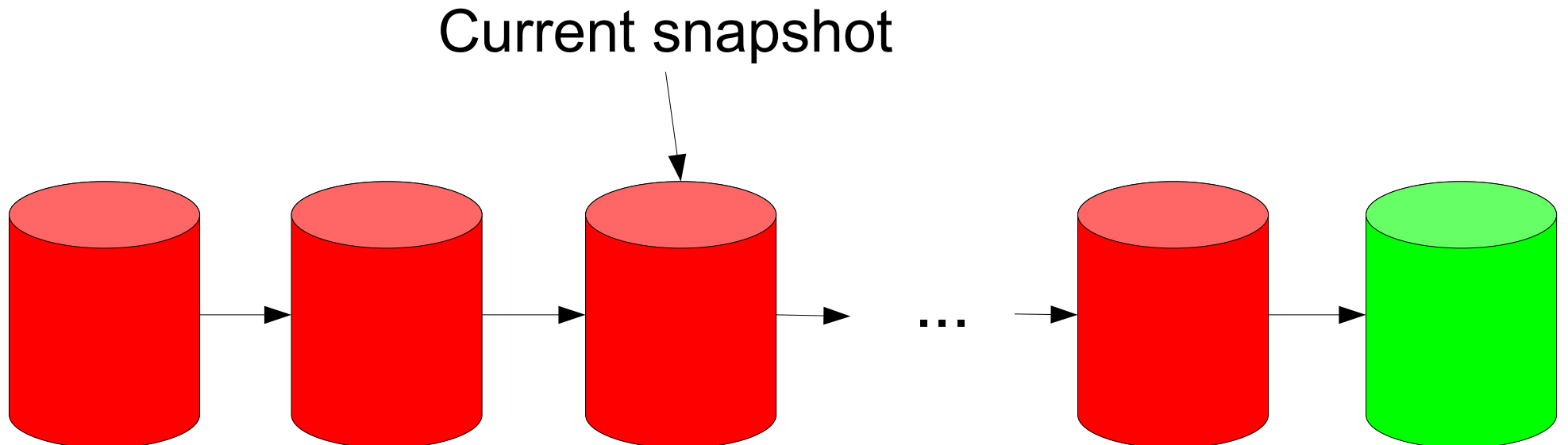
person.desk		
+ - 🔄	alice	desk1
+ - 🔄	bob	desk1
+ - 🔄	carol	desk2
+ - 🔄	david	desk3
+		

Current snapshot

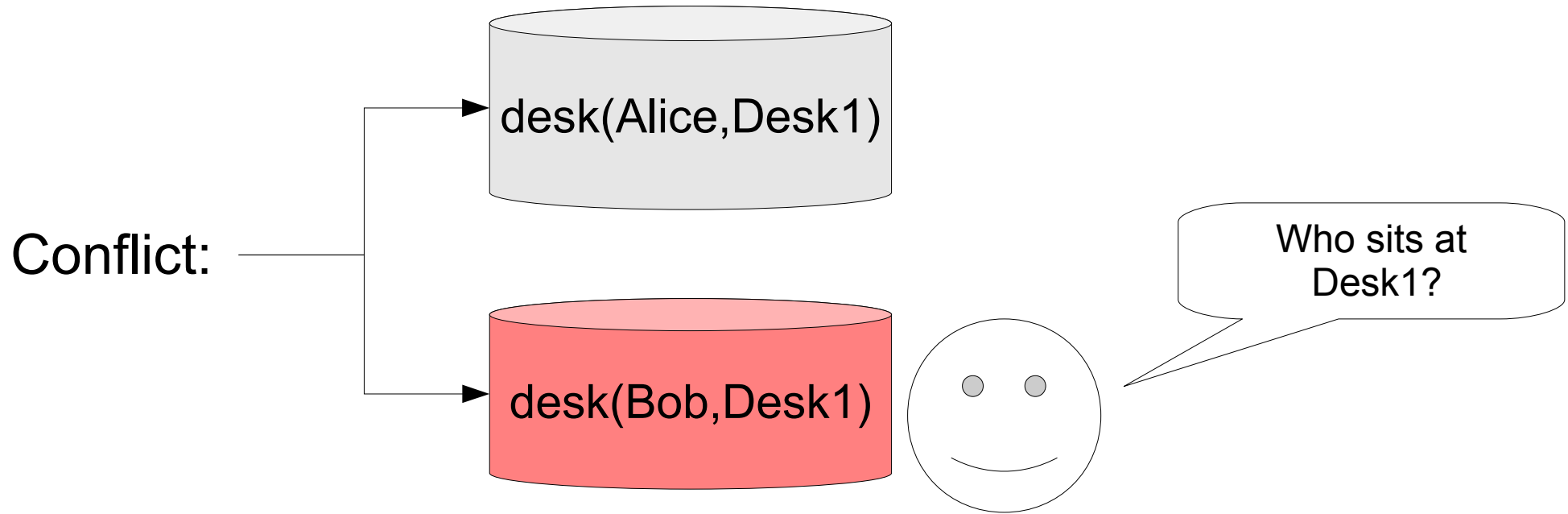


Queries

- Users would like to query the eventual, consistent data
- But all we have is the current, inconsistent, snapshot.



