A link between inconsistent databases and CSP

Gaëlle Fontaine

joint work with Balder ten Cate and Phokion Kolaitis

University of California, Santa Cruz

Inconsistent databases

- databases are finite models and queries are formulas
  example: conjunctive queries are FO formulas built from $\wedge$ and $\exists$

- given a database I and a query $q(\mathbf{x})$, the answer of q on I is given by

$$q(I) = \{\mathbf{a} : I \models q(\mathbf{a})\}$$

Inconsistent databases

- databases are finite models and queries are formulas
  example: conjunctive queries are FO formulas built from $\wedge$ and $\exists$
- given a database I and a query $q(\mathbf{x})$, the answer of q on I is given by

$$q(I) = \{\mathbf{a} : I \models q(\mathbf{a})\}$$

- constraints are semantic properties the database should obey
  a set of constraints Σ is a set of formulas
- a database I is consistent (w.r.t. Σ) if $I \models \Sigma$

How to handle inconsistency?

Not all databases are consistent: inconsistency arises in a variety of contexts such as data integration, data warehousing.

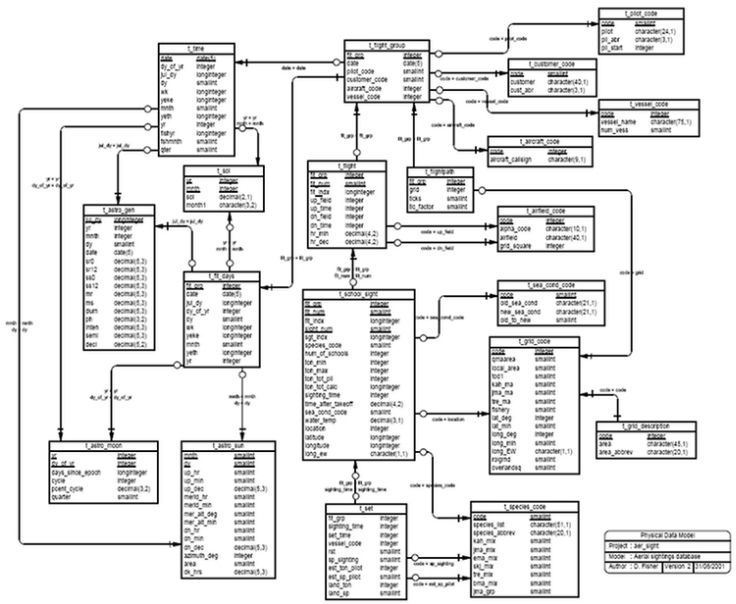


Figure 1: Entity Relationship Diagram (ERD) for the aer_sight database

How to handle inconsistency?

Not all databases are consistent: inconsistency arises in a variety of contexts such as data integration, data warehousing.

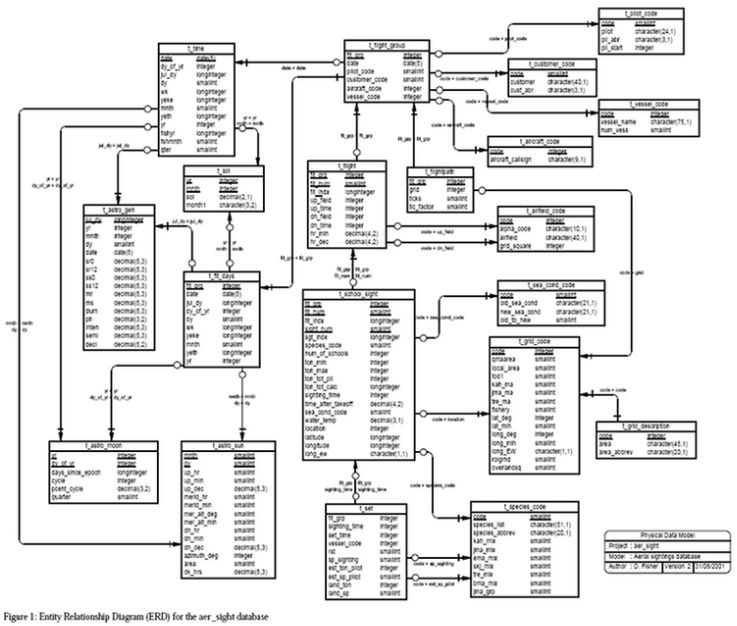


Figure 1: Entity Relationship Diagram (ERD) for the aer_sight database

How to handle inconsistency?

Not all databases are consistent: inconsistency arises in a variety of contexts such as data integration, data warehousing.

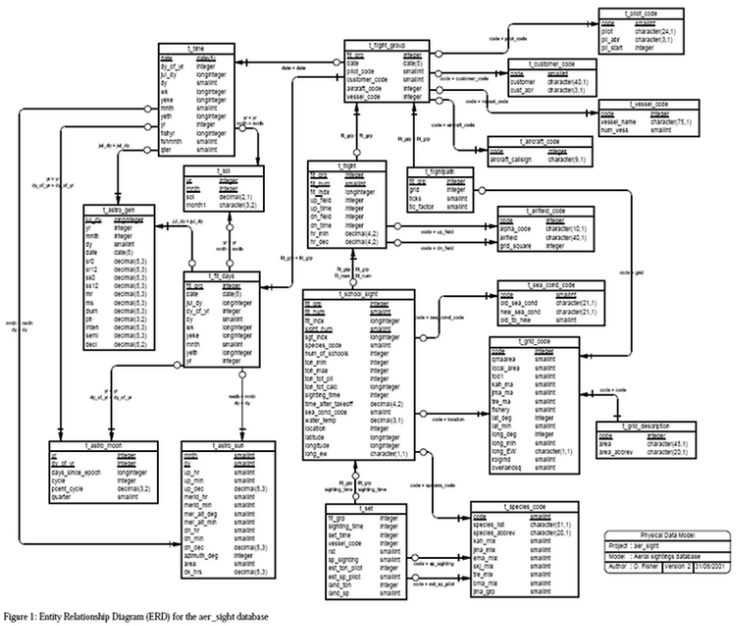


Figure 1: Entity Relationship Diagram (ERD) for the aer_sight database

How to handle inconsistency?

Not all databases are consistent: inconsistency arises in a variety of contexts such as data integration, data warehousing.

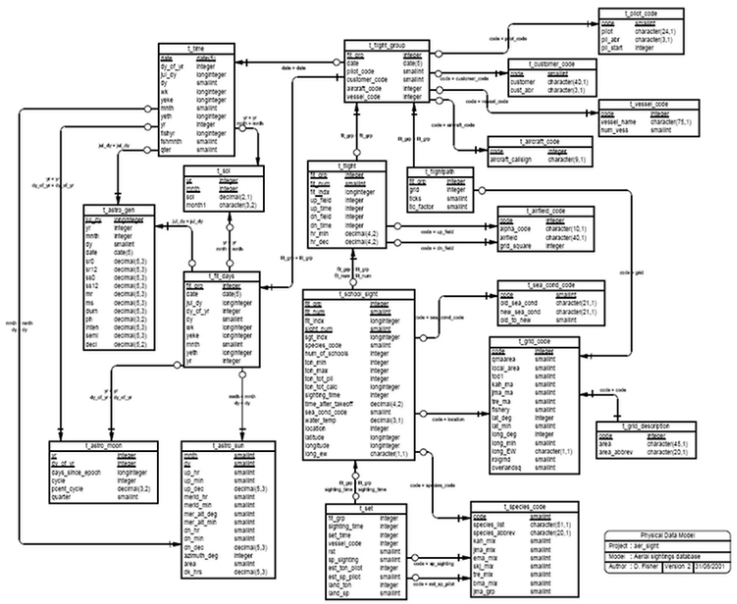


Figure 1: Entity Relationship Diagram (ERD) for the aer_sight database

Repairs

Arenas, Bertossi, Chomicki: J is a repair of I w.r.t. Σ if

- J is consistent,
- J differs from I in a "minimal way"

Repairs

Arenas, Bertossi, Chomicki: J is a repair of I w.r.t. Σ if

- J is consistent,
- J differs from I in a "minimal way"

How to define minimal?

