Using SAT-Solvers to Compute Inference-Proof Database Instances

Cornelia Tadros and Lena Wiese

Fakultät für Informatik
LS 6 (ISSI)

September 24, 2009

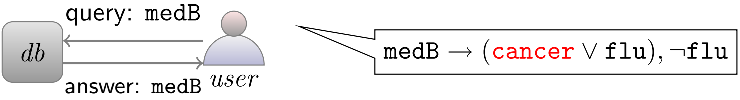
technische universität dortmund

Outline

1. Introduction
   - Inference Control
   - Controlled Query Evaluation
   - Preprocessing for CQE
2. Using SAT-Solvers for $pre\mathrm{CQE}$
3. $pre\mathrm{CQE}$ Prototype and Tests
4. Conclusion

Inference control

- Protect confidential and private information in database instance db
- Personalized security (confidentiality) policy pot_sec
- Personalized availability requirements $avail$
- User profile (a priori knowledge) $prior$
- IC system automatically distorts some answers
  - Avoids harmful user inferences:

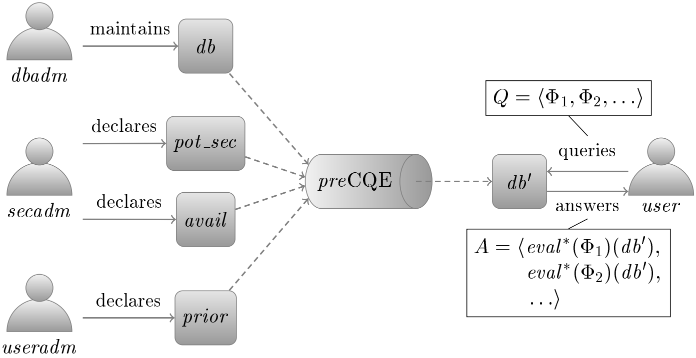


- Here: modify input database, i.e., remove or add entries (similiar to Cover Stories; eg. Cuppens et al, 2001)
- Automatically generate “inference-proof” output instance

Prior work: Controlled Query Evaluation (CQE)

- (Biskup/Bonatti, 2007; Biskup/Wiese, 2008)...
- In (Biskup/Bonatti, 2007)... logical view of relational data model
- In (Biskup/Wiese, 2008) and in the following: propositional data model
- Infinite propositional alphabet $\mathcal{P}$
- Complete database instance as finite set of entries (propositional variables in $\mathcal{P}$)
- All propositional formulas over $\mathcal{P}$ as query language
- (Biskup/Wiese, 2008) presents a branch & bound algorithm to precompute an inference-proof propositional instance
- In this work precomputation employing SAT-solver technology

preCQE: Preprocessing for CQE



Example

$$\mathcal{P} = \{\texttt{cancer}, \texttt{aids}, \texttt{flu}, \texttt{cough}, \ldots, \texttt{medA}, \texttt{medB}, \texttt{medC}, \ldots\}$$
$$db = \{\texttt{cancer}, \texttt{aids}, \texttt{medA}, \texttt{medB}\}$$

with the query evaluation function

$$eval^*(\Phi)(db) = \text{ if } I^{db} \models \Phi \text{ then } \Phi \text{ else } \neg\Phi$$

In our example: $eval^*(\texttt{flu})(db) = \neg\texttt{flu}$ and $eval^*(\texttt{cancer})(db) = \texttt{cancer}$

$$pot\_sec = \{\texttt{cancer}, \texttt{aids}\}$$
$$avail = \{\texttt{medA} \wedge \texttt{medB}, \texttt{medB}\}$$
$$prior = \{\neg\texttt{medA} \vee \texttt{cancer} \vee \texttt{aids}, \neg\texttt{medB} \vee \texttt{cancer} \vee \texttt{flu}\}$$

Preprocessing for CQE

Definition: Inference-proofness of db'

1. [Consistency] $I^{db'} \models prior$
2. [Confidentiality] $I^{db'} \not\models \Psi$ for every $\Psi \in pot\_sec$

- $db = \{\texttt{cancer}, \texttt{aids}, \texttt{medA}, \texttt{medB}\}$ is not inference-proof for $pot\_sec = \{\texttt{cancer}, \texttt{aids}\}$ and $prior = \{\neg\texttt{medA} \vee \texttt{cancer} \vee \texttt{aids}, \neg\texttt{medB} \vee \texttt{cancer} \vee \texttt{flu}\}$
- $db'_1 = \{\texttt{medB}, \texttt{flu}\}$ and $db'_2 = \emptyset$ are inference-proof
- Find db' that satisfies constraint set $C := prior \cup Neg(pot\_sec)$ with $Neg(pot\_sec) := \{\neg\Psi \mid \Psi \in pot\_sec\}$.
  In our example: $C :=$
  $\{\neg\texttt{medA} \vee \texttt{cancer} \vee \texttt{aids}, \neg\texttt{medB} \vee \texttt{cancer} \vee \texttt{flu}, \neg\texttt{cancer}, \neg\texttt{aids}\}$

