Towards Practical Computation of Consistent Query Answers

Jan Chomicki
Dept. CSE
University at Buffalo
State University of New York
`http://www.cse.buffalo.edu/~chomicki`

Joint work with Jurek Marcinkowski and Slawek Staworko.

Integrity constraints

Integrity constraints describe valid database instances.

Here:

- functional dependencies: *"every student has a single address."*
- denial constraints: *"no employee can make more than her manager."*
- referential integrity: *"students can enroll only in the offered courses."*

The constraints are formulated in first-order logic:

$$\forall n, s, m, s', m'. \neg[Emp(n, s, m) \land Emp(m, s', m') \land s > s'].$$

Inconsistent databases

There are situations when we want/need to live with inconsistent data in a database (data that violates given integrity constraints):

- integration of heterogeneous databases with overlapping information
- the consistency of the database will be restored by executing further transactions
- inconsistency wrt "soft" integrity constraints (those that we hope to see satisfied but do not/cannot check) process

How to distinguish between reliable and unreliable information in an inconsistent database?

Goals

A formal definition of reliable ("consistent") information in an inconsistent database.

Computational mechanisms for obtaining consistent information.

Computational complexity analysis.

Implementation:

- preferably using DBMS technology

Plan of the talk

1. repairs and consistent query answers
2. computing consistent query answers to quantifier-free queries
3. why quantification is difficult
4. related and further work

Consistent query answers

[Arenas, Bertossi, Chomicki, PODS'99]

Repair:

- a database that satisfies the integrity constraints
- difference from the given database is minimal (the set of inserted/deleted facts is minimal under set inclusion)

A tuple $(a_1, \ldots, a_n)$ is a consistent query answer to a query $Q(x_1, \ldots, x_n)$ in a database r if it is an element of the result of Q in every repair of r.

Geography Bee database

GeoBee

Continent	*Discoverer*	*LandArea*
N. America	C. Columbus	24M
N. America	L. Ericson	24M
Australia	J. Cook	8M

Functional dependency:

$Continent \longrightarrow Discoverer$

Repairs:

N. America	C. Columbus	24M
Australia	J. Cook	8M

N. America	L. Ericson	24M
Australia	J. Cook	8M

Query languages and integrity constraints

Ultimately: SQL2.

Here:

- quantifier-free queries (equivalently: relational algebra without projection)
- denial integrity constraints:

$$\forall \neg (P_1(\bar{x}_1) \wedge \cdots \wedge P_n(\bar{x}_n) \wedge \phi)$$

Consistent query answers

$GeoBee(C, D, A)$ $\Rightarrow$

Australia	J. Cook	8M

$GeoBee(C, \text{'L.Ericson'}, 24M) \vee GeoBee(C, \text{'C.Columbus'}, 24M)$ $\Rightarrow$

N. America

$\exists D.\ GeoBee(C, D, A)$ $\Rightarrow$

N. America	24M
Australia	8M

There are too many repairs to evaluate the query in each of them.

A	B
a_1	b_1
a_1	b'_1
a_2	b_2
a_2	b'_2
$\ldots$	
a_n	b_n
a_n	b'_n

Under the functional dependency $A \rightarrow B$, this instance has 2^n repairs.

Computing consistent query answers

Query transformation: given a query Q and a set of integrity constraints, construct a query Q' such that for every database instance r

> the set of answers to Q' in $r =$ the set of consistent answers to Q in r.

Representing all repairs: given a set of integrity constraints and a database instance r:

1. construct a space-efficient representation of all repairs of r
2. use this representation to answer (many) queries.

Specifying repairs as logic programs.