- several possibilities: set-based, cardinality-based, attribute-based repairs

Repairs

Arenas, Bertossi, Chomicki: J is a repair of I w.r.t. Σ if

- J is consistent,
- J differs from I in a "minimal way"

How to define minimal?

- several possibilities: set-based, cardinality-based, attribute-based repairs

J is a (set-based) repair of I if

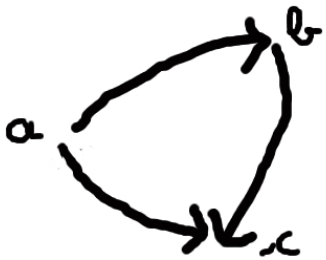
- J is consistent,
- if J' is s.t. $J' \oplus I \subsetneq J \oplus I$, then J' is inconsistent

Examples

J is a (set-based) repair of I if

- J is consistent,
- if J' is s.t. $J' \oplus I \subsetneq J \oplus I$, then J' is inconsistent

Example 1: $\Sigma = \{\forall x, y, x(R(x, y) \wedge R(x, z) \rightarrow y = z)\}$

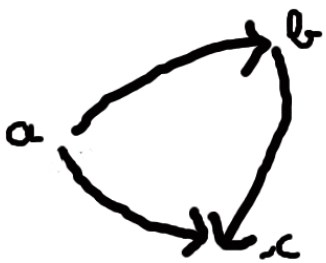


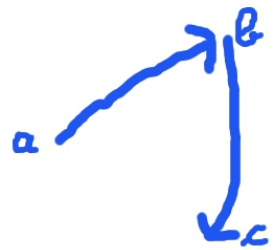
Examples

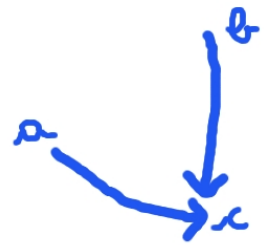
J is a (set-based) repair of I if

- J is consistent,
- if J' is s.t. $J' \oplus I \subsetneq J \oplus I$, then J' is inconsistent

Example 1: $\Sigma = \{\forall x, y, x(R(x, y) \wedge R(x, z) \rightarrow y = z)\}$





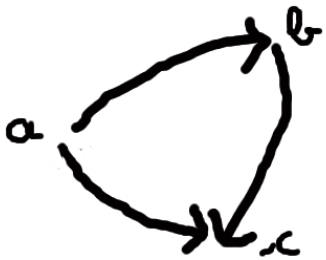


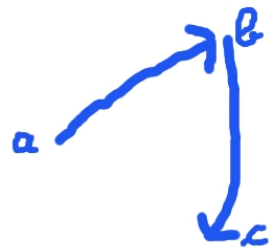
Examples

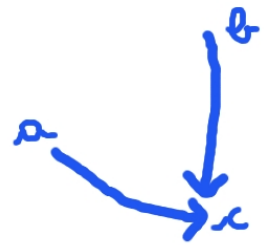
J is a (set-based) repair of I if

- J is consistent,
- if J' is s.t. $J' \oplus I \subsetneq J \oplus I$, then J' is inconsistent

Example 1: $\Sigma = \{\forall x, y, x(R(x, y) \wedge R(x, z) \rightarrow y = z)\}$







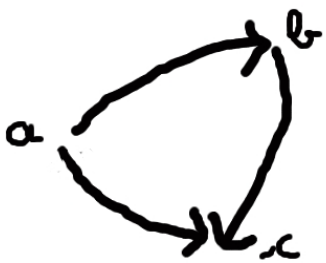
Example 2: $\Sigma = \{\forall x, y(R(x, y) \rightarrow \exists z R(y, z))\}$

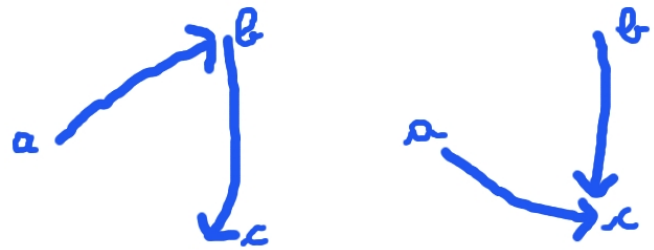
Examples

J is a (set-based) repair of I if

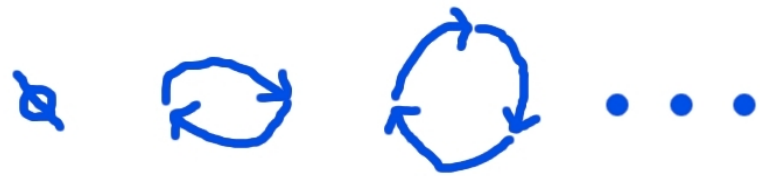
- J is consistent,
- if J' is s.t. $J' \oplus I \subsetneq J \oplus I$, then J' is inconsistent

Example 1: $\Sigma = \{\forall x, y, x(R(x, y) \wedge R(x, z) \rightarrow y = z)\}$