Queries



- Best guess: Alice or Bob, but not both!

Relational Calculus Syntax

base formulas are atoms over S and equality (disequality) atoms of the form $e = e'$ ($e \neq e'$) for terms e and e' . The well-formed formulas of the relational calculus over S include the base formulas and formulas of the form

1. $(\phi \wedge \psi)$, where ϕ and ψ are formulas over S ;
2. $(\phi \vee \psi)$, where ϕ and ψ are formulas over S ;
3. $\neg\phi$, where ϕ is a formula over S ;
4. $\exists x\phi$, where x is a variable ϕ is a formula over S ;
5. $\forall x\phi$, where x is a variable ϕ is a formula over S .

Database, Constraints

- Database instance
A finite set of ground atoms
 - e.g., $D = \{ p(a,a), p(b,c), q(a), r(a,d) \}$
- Constraint set
A finite set of relational calculus sentences
(no free variables)
 - e.g., $C = \left\{ \begin{array}{l} \forall x (\neg p(x,x) \vee q(x)), \\ \exists x q(x) \end{array} \right\}$

Database Query

- Query

A relational calculus formula

- e.g., $Q_1 = p(a,b)$
- e.g., $Q_2(x) = \exists y p(x,y)$
- e.g., $Q_3(x) = \exists z \forall y (\neg p(x,y) \vee r(y,z))$

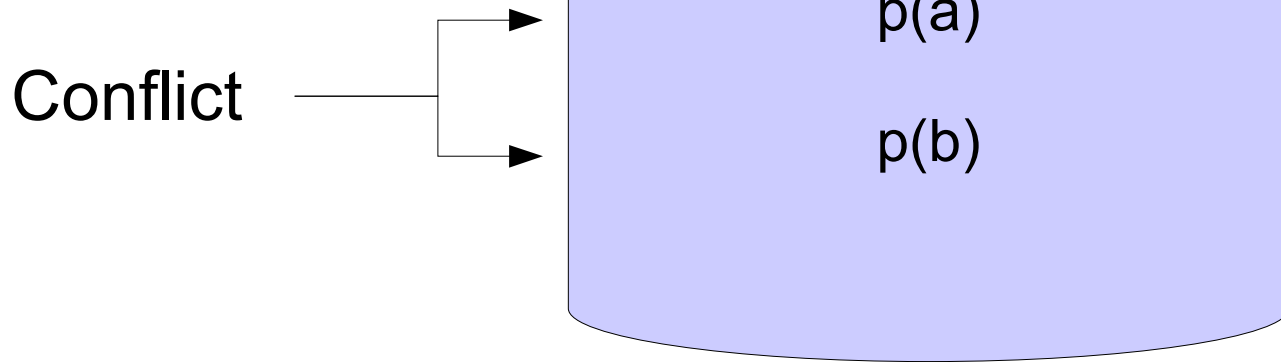
Min-change Repairs

- Given:
 - D : database e.g., $\{ p(a), p(b) \}$
 - Ω : set of constraints e.g., $\{ \neg p(a) \vee \neg p(b) \}$
- A repair is a database D^* that is consistent with Ω .
e.g., $\{ p(a) \}$, $\{ p(b) \}$, $\{ \}$, $\{ p(a), p(c) \}$, ...
- A minimal change repair is a repair that is minimally different from D .
e.g., $\{ p(a) \}$, $\{ p(b) \}$

Consistent query answers

- CQA: answers that arise no matter how the conflicts are resolved (in a minimal-change fashion) [ABC]
- $D \models_{\Omega}^{\text{CQA}} Q(t)$ iff
for all minimal-change repairs D^* of D w.r.t. to Ω , $D^* \models Q(t)$

CQA



Constraint:
 $\neg p(a) \vee \neg p(b)$

- $p(a)$? NO {p(b)}
- $p(b)$? NO {p(a)}
- $p(a) \wedge p(b)$? NO {p(b)}
- $p(a) \vee p(b)$? YES {p(a)}, {p(b)}

More lenient semantics

- In some situations, we would like to see plausible though non-guaranteed answers
 - e.g., a student checking out whom his office mates might be
- Possible answers from repairs: answers that arise from some minimal-change repair

Relax, not repair

- Constraint: every person must be assigned a desk
- Data: Charlie is missing a desk assignment
- Minimal-change repairs:
{desk(Charles,desk1)}, {desk(Charles,desk2)},
{desk(Charles,desk3)}, ...
- Possible answers:
desk1, desk2, desk3, ...

