active integrity constraints for multi-context systems

luís cruz-filipe[1]

(with graça gaspar[2], isabel nunes[2], peter schneider-kamp[1])

[1]department of mathematics and computer science
university of southern denmark

[2]department of informatics
faculty of sciences, university of lisbon

ekaw 2016, bologna, italy
november 23rd, 2016

structure

1. motivation
2. the context
3. our formalism
4. evaluation

integrity constraints in reasoning systems

relational dbs	the classical setting
deductive dbs	mostly mid-1980s $\rightsquigarrow$ separate integrity constraints from data $\rightsquigarrow$ integrity constraints as preferred models
ontologies	last 15–20 years $\rightsquigarrow$ open-world semantics makes the problem different $\rightsquigarrow$ integrity constraints as terminological axioms (but with a different semantics)
heterogeneous systems	last 10 years $\rightsquigarrow$ in multi-context systems (our setting) $\rightsquigarrow$ internalize integrity constraints
in general	no continuation, no apparent consensus

our goal

- generalize existing notions in particular frameworks (e.g. relational databases)
- expressive enough to capture conditions spanning several systems
- decidability? good complexity bounds?
- algorithms for repairing inconsistencies

our target active integrity constraints (flesca *et al.*, '04)

- defined for relational databases
- allow to express both *constraints* and *repair actions*
- "good" algorithms for repair
- . . . and it's kind of a nice formalism

our contribution

- active integrity constraints in a general-purpose framework
- captures previous constructions as special cases
- clean separation between consistency and integrity
- including repair actions avoids need for abduction
- repairs can be computed automatically
  (with a grain of salt)

active integrity constraints (for databases)

main idea

datalog-style rules

- body specifies an integrity constraint (clausal, denial form)
- heads are sets of "repair actions" (alternative)
- several different semantics

algorithms

tree-based algorithms

- can compute different kinds of repairs
- (non-deterministic) polynomial complexity
- sometimes require extra testing (complexity...)

multi-context systems

main idea

- reasoning systems (“contexts”)
- connected by datalog-style rules (“bridge rules”)

brewka & eiter '07

heterogenous non-monotonic multi-context systems

heterogeneous contexts can use different logics

non-monotonic bridge rules can contain negation

multi-context several different systems

equilibrium

an equilibrium is a set of beliefs that is compatible with all knowledge bases and bridge rules

$\rightsquigarrow$ think logic programming...

ontologies

in particular, we can view an ontology as an mcs

- the a-box is one context
- the t-box is another context
- there are bridge rules injecting all instances from the a-box to the t-box
- equilibria are sets all queries that return true

$\rightsquigarrow$ separation between a-box and t-box is useful for some types of integrity constraints

active integrity constraints

syntax an active integrity constraint is written as a bridge rule with disjunctive head

validity actions on the head must satisfy some constraints

- for every action, there is an inconsistent state it repairs
- for every inconsistent state, there is an action that repairs it

$\rightsquigarrow$ validity is undecidable in general, arguably simple in practice

evaluation

- capture all classes of ontology integrity constraints from a 2013 survey
- are able to define actions for their heads and show validity
- examples in paper, but approach is systematic
- also discuss some other types of ontology integrity constraints

some examples

specific type constraints

(A:gradStudent(X)),(A:student(X))
$\Longrightarrow$ (A:del(student(X)))

property domain constraints

(T: enrolled(X,Y), not (T: student(Y))
$\Longrightarrow$ (A: add(student(Y)))

functional dependencies

(A:hasEmail(X,Z)),(A:hasEmail(Y,Z)),not (T:(X=Y))
$\Longrightarrow$ (A:del(hasEmail(X,Z))) | (A:assert(X=Y))

minimum cardinality constraints

(T:($\leq$ 10.enrolled)(X))
$\Longrightarrow$ (A:redistribute($\neg$class(X)))

thank you!