Example 2: $\Sigma = \{\forall x, y(R(x, y) \rightarrow \exists z R(y, z))\}$

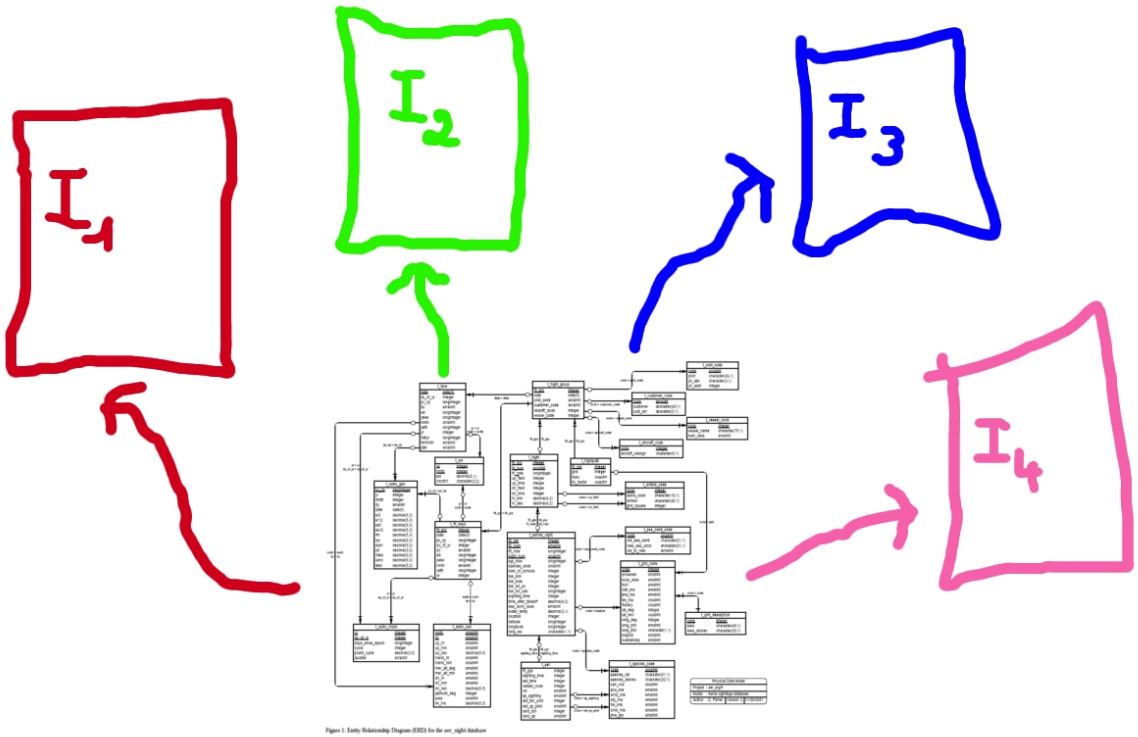


the repairs are all cycles

Consistent query answering

The consistent query answers of q on I, $Cons(q, I)$, is given by

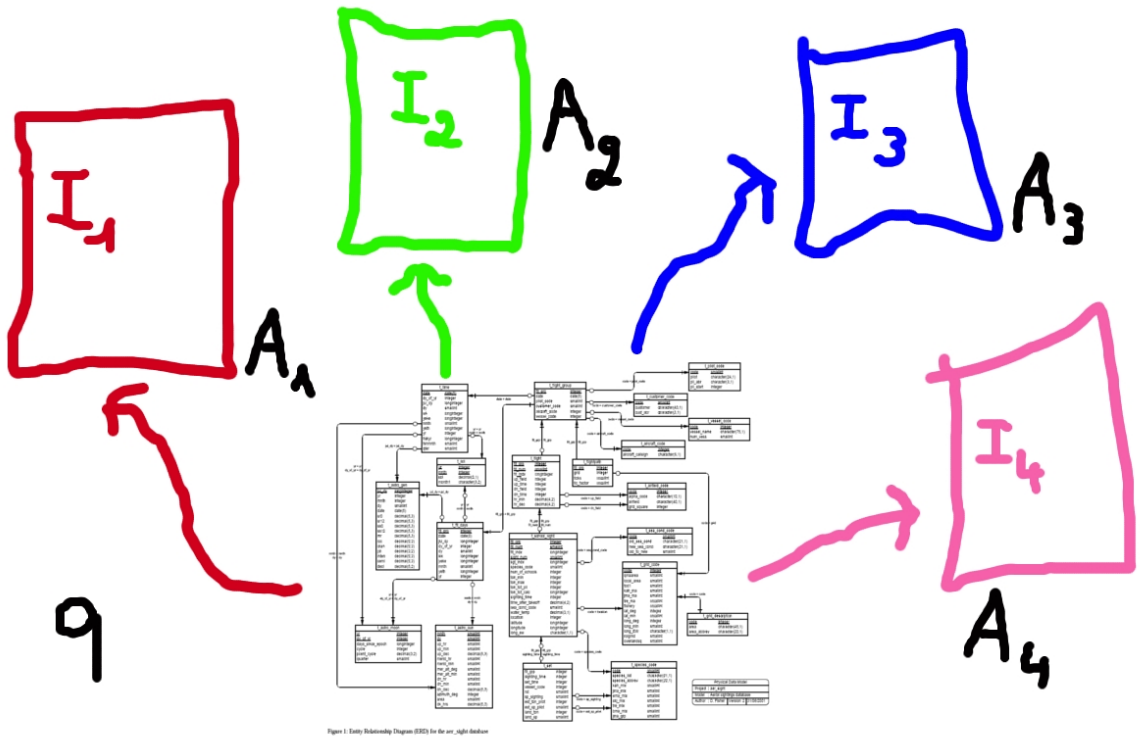
$$Cons(q, I) = \bigcap \{q(J) : J \text{ is a repair of } I\}$$



Consistent query answering

The consistent query answers of q on I, $Cons(q, I)$, is given by

$$Cons(q, I) = \bigcap \{q(J) : J \text{ is a repair of } I\}$$



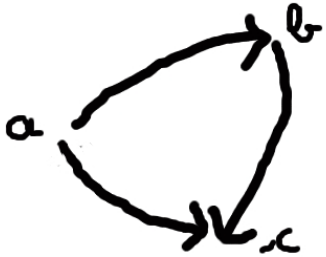
$$Cons(q, I) = A_1 \cap A_2 \cap A_3 \cap A_4$$

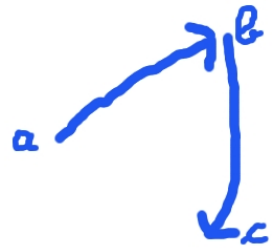
Example

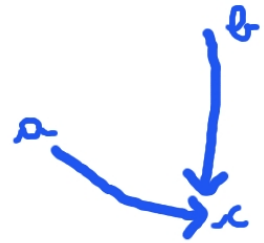
The consistent query answers of q on I, $Cons(q, I)$, is given by

$$Cons(q, I) = \bigcap \{q(J) : J \text{ is a repair of } I\}$$

Example: $\Sigma = \{\forall x, y, xR(x, y) \wedge R(x, z) \rightarrow y = z\}$







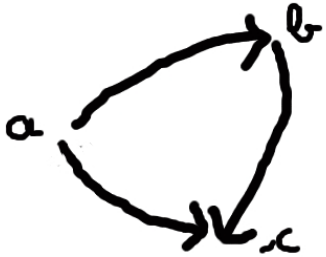
- $q(y) = \exists x R(x, y)$

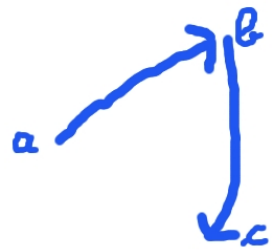
Example

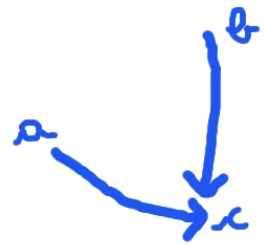
The consistent query answers of q on I, $Cons(q, I)$, is given by

$$Cons(q, I) = \bigcap \{q(J) : J \text{ is a repair of } I\}$$

Example: $\Sigma = \{\forall x, y, xR(x, y) \wedge R(x, z) \rightarrow y = z\}$







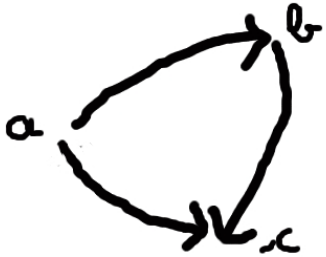
- $q(y) = \exists x R(x, y)$
- $q(I_1) = \{b, c\}$, $q(I_2) = \{c\}$

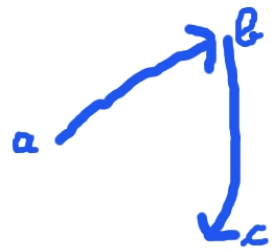
Example

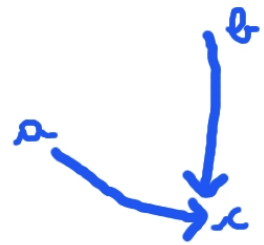
The consistent query answers of q on I, $Cons(q, I)$, is given by

$$Cons(q, I) = \bigcap \{q(J) : J \text{ is a repair of } I\}$$

Example: $\Sigma = \{\forall x, y, xR(x, y) \wedge R(x, z) \rightarrow y = z\}$







- $q(y) = \exists x R(x, y)$
- $q(I_1) = \{b, c\}$, $q(I_2) = \{c\}$
- $Cons(q, I) = \{c\}$

Complexity of consistent query answering

Data complexity of consistent query answering:

- fix a query q and a set of constraints Σ
- $CQA(q, \Sigma)$ is the problem that given a database I and a tuple $\mathbf{a}$, decide whether $\mathbf{a}$ belongs to the consistent answers of q on I

Complexity of consistent query answering

Data complexity of consistent query answering:

- fix a query q and a set of constraints Σ
- $CQA(q, \Sigma)$ is the problem that given a database I and a tuple $\mathbf{a}$, decide whether $\mathbf{a}$ belongs to the consistent answers of q on I

Data complexity of CQA has been extensively studied for different classes of constraints and queries

- constraints: key constraints, denial constraints, tgds (including LAV, GAV)
- complexity of CQA for conjunctive queries: from first-order rewritability to undecidability
- for an overview, see "Database repairing and consistent query answering" by Bertossi (2011)

Dichotomy conjecture

Fix a set Σ of key constraint and a conjunctive query q.

Fact: $CQA(q, \Sigma)$ is in co-NP.

Conjecture: (Afrati, Kolaitis)

- either $CQA(q, \Sigma)$ is polynomial,
- or $CQA(q, \Sigma)$ is co-NP complete

.

Dichotomy conjecture

Fix a set Σ of key constraint and a conjunctive query q.