Incomplete Database

- An incomplete database is a pair $\langle P, N \rangle$ where
 - P, N are sets of ground atoms
 - $P \cap N = \{\}$
- e.g., $K = \langle \{p(a), q(a)\}, \{p(c), q(b)\} \rangle$
- Intuitively,
 - P gives the known true atoms
 - N gives the known false atoms
 - The rest are unknown
- $\langle P, N \rangle \models_d Q$ iff $D \models_d Q$ for all $D \in \text{Instances}_d(\langle P, N \rangle)$

Relaxations of a database

- A relaxation of D w.r.t. Ω is an incomplete database $K = \langle P, N \rangle$, where
 - $P, N \subseteq \text{base}(\mathbf{d})$
 - $P \subseteq D$
 - $N \cap D = \emptyset$
 - $\langle P, N \rangle \not\models_{\mathbf{d}} \neg \Omega$
- Let the set of relaxations of D w.r.t. to Ω be denoted $\text{Re}_{\mathbf{d}}(D, \Omega)$

Example

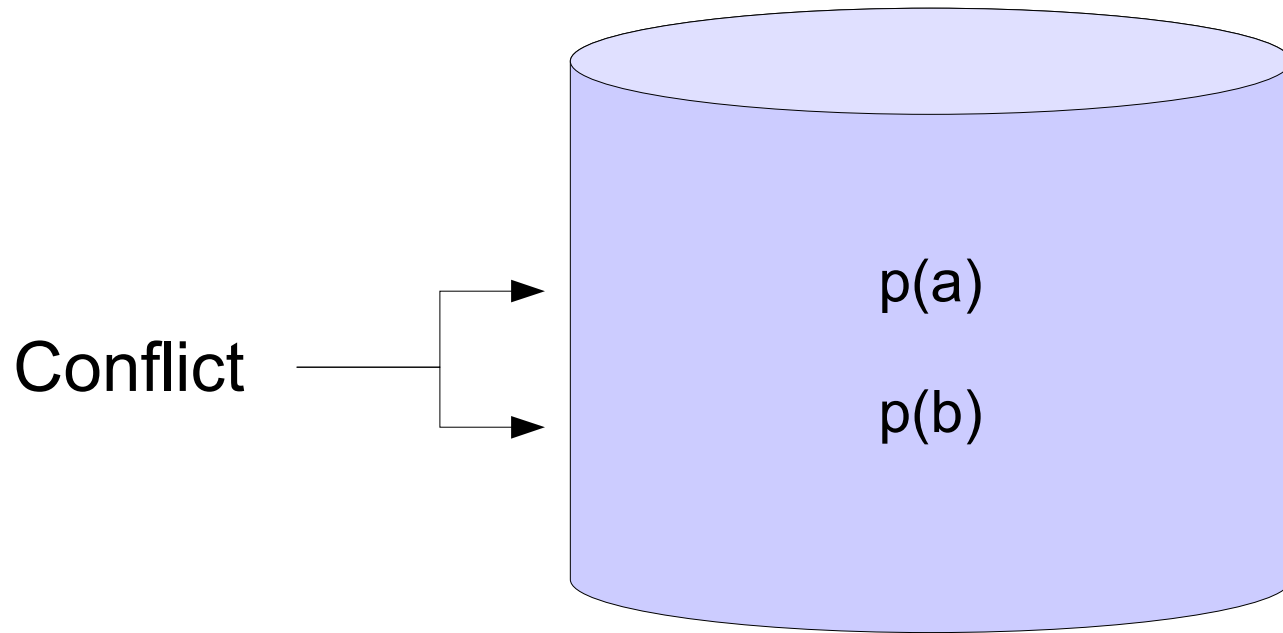
- $D = \{p(a), p(b)\}$
- $\Omega = \{\neg p(a) \vee \neg p(b)\}$
- $\mathbf{d} = \{a, b, c\}$
- $\text{Re}_{\mathbf{d}}(D, \Omega) = \{$

$\langle \{p(a)\}, \{p(c)\} \rangle,$	$\langle \{p(b)\}, \{p(c)\} \rangle$
$\langle \{p(a)\}, \{\} \rangle,$	$\langle \{p(b)\}, \{\} \rangle$
$\langle \{\}, \{p(c)\} \rangle,$	$\langle \{\}, \{\} \rangle$