Conflict hypergraph

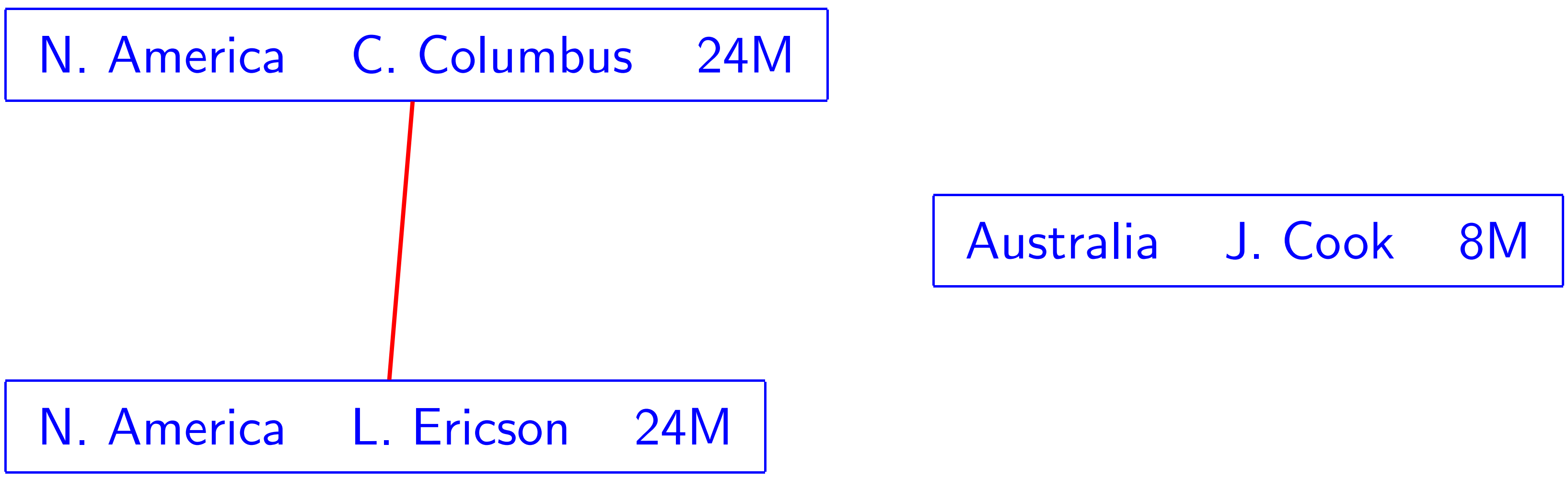
Vertices:

- facts in the original instance.

Edges:

- (minimal) sets of facts that violate some constraint.

Repair: a maximal independent set.


N. America C. Columbus 24M
Australia J. Cook 8M
N. America L. Ericson 24M

Ground queries

Observations:

- the query is in CNF $\Rightarrow$ each conjunct can be processed separately
- all repairs satisfy $\Phi \Leftrightarrow$ no repair satisfies $\neg\Phi$

Algorithm HProof:

1. $\neg\Phi = P_1(t_1) \wedge \cdots \wedge P_m(t_m) \wedge \neg P_{m+1}(t_{m+1}) \wedge \cdots \wedge \neg P_n(t_n)$
2. find a repair including $P_1(t_1), \ldots, P_m(t_m)$ and excluding $P_{m+1}(t_{m+1}), \ldots, P_n(t_n)$ by enumerating the appropriate edges.

Excluding a fact A:

- A is not in the original instance, or
- A belongs to an edge $\{A, B_1, \ldots, B_k\}$ in the conflict hypergraph and $B_1, \ldots, B_k$ belong to the repair.

Properties of HProof

Algorithm HProof works in PTIME (data complexity):

- $n - m$ choices from a set of polynomial size
- if all choices successful, a repair can be completed.

Generalizing to open queries:

- how to generate bindings for free variables?

In the presence of negation (set difference), there may be consistent query answers which are not query answers in the original instance.

Database schema: $R(AB), \ , S(ABC)$.

Integrity constraint over R: $A \rightarrow B$.

Query: $S - (R(A, B_1) \underset{B_1 \neq B_2}{\bowtie} R(A, B_2))$.

Instance: $\{R(a, b), R(a, c), S(a, b, c)\}$.