Fact: $CQA(q, \Sigma)$ is in co-NP.

Conjecture: (Afrati, Kolaitis)

- either $CQA(q, \Sigma)$ is polynomial,
- or $CQA(q, \Sigma)$ is co-NP complete

. Partial answers:

- the conjecture holds if we only consider queries with 2 atoms containing 2 distinct relation symbols (Kolaitis, Pema)
- a dichotomy holds if we consider counting consistent query answering for acyclic self join free (Maslowski, Wijsen)

Aim

Study the dichotomy conjecture when Σ is a set of GAV constraints and q is a union of conjunctive queries

Aim

Study the dichotomy conjecture when Σ is a set of GAV constraints and q is a union of conjunctive queries

- a GAV constraint is a formula of the form

$$\forall \mathbf{x}(R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)) \rightarrow R(\mathbf{y}))$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}$ occur in $\mathbf{x}$

Aim

Study the dichotomy conjecture when Σ is a set of GAV constraints and q is a union of conjunctive queries

- a GAV constraint is a formula of the form

$$\forall \mathbf{x}(R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)) \rightarrow R(\mathbf{y}))$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}$ occur in $\mathbf{x}$

Aim

Study the dichotomy conjecture when Σ is a set of GAV constraints and q is a union of conjunctive queries

- a GAV constraint is a formula of the form

$$\forall \mathbf{x}(R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)) \rightarrow R(\mathbf{y}))$$

  where $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}$ occur in $\mathbf{x}$

- a union of conjunctive queries is a FO formula built from $\wedge$, $\vee$ and $\exists$

CAQ for GAV constraints and UCQ

Fix a set Σ of GAV constraints and a UCQ q.

Complexity result: $CQA(q, \Sigma)$ is in co-NP (Staworko; ten Cate, F., Kolaitis)

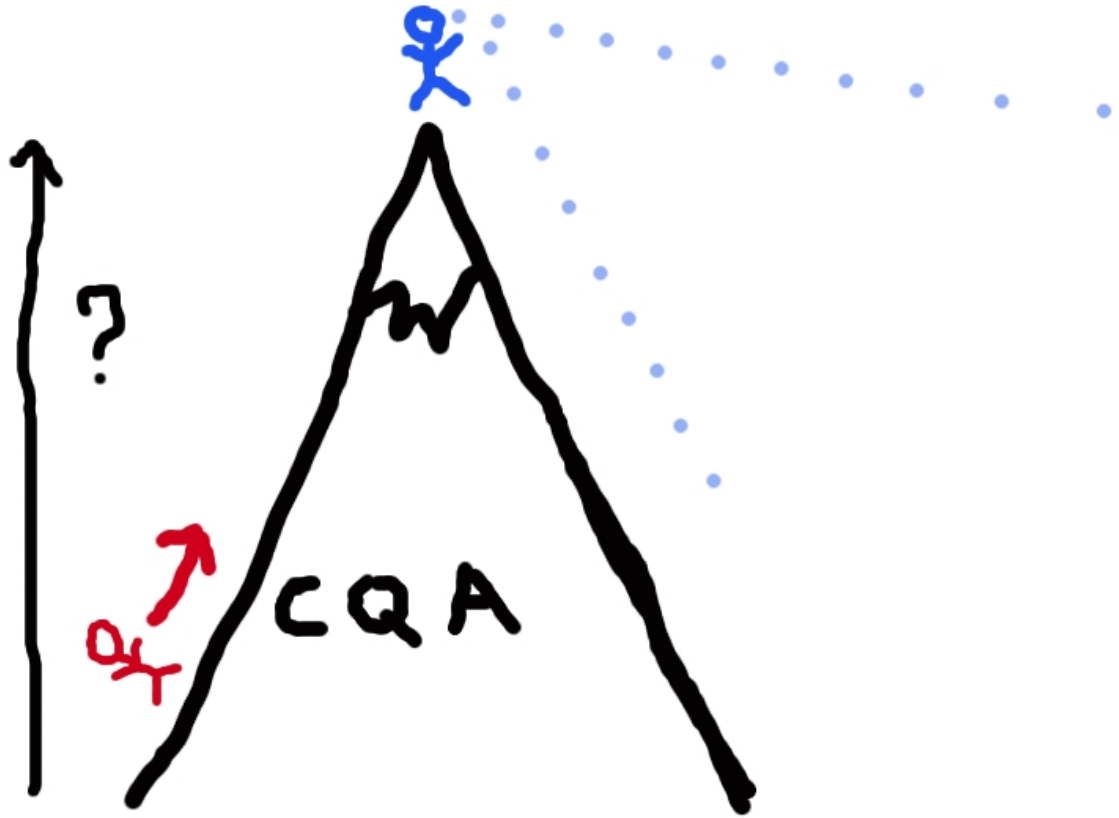
Conjecture:

- either $CQA(q, \Sigma)$ is polynomial,
- or $CQA(q, \Sigma)$ is co-NP complete

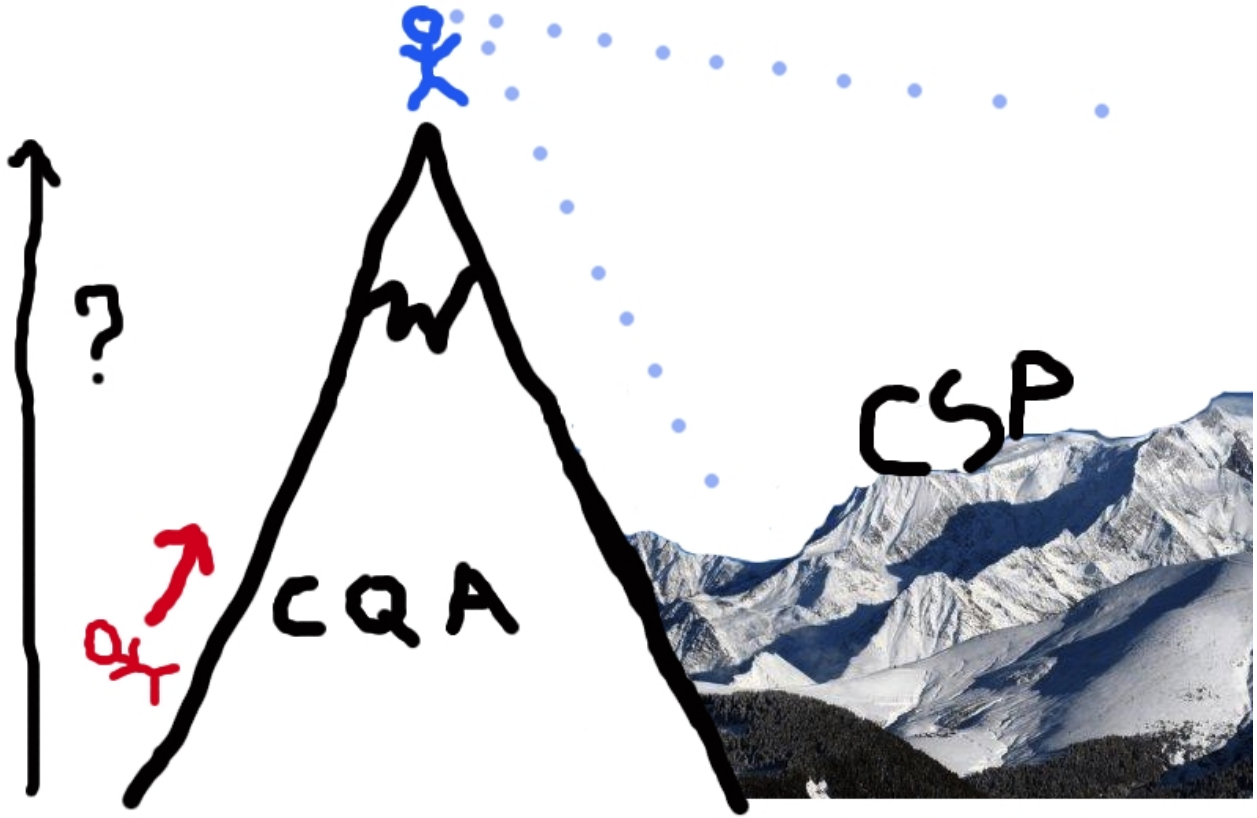
Strategy

Strategy

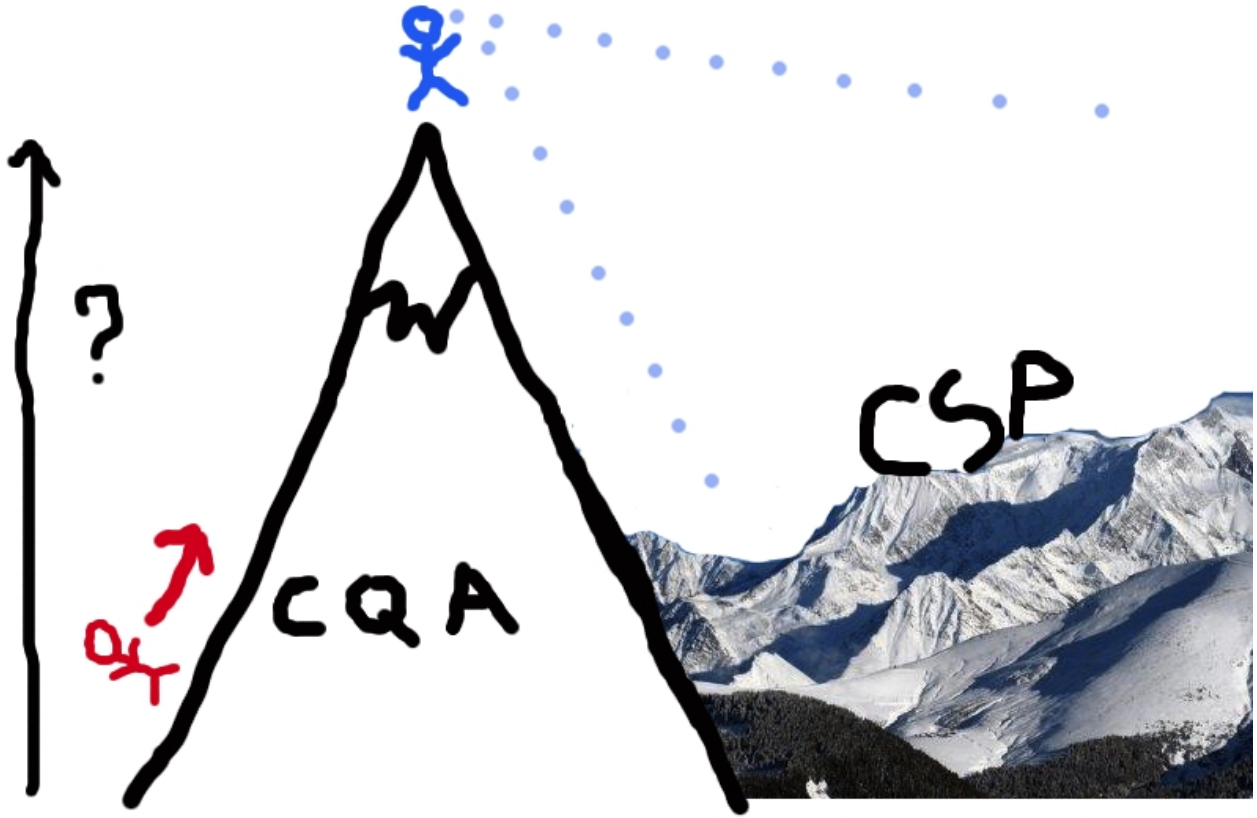
Strategy



Strategy

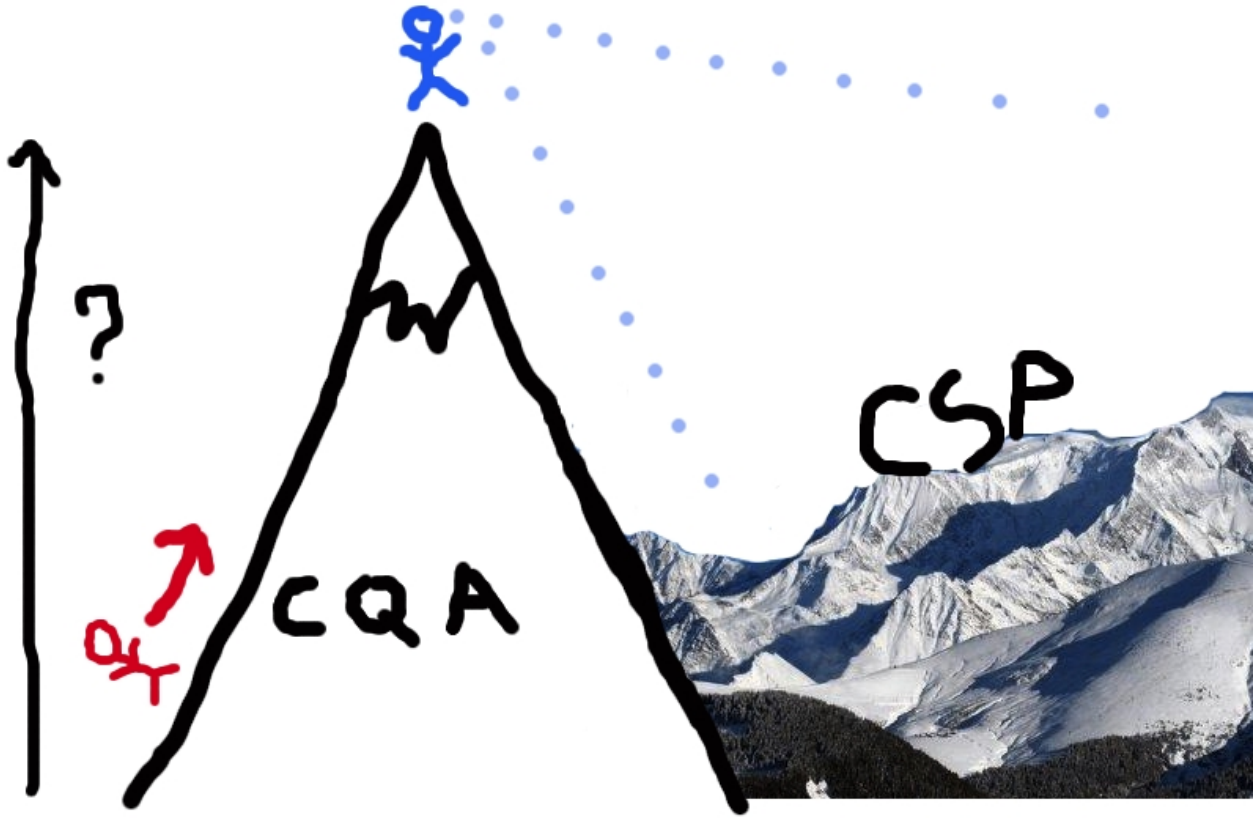


Strategy



Thm: a dichotomy for consistent query answering w.r.t GAV constraints and UCQs implies a dichotomy for CSP

Strategy



Thm: a dichotomy for consistent query answering w.r.t GAV constraints and UCQs implies a dichotomy for CSP

Consequence: our result gives insight why investigating dichotomy for consistent query answering has been so difficult

Structure of the proof

Thm: a dichotomy for consistent query answering w.r.t GAV constraints and UCQ implies a dichotomy for CSP

Fix a structure B.

- Step 1: show that proving a dichotomy for $CSP(B)$ is equivalent to prove a dichotomy for $CSP^*(B)$, a variant of CSP (Bulatov)
- Step 2: construct a UCQ q and a set of GAV constraints Σ such that $CSP^*(B)$ and the complement of $CQA(q, \Sigma)$ are polynomially inter-reducible.