Definition: Availability / distortion distance

- $avail\_dist(db') := ||\{\Theta \in avail \mid eval^*(\Theta)(db') \neq eval^*(\Theta)(db)\}||$
  $avail_dist$counts amount of modifications regarding the availability policy
- $db\_dist(db') := ||\{A \in \mathcal{P} \mid eval^*(A)(db') \neq eval^*(A)(db)\}||$
  db_distcounts amount of modifications between $db'$and db

- Objectives (in descending order of priority):
  1. construct inference-proof instance db'
  2. minimize availability $avail\_dist(db') \rightarrow min$
  3. minimize distortion $db\_dist(db') \rightarrow min$
- No user history (log file) has to be stored
- db' can be accessed by a user modeled by $prior$ without control but with the protection of pot_sec

Outline

1. Introduction
2. Using SAT-Solvers for preCQE
3. preCQE Prototype and Tests
4. Conclusion

SAT-Solver technology

- Motivation:
  - Goal: Find (propositional) interpretation satisfying $C := prior \cup Neg(pot\_sec)$ with $Neg(pot\_sec) := \{\neg\Psi \mid \Psi \in pot\_sec\}$ under optimization
  - benefit from elaborated SAT-Solver technology
- yearly “SAT competition” including the “MAXSAT evaluation” for optimization problems connected with SAT, e.g. W-PMSAT

Definition W-PMSAT

- Given (propositional) clauses associated with weights (non-negative integers)
- Find an interpretation with maximal sum of weights of satisfied clauses, so that all clauses with highest weight (hard constraints) are satisfied

W-PMSAT instances for $\mathit{pre}\mathrm{CQE}$

- Three levels of constraints (clauses):
  1. C_1 with weight $w_1 := 1$
  2. C_2 with weight $w_2 := card(C_1) * w_1 + 1$
  3. C_3 with weight $w_3 := card(C_2) * w_2 + card(C_1) * w_1 + 1$
- for three objectives (in ascending order of priority):
  1. Distortion Minimization:
     $C_1 := \bigcup_{A \in \mathcal{P}_{decision}} eval^*(A)(db)$ with weight w_1

$$\begin{aligned} \mathcal{P}_{decision} &= \{\texttt{cancer}, \texttt{aids}, \texttt{flu}, \texttt{medA}, \texttt{medB}\} \\ db &= \{\texttt{cancer}, \texttt{aids}, \texttt{medA}, \texttt{medB}\} \\ C_1 &= \{\texttt{cancer}, \texttt{aids}, \neg\texttt{flu}, \texttt{medA}, \texttt{medB}\} \text{ with weight } w_1 = 1 \end{aligned}$$

2. Availability Preservation:
$C_2 := \{S_\Theta \mid \Theta \in avail\}$ with weight w_2

$$\begin{aligned} avail &= \{\texttt{medA} \wedge \texttt{medB}, \texttt{medB}\} \\ C_2 &= \{S_{\texttt{medA}\wedge\texttt{medB}}, S_{\texttt{medB}}\} \text{ with weight } w_2 = card(C_1) + 1 = 6 \end{aligned}$$

3. Hard Constraints (incl. Inference-Proofness):
$C_3 := C \cup \{c \vee \neg S_\Theta \mid c$ clause of $cnf(eval^*(\Theta)(db)), \Theta \in avail\}$ with weight w_3 and $C := prior \cup Neg(pot\_sec)$

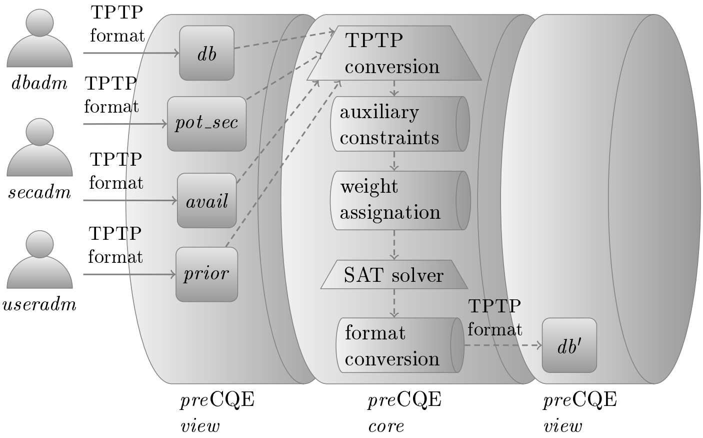
$$C_3 := C \cup \{\texttt{medA} \vee \neg S_{\texttt{medA}\wedge\texttt{medB}}, \texttt{medB} \vee \neg S_{\texttt{medA}\wedge\texttt{medB}}, \texttt{medB} \vee \neg S_{\texttt{medB}}\}$$

with weight
$w_3 = card(C_2) * w_2 + card(C_1) * w_1 + 1 = 2 * 6 + 5 * 1 + 1 = 18$