Answers with Consistent Support

- ACS: answers that are supported by a consistent relaxation of the database. [KG]
- $D \models_{\Omega}^{CS} Q(t)$ iff there exists $K \in \text{Re}_d(D, \Omega)$ st $K \models_d Q(t)$
 - Where $d = \text{adom}(D, Q)$

ACS



Constraint:
 $\neg p(a) \vee \neg p(b)$

- $p(a)$? YES $\langle \{p(a)\}, \{\} \rangle$
- $p(b)$? YES $\langle \{p(b)\}, \{\} \rangle$
- $p(a) \wedge p(b)$? NO
- $p(a) \vee p(b)$? YES $\langle \{p(a)\}, \{\} \rangle$

More complex queries

p		
	a	1
	b	1
	c	2
	d	3

q		r	
	a		c
	b		d

Constraint:

$$\forall xyz (\neg p(x,z) \vee \neg p(y,z) \vee x=y)$$

- $\forall x [q(x) \rightarrow p(x,1)]$?
 - ACS: NO

Data Complexity

- Both ACS and CQA are NP-Hard problems [KG] [CM]
 - Reduction from monotone 3-SAT
- ACS is tractable for $\exists^* \forall^*$ queries†. [KG]
- CQA is tractable for
 - \forall^* queries† [KG]
 - Some subclasses of \exists^* queries† [ABC, CM, FM]

† with suitable restrictions on the class of constraints

Query rewriting

- How can we use existing database infrastructure to support these query semantics?
- Solution: Query rewriting
 - Algorithm ACS-Rewrite[Q,Ω]
Rewrite Q ($\exists^* \forall^*$) into Q' so that evaluating Q' on a standard DBMS gives ACS answers to Q. [KG]
 - Similar rewriting algorithms have been devised for CQA [ABC, CM, FM, KG]

Results

- Tractable subclasses identified for ACS

query \ constraints	Ground	Finitely-closed A^*	A^*
E^*	P	P	P
E^*A^*	P	P	Open
A^*E^*	P	NP-Complete	NP-Complete
CALC	P	NP-Complete	NP-Complete

† Finitely-closed means the set of constraints has a finite closure under resolution.

ACS: Ground conjunctive queries

- $Q = p(a,1) \wedge p(b,1)$
- $C = \{ \forall x_1 x_2 y (\neg p(x_1,y) \vee \neg p(x_2,y) \vee x_1=x_2) \}$
- Q contradicts C .
- Answer: NO
- Rewrite:
 $Q' = \text{False}$

ACS: qf conjunctive queries

- $Q(x_1, y_1, x_2, y_2) = p(x_1, y_1) \wedge p(x_2, y_2)$
- $C = \{ \forall x_1 x_2 y (\neg p(x_1, y) \vee \neg p(x_2, y) \vee x_1 = x_2) \}$
- $Q(x_1, y_1, x_2, y_2)$ is consistent w.r.t. C if and only if $(x_1 = x_2 \vee y_1 \neq y_2)$
- Rewrite:
 $Q' = Q(x_1, x_2, y_1, y_2) \wedge (x_1 = x_2 \vee y_1 \neq y_2)$

ACS: \forall^* queries

- $Q = \forall x,y (\neg q(x,y) \vee \neg r(x,y))$
- $C = \{ \forall x q(x,x) \}$
- $\neg q(x,y) \vee \neg r(x,y)$
 $q(x,x)$

$\neg r(x,x)$

- Rewrite:
 $Q' = Q \wedge \forall x \neg r(x,x)$

Ongoing work

- Identify other tractable classes
- Recursive queries and aggregates
- Other answer semantics
 - preferentially-defeasible data?
 - coarser granularity?
 - Semantics that consider history?

Issues

Constraint Language

- Expressiveness (Skolems, aggregates)

- Evaluation and Rewriting Techniques

Managing Inconsistency

- Inconsistencies and interdependencies

- Automatic update in the presence of inconsistencies

- Queries in the presence of inconsistencies

Dynamic Constraints

- Update Preferences

Workflow Analysis, Use, Design

- Privacy and Security

- Promoting Convergence

References

- [ABC] M. Arenas, L. Bertossi, and J. Chomicki. *Consistent query answers in inconsistent databases*. Symposium on principles of database systems, Pages 68–79, 1999.
- [CM] J. Chomicki and J. Marcinkowski. *Minimal-change integrity maintenance using tuple deletions*. Inf. Comput., 197(1-2):90–121, 2005.
- [FM] A. Fuxman and R. J. Miller. *First-order query rewriting for inconsistent databases*. J. Comput. Syst. Sci., 73(4):610–635, 2007.
- [KG] E. Kao and M. Genesereth. *Answers with consistent support – complexity and rewriting*. [Working paper](#), 2010.