Step 1

Fix a structure B. $CSP^*(B)$ is the following problem:

- input: a structure A and a partial map $f : A \to B$
- goal: decide if there is a (total) homomorphism $g : A \to B$ extending f

Step 1

Fix a structure B. $CSP^*(B)$ is the following problem:

- input: a structure A and a partial map $f : A \to B$
- goal: decide if there is a (total) homomorphism $g : A \to B$ extending f

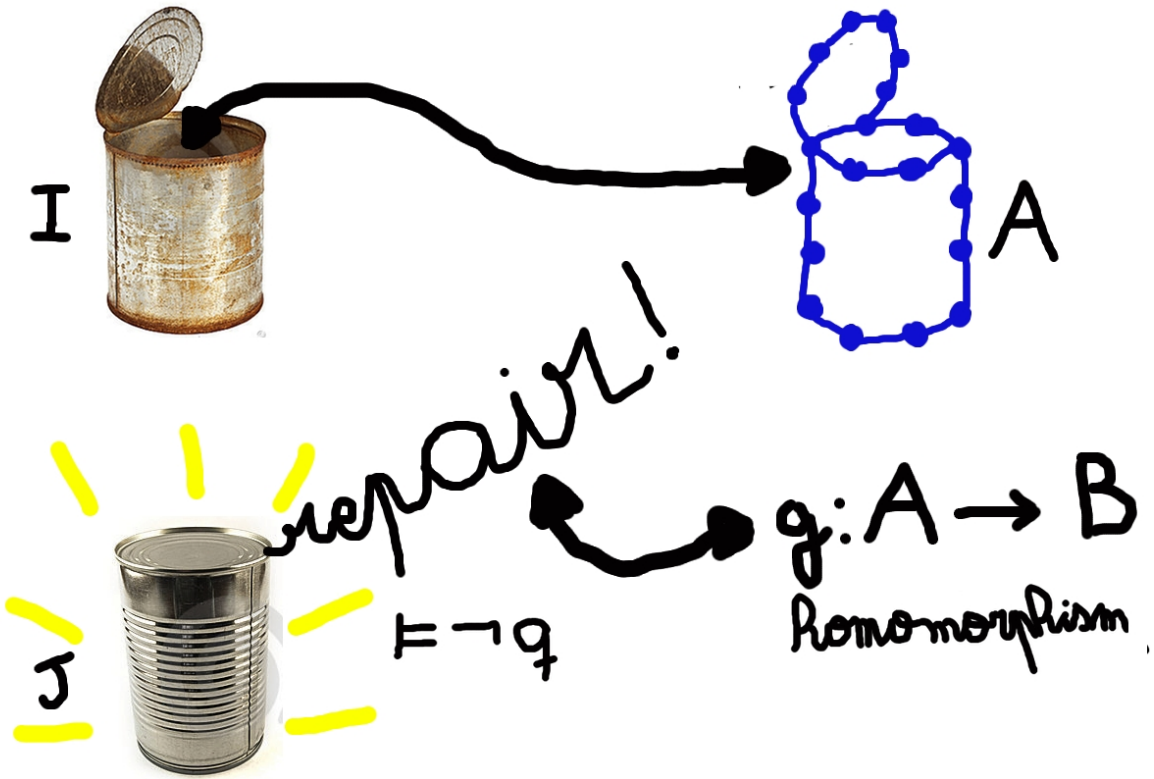
Bulatov: Proving a dichotomy for CSP is equivalent to proving a dichotomy for CSP^*

Step 2

Fix a graph B. Find q and Σ s.t. $CSP^*(B)$ and the complement of $CQA(q, \Sigma)$ are polynomially inter-reducible.

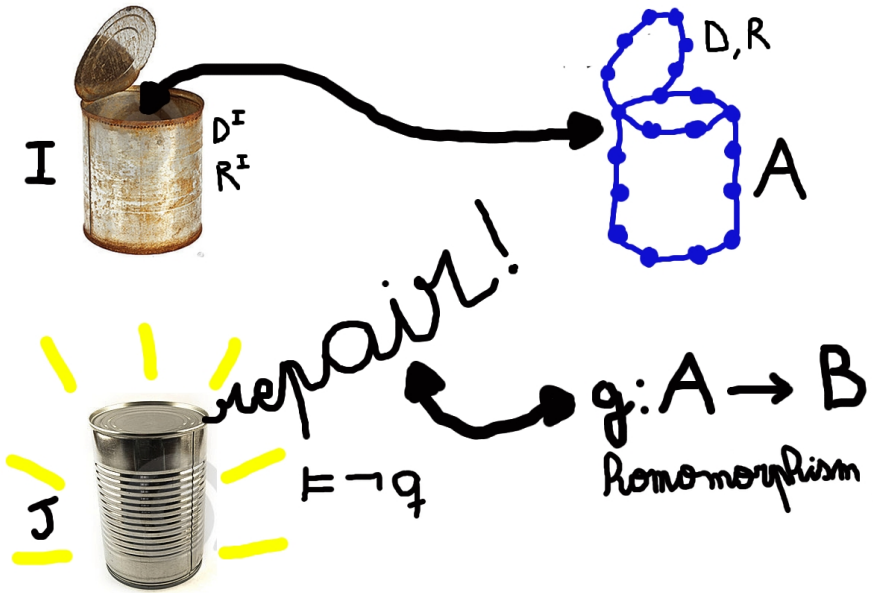
Step 2

Fix a graph B. Find q and Σ s.t. $CSP^*(B)$ and the complement of $CQA(q, \Sigma)$ are polynomially inter-reducible.



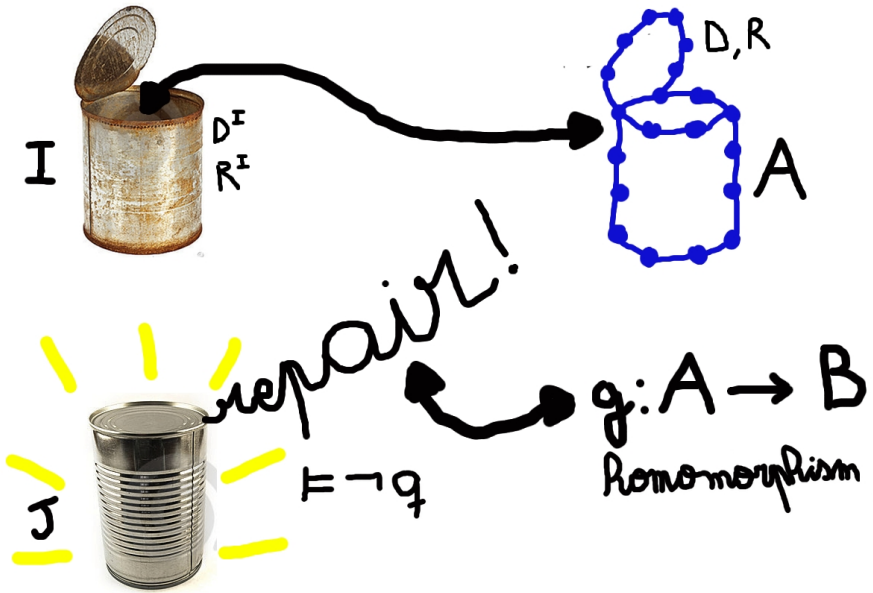
- How to encode g?
- How to ensure g is an homomorphism?
- How to ensure g is total?