Query answers: $\emptyset$

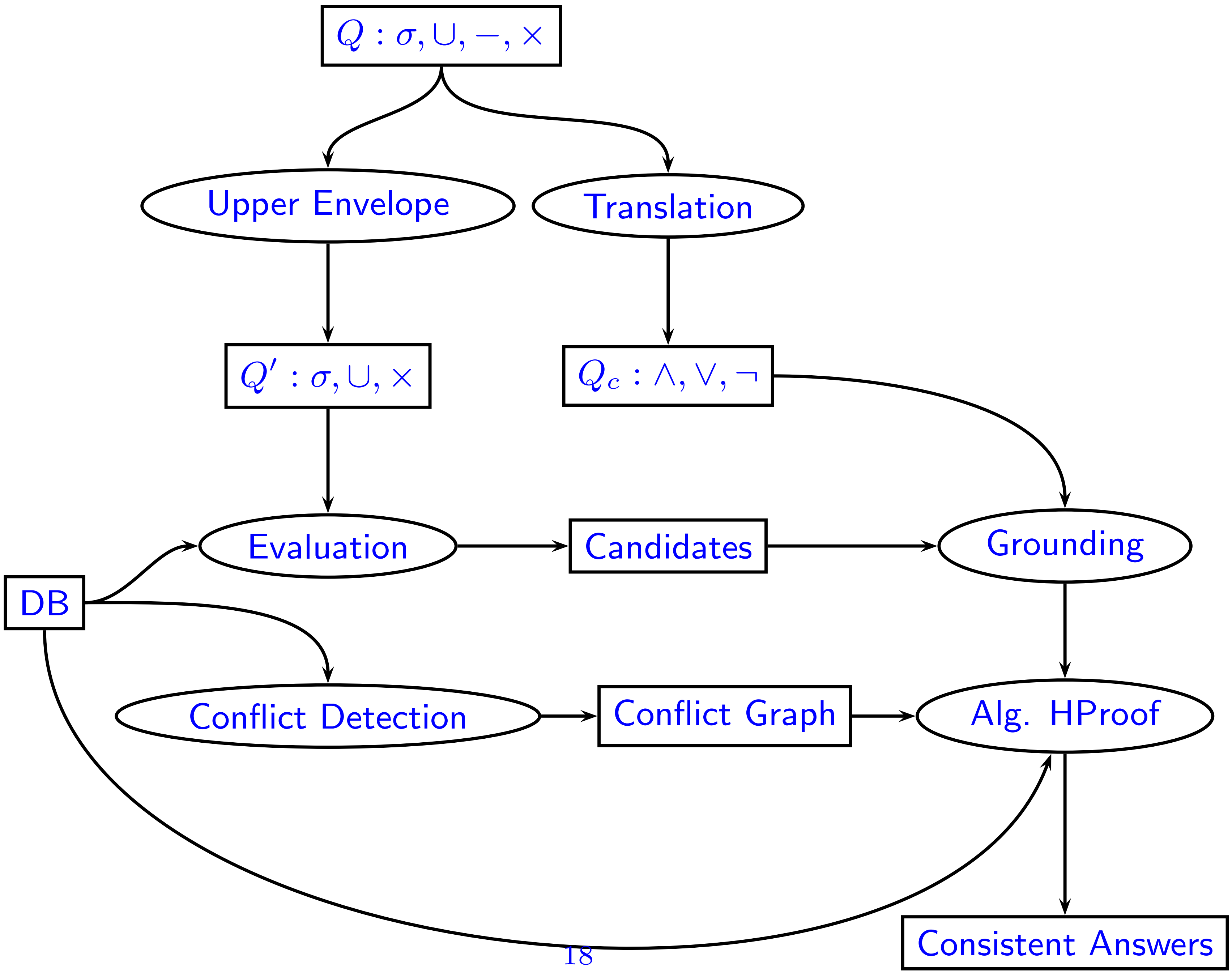
Consistent query answers: $\{(a, b, c)\}$

Upper envelope

Construct an upper envelope query $U(Q)$ such that the set of answers to $U(Q)$ in r is

- a superset of the set of consistent answers to Q in r
- a superset of the set of answers to Q in r.

$$
\begin{aligned}
U(R) &= R \\
U(E_1 \cup E_2) &= U(E_1) \cup U(E_2) \\
U(E_1 \times E_2) &= U(E_1) \times U(E_2) \\
U(\sigma_\chi(E)) &= \sigma_\chi(U(E)) \\
U(E_1 - E_2) &= U(E_1).
\end{aligned}
$$


