Inconsistency-Tolerant Integrity Checking

Hendrik Decker
Instituto Tecnológico de Informática, Valencia
hendrik@iti.upv.es
Overview

• What is Inconsistency-tolerant Integrity Checking?

• Classifying Integrity Checking Methods

• Incons.-tolerant Knowledge Engineering
  * Knowledge Assimilation (Updating, Repair)
  * Consistent Query Answering
  * Semantic Query Optimization
  * Inconsistency Measuring
Many methods are inconsistency-tolerant. *Not all!*

**Classify methods:**

- Discard irrelevant constraints
- Focus on relevant cases
- Simplify relevant cases

Inconsistency tolerance is preserved only if *total integrity premise*, that $IC$ is satisfied in $D$, is *not* used!
Inconsistency-tolerant Integrity Checking

• **Basic Idea:**
  think of each constraint $I$ as a set of instances, “cases” of $I$, e.g., $I = \leftarrow p(x,x) :$

  $\text{Cases}(I) = \{ \leftarrow p(a,a), \leftarrow p(b,b), \leftarrow p(c,c), \ldots \}$

• **Instantiate only “global” variables:**

  Cases of $\forall X, Y \exists Z \text{ works-in}(X,Y) \rightarrow \exists Z \text{ emp}(X, Z)$
  are, e.g., $\forall X \exists Z \text{ works-in}(X, \text{ toy}) \rightarrow \exists Z \text{ emp}(X, Z)$
  $\exists Z \text{ works-in}(\text{ jim, toy}) \rightarrow \exists Z \text{ emp}(\text{ jim, Z})$
  but not $\text{ works-in}(\text{ jim, toy}) \rightarrow \text{ emp}(\text{ jim, 40000})$
Inconsistency-tolerant Integrity Checking

• Only consider cases that are satisfied before update:
  If, for each $C \in \text{Cases}(IC)$ that is satisfied in state $D$ before update $U$, $C$ is also satisfied in state $D^U(C)$ after $U$, then $U$ is ok, otherwise ko.

• Ignore cases that are violated before update:
  e.g., for $D = \{ p(a,b), p(b,c), p(c,c) \}$, $I \leftarrow p(x,x)$, $U = \text{insert } p(a,c)$, all cases satisfied in $D$ remain so in $D^U$. Thus, $U$ is ok, though $D^U(I) = \text{violated}$.

  For $U' = \text{insert } p(a,a)$, $U'$ leads to ko.
Counter-Example

\[ D = \{ q(a), r(b,b) \} \quad IC = \{ \leftarrow q(a), r(x,x), \leftarrow q(x), r(b,x) \} \]

violated case: \( \leftarrow q(a), r(b,b) \)  satisfied case: \( \leftarrow q(b), r(b,b) \)

\( U = \text{insert } q(b) \)  violates \( \leftarrow q(b), r(b,b) \)

Clever but not inconsistency-tolerant method \( M \):
\[ I = \leftarrow q(a), r(x,x) \]  not relevant \( \text{wrt } U \). If \( IC \) satisfied in \( D \),
then \( I \) satisfied in \( D \) and \( D^U \), thus \( \leftarrow r(x,x) \) satisfied in \( D^U \).

Simplified relevant case \( \leftarrow r(b,b) \) subsumed by \( \leftarrow r(x,x) \),
hence \( \leftarrow r(b,b) \) satisfied in \( D^U \), thus \( M \) outputs \( \text{ok} \).
That’s wrong because \( \leftarrow q(b), r(b,b) \) is violated by \( U \).
Inconsistency-tolerant Updates

• *Basic Idea:*
  If all cases satisfied before update remain satisfied after update, then update is *ok*.

Inconsistency-tolerant Repairs

• *Basic Idea:*
  Repair only some, not all violated cases. If all cases satisfied before repair remain satisfied after repair, then repair is *ok*. 
Inconsistency-tolerant Repairs for CQA

• **Basic Idea:**
  Repair only cases that are relevant wrt query

• This is what happens if, as in CQA, SQO is applied to queries in $D$ if $IC$ is violated in $D$.

• **Wanted:** definition of relevance wrt query.

• **Conjecture:**
  If answer is true in all ok-repairs of relevant cases, then answer is consistent.
Inconsistency Measures

• *Use inconsistency measure for integrity checking*: Accept updates *if* they do not increase measured amount of inconsistency.

• *All inconsistency-tolerant methods accept updates only if* they do not increase amount of inconsistent cases.
Conclusion

- Understanding of inconsistency-tolerant integrity checking is getting better
- Proofs of inconsistency tolerance become easier
- Doors are open to combine inconsistency-tolerant integrity checking with Knowledge Assimilation, SQO, CQA, Inconsistency Measuring, more