Step 2



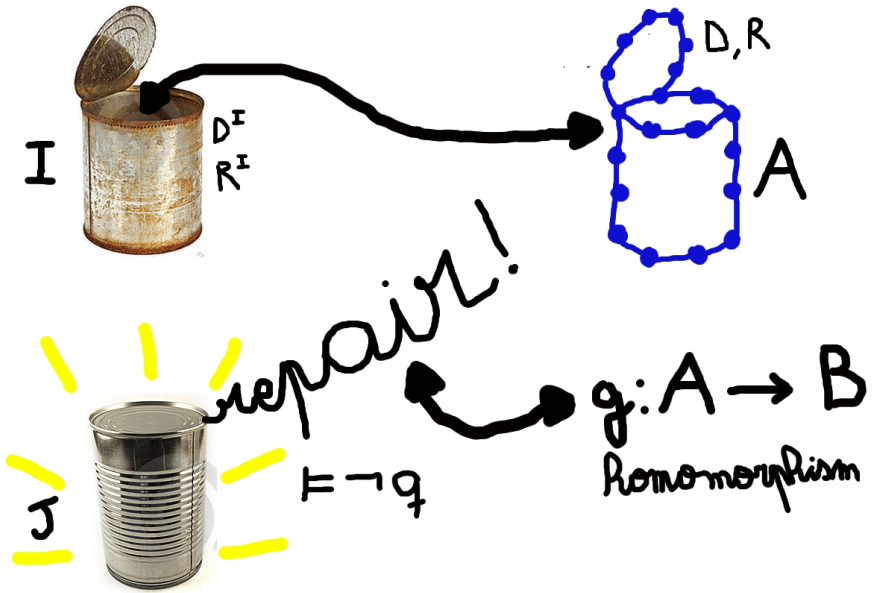
- How to encode g? introduce a formula $\phi_b(x)(b \in B)$
  - $g(x) = b$ iff $J \models \varphi_b(x)$

Step 2



- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure g is an homomorphism?

Step 2

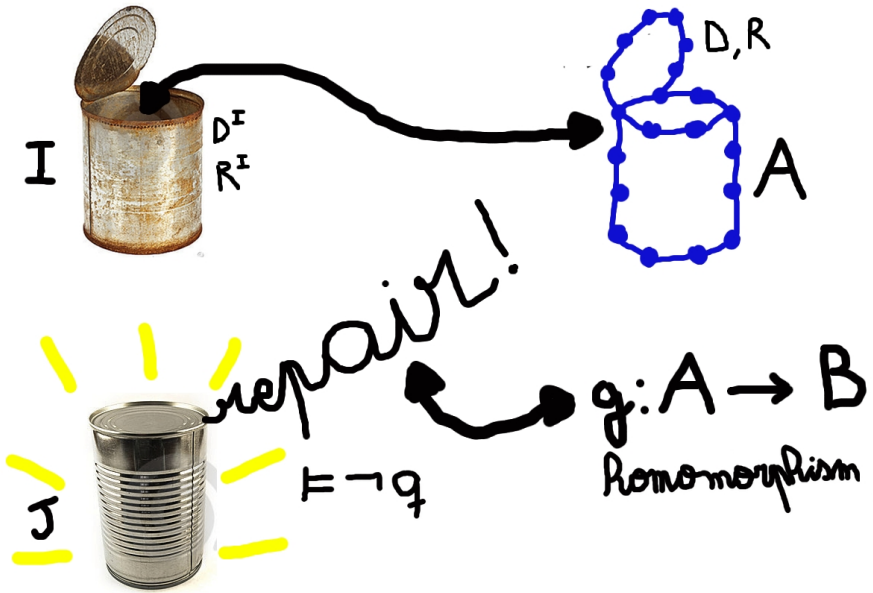


- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure g is an homomorphism?
  - if $(b, c) \notin R^B$, add the constraint

    $$\varphi_b(x) \wedge \varphi_c(y) \rightarrow C_R(x, y)$$

    where C_R encodes a subset of the complement of R

Step 2



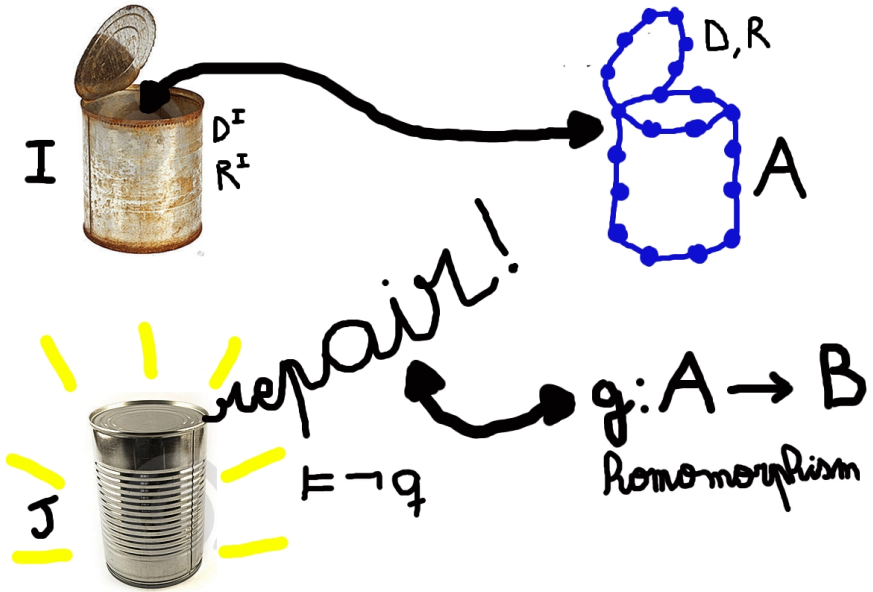
- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure g is an homomorphism?
  - if $(b, c) \notin R^B$, add the constraint

    $$\varphi_b(x) \wedge \varphi_c(y) \rightarrow C_R(x, y)$$

    where C_R encodes a subset of the complement of R
  - add the following formula as a conjunct of $\neg q$

    $$\neg \exists x, y(R(x, y) \wedge C_R(x, y))$$

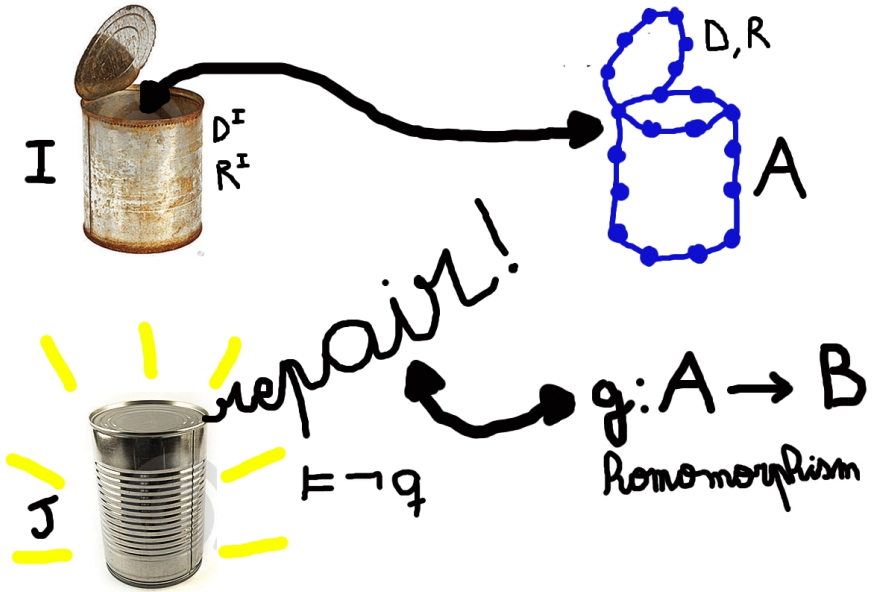
Step 2



- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure that g is total, i.e. $dom(g) = D^I$?