$Q : \sigma, \cup, -, \times$
Upper Envelope
Translation
$Q' : \sigma, \cup, \times$
$Q_c : \wedge, \vee, \neg$
Evaluation
Candidates
Grounding
DB
Conflict Detection
Conflict Graph
Alg. HProof
Consistent Answers

Comparison with query transformation

	Query transformation (QT)	Conflict hypergraph (CH)
Integrity constraints	Binary universal	Denial
Queries	$\sigma, \times, -$	$\sigma, \times, -, \cup$

Preliminary experimental results:

- optimized query QT (outerjoins) generally faster than optimized CH
- time required by CH grows slower with the instance size than the time required by QT

Further extensions

Beyond denial constraints:

- how to compactly represent all repairs?
- the same approach works if non-denial constraints can be repaired first:
  - key and foreign constraints, with one key per relation

Quantifiers in queries:

- co-NP-hardness

Data complexity of consistent query answers

Chomicki, Marcinkowski [submitted]:

Queries	Functional dependencies		Denial constraints
	$\|F\| = 1$	$\|F\| \geq 2$	
$\sigma, \times, -, \cup$	PTIME	PTIME	PTIME
$\pi, \sigma, \times$ (no join)	PTIME	co-NP-complete	co-NP-complete
$\pi, \sigma, \times$ (join)	co-NP-complete	co-NP-complete	co-NP-complete

Specifying repairs as logic programs

Arenas, Bertossi, Chomicki [FQAS'00]:

- using logic programs negation and disjunction
- implemented using main-memory LP systems (`dlv`, Smodels)
- Π_2^p-complete problems

Scope:

- arbitrary universal constraints, inclusion dependencies
- arbitrary first-order queries
- queries can be "modalized" and nested

Also Greco and Zumpano [LPAR'00, ICLP'01] and Barcelo and Bertossi [NMR'02, PADL'03].

Related work

Belief revision:

- revising database with integrity constraints
- revised theory changes with each database update
- emphasis on semantics (AGM postulates), not computation
- complexity results [Eiter, Gottlob, AI'92] do not quite transfer

Disjunctive information:

- repair $\equiv$ possible world (sometimes)
- using disjunctions to represent resolved conflicts
- query languages: representation-specific, relational algebra or calculus
- complexity results [Imielinski et al., JCSS'95] do not quite transfer

Future work

Broadening scope:

- SQL:
  - relational algebra and aggregation
  - integrating different techniques
  - keys and foreign keys
- preferences:
  - source rankings
  - timestamps
- alternative semantics:
  - repairing attribute values [Wijsen, ICDT'03]
  - minimum-cardinality changes

New paradigms:

- data integration and exchange:
  - Bertossi, Chomicki, Cortes, Gutierrez [FQAS'02]
  - Bravo, Bertossi [IJCAI'03]
  - Cali, Lembo, Rosati [PODS'03]
- data cleaning
- *evidence* databases
- XML
- spatial/spatiotemporal databases

Selected papers:

1. M. Arenas, L. Bertossi, J. Chomicki, *"Consistent Query Answers in Inconsistent Databases,"* ACM Symposium on Principles of Database Systems (PODS), Philadelphia, May 1999.
2. M. Arenas, L. Bertossi, J. Chomicki, *"Specifying and Querying Database Repairs using Logic Programs with Exceptions,"* International Symposium on Flexible Query Answering Systems (FQAS), Warsaw, Poland, October 2000. Full version: *Theory and Practice of Logic Programming*, 2003.
3. M. Arenas, L. Bertossi, J. Chomicki, *"Scalar Aggregation in FD-Inconsistent Databases,"* International Conference on Database Thory (ICDT), London, UK, January 2001. Full version: *Theoretical Computer Science*, 2003.
4. J. Chomicki, J. Marcinkowski, *"Minimal-Change Integrity Maintenance Using Tuple Deletions,"* submitted.
5. L. Bertossi, J. Chomicki, *"Query Answering in Inconsistent Databases,"* in *Logics for Emerging Applications of Databases,* J. Chomicki, R. van der Meyden, G. Saake [eds.], Springer-Verlag, 2003, to appear.