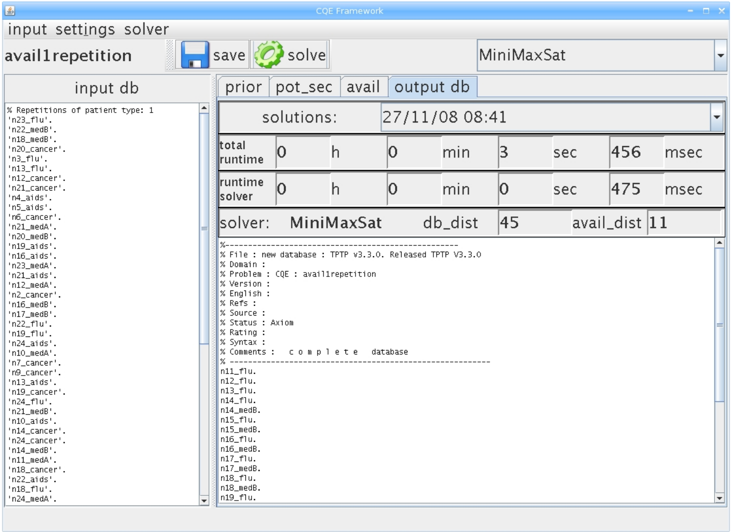
Outline

1. Introduction
2. Using SAT-Solvers for *pre*CQE
3. *pre*CQE Prototype and Tests
   - Implementation
   - Test Cases
4. Conclusion

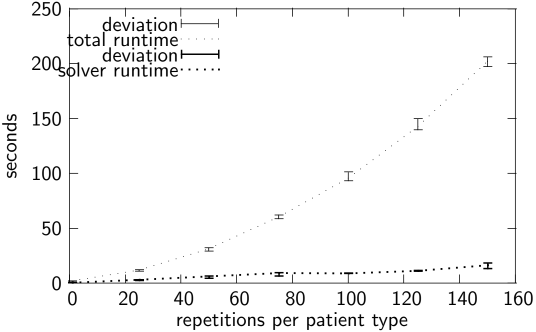
Implementation of Prototype



User interface



Test Case: Different patient types

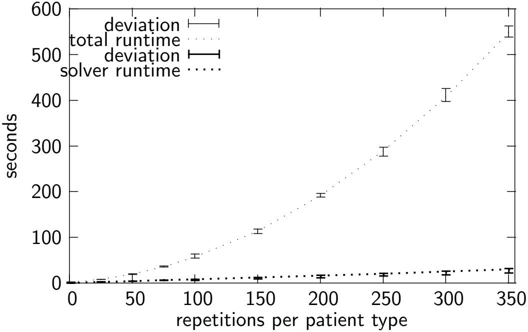


rep.	dec. vars.	clauses	
		soft	hard
1	120	120	96
25	3000	3000	2400
50	6000	6000	4800
75	9000	9000	7200
100	12000	12000	9600
125	15000	15000	12000
150	18000	18000	16800

$pot\_sec = \{\texttt{cancer}, \texttt{aids}\}$ and

$prior = \{\neg\texttt{medA} \vee \texttt{cancer} \vee \texttt{aids}, \neg\texttt{medB} \vee \texttt{cancer} \vee \texttt{flu}\}$

Test Case: Availability Policy



rep.	dec. vars.	clauses low	clauses aux.	clauses hard
1	65	65	26	78
25	1625	1625	650	1950
50	3250	3250	1300	3900
75	4875	4875	1950	5850
100	6500	6500	2600	7800
150	9750	9750	3900	11700
200	13000	13000	5200	15600
250	16250	16250	6500	19500
300	19500	19500	7800	23400
350	22750	22750	9100	27300

$$avail = \{\texttt{n1\_medA}, \texttt{n1\_medB}, \texttt{n2\_medA}, \texttt{n2\_medB}, \dots\}$$

Outline

1 Introduction

2 Using SAT-Solvers for *pre*CQE

3 *pre*CQE Prototype and Tests

4 Conclusion

Achievements and open subjects

- precomputation of an inference-proof, availability-preserving and distortion-minimal instance by data modification ("lying")
- by translation to W-PMSAT and use of SAT-Solver technology
- promising test results for a large number of database entries

- extension to relational data model
- update of policies or database instance