  $dom(g) = \{x : J \models \phi_b(x) \text{ for some } b \in B\}$

Step 2



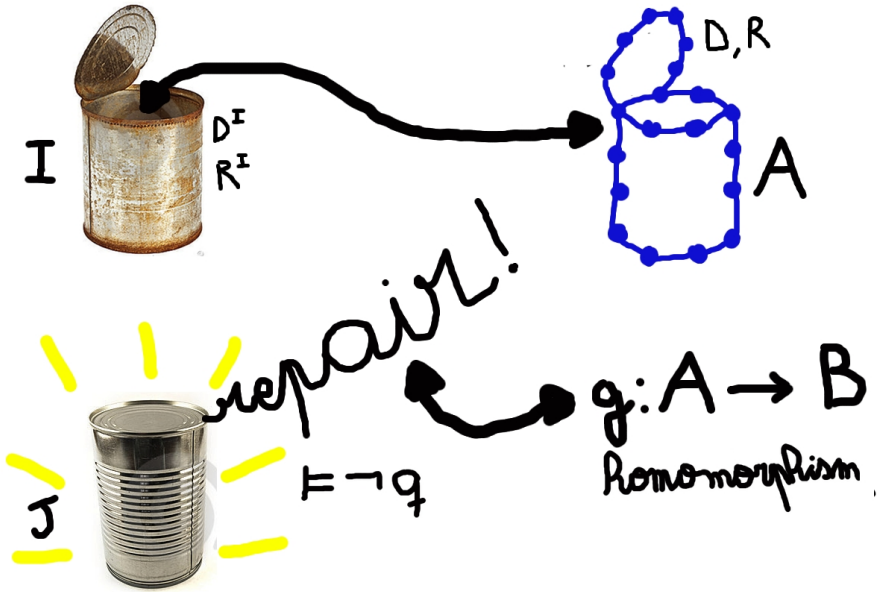
- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure that g is total, i.e. $dom(g) = D^I$?

  $dom(g) = \{x : J \models \phi_b(x) \text{ for some } b \in B\}$

  - ensure $D^J = D^I \setminus dom(g)$

  - ensure $D^J = \emptyset$

Step 2



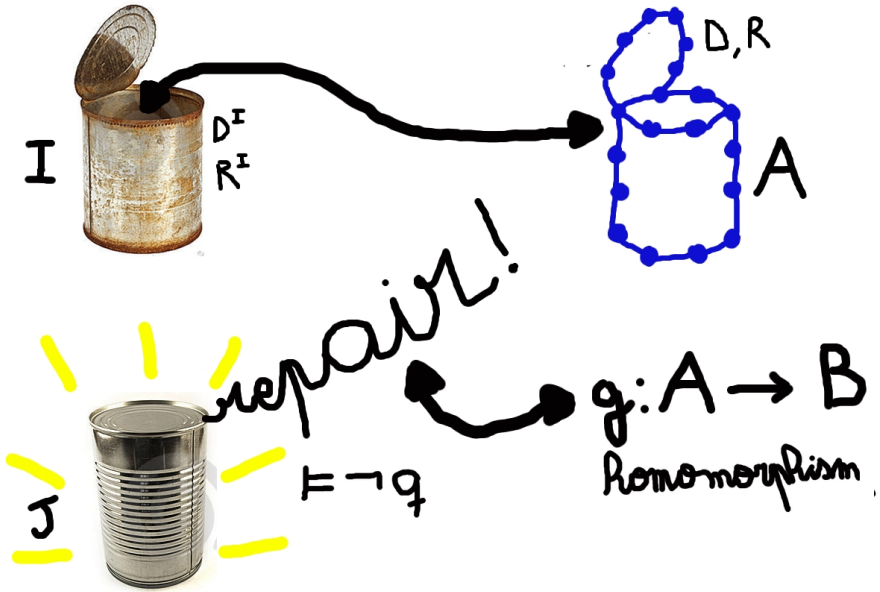
- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure that g is total, i.e. $dom(g) = D^I$?

  $dom(g) = \{x : J \models \phi_b(x) \text{ for some } b \in B\}$

  - ensure $D^J = D^I \setminus dom(g)$

  - ensure $D^J = \emptyset$
    - add $\neg \exists y D(y)$ as a conjunct of $\neg q$

Step 2

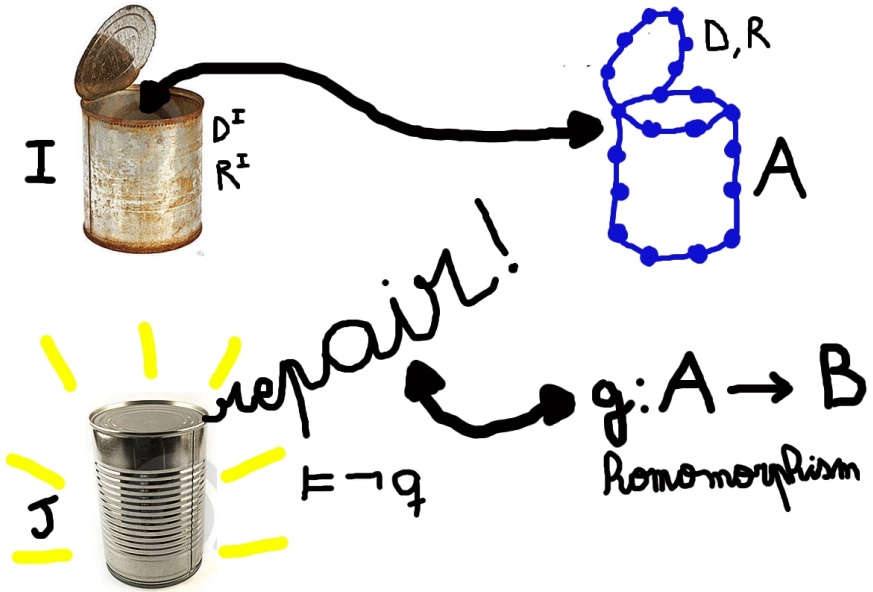


- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure that g is total, i.e. $dom(g) = D^I$?

  $dom(g) = \{x : J \models \phi_b(x) \text{ for some } b \in B\}$

  - ensure $D^J = D^I \backslash dom(g)$
    - add the constraints $D(x) \wedge \phi_b(x) \rightarrow E(x)$, where $E^J = \emptyset$
  - ensure $D^J = \emptyset$
    - add $\neg \exists y D(y)$ as a conjunct of $\neg q$

Step 2



- How to encode g? $g(x) = b$ iff $J \models \varphi_b(x)$
- How to ensure that g is total, i.e. $dom(g) = D^I$?

  $dom(g) = \{x : J \models \phi_b(x) \text{ for some } b \in B\}$

  - ensure $D^J = D^I \backslash dom(g)$
    - add the constraints $D(x) \wedge \phi_b(x) \rightarrow E(x)$, where $E^J = \emptyset$
    - how to ensure $E^J = \emptyset$? add $\neg\exists y E(y)$ as a conjunct of $\neg q$
  - ensure $D^J = \emptyset$
    - add $\neg\exists y D(y)$ as a conjunct of $\neg q$

Summary of step 2

Constraints:

- if there is no edge from b to c in B,

$$\varphi_b(x) \wedge \varphi_c(y) \rightarrow C_R(x, y)$$

- if $b \in B$, $\varphi_b(x) \wedge D(x) \rightarrow E(x)$

Summary of step 2

Constraints:

- if there is no edge from b to c in B,

$$\varphi_b(x) \wedge \varphi_c(y) \rightarrow C_R(x, y)$$

- if $b \in B$, $\varphi_b(x) \wedge D(x) \rightarrow E(x)$

Query $\neg q$:

$$\neg\exists x E(x) \wedge$$
$$\neg\exists x D(x) \wedge$$
$$\neg\exists x, y(R(x, y) \wedge C_R(x, y))$$

Other results

- a dichotomy for consistent query answering w.r.t. egds and UCQs implies a dichotomy for list-CSP
- a dichotomy for consistent query answering w.r.t. key constraints and non-boolean UCQs implies a dichotomy for list-CSP

Other results

- a dichotomy for consistent query answering w.r.t. egds and UCQs implies a dichotomy for list-CSP
- a dichotomy for consistent query answering w.r.t. key constraints and non-boolean UCQs implies a dichotomy for list-CSP

Fix a structure B. List-CSP is the following problem:

- input: a structure A and for all $a \in A$, a subset L_a of B
- goal: find an homomorphism $f : A \to B$ s.t. $f(a) \in L_a$ for all $a \in A$
- there is a dichotomy for list-CSP, but the proof is extremely complicated (Bulatov)

Further work

- find a similar results for different classes of queries and constraints
- show the other direction: a dichotomy for CSP implies a dichotomy for consistent query answering