Application: context-aware systems

Our application: situation recognition for context-aware systems

- observe complex hard- & software system
  Goal: save energy
- recognize situations, where adaptation is beneficial
- Description Logic-based approach:
  TBox for background knowledge, ABox for observations
- employ Description Logic reasoning: conjunctive query answering
- collecting data from several sources! — sensors

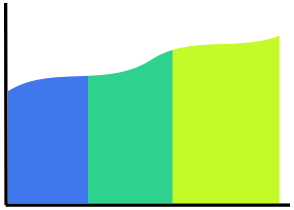
Application setting

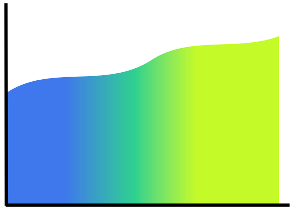
- data (& ontology) stored in a data base
- sliding windows approach to monitored data
  - observations are recorded at fixed intervals
  - sequence of system snapshots is obtained
- raw data is preprocessed by data base
  - raw data is ‘cleaned’ and aggregated
- situation descriptions are stored as DB queries

Why fuzzy information?

Represent sensor values 'faithfully'

- Logic: symbolic representation of values
- values: low, medium, high
- small change in value $\longrightarrow$ similar classification
- can level out noise





Description logics and fuzzy information

Semantics:

- Classical logic: $\{0, 1\}$ binary
- Fuzzy logic: $\{0, \cdots, 1\}$ infinitely many membership degrees

Results on DL & fuzzy information:

- easily undecidable!
- decidable, if finitely many membership degrees are used.
  But: costly reduction!

Our goal: employ reasoner for DL-Lite$_R$ (a.k.a. OWL 2 QL)

- DL-Lite$_R$-reasoners: optimized implementations!
- only the ABox and the queries are fuzzy!
  (TBox stays crisp)

The description logic DL-Lite$_R$ and its fuzzy variant

DL-Lite$_R$ concepts: $B \rightarrow A \mid \exists Q \qquad C \rightarrow \top \mid B \mid \neg B$

$Q \rightarrow P \mid P^- \qquad R \rightarrow Q \mid \neg Q$

DL-Lite$_R$ axioms: $B \sqsubseteq C \qquad Q \sqsubseteq R \qquad \mathsf{funct}(Q)$

Fuzzy DL-Lite$_R$ assertions: $\langle B(a), d \rangle \qquad \langle P(a,b), d \rangle$

Interpretation for fuzzy DL-Lite$_R$: $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$

- individuals: $a^\mathcal{I} \in \Delta^\mathcal{I}$
- named concepts: $A^\mathcal{I} : \Delta^\mathcal{I} \rightarrow [0,1]$
- atomic roles: $P^\mathcal{I} : \Delta^\mathcal{I} \times \Delta^\mathcal{I} \rightarrow [0,1]$

Fuzzy operators and semantics of complex concepts

	t-norm $a \otimes b$	negation $\ominus a$
Gödel	$\min(a, b)$	$\begin{cases} 1, & a = 0 \\ 0, & a > 0 \end{cases}$
Lukasiewicz	$\max(a + b - 1, 0)$	$1 - a$
Product	$a \times b$	$\begin{cases} 1, & a = 0 \\ 0, & a > 0 \end{cases}$

Let: $\delta, \delta' \in \Delta^{\mathcal{I}}$

complex concepts:

- $(\exists Q)^{\mathcal{I}}(\delta) = \sup_{\delta' \in \Delta^{\mathcal{I}}} Q^{\mathcal{I}}(\delta, \delta')$
- $(\neg B)^{\mathcal{I}}(\delta) = \ominus B^{\mathcal{I}}(\delta)$
- $\top^{\mathcal{I}}(\delta) = 1$

fuzzy conjunctive query:

$$q(\vec{x}) = \exists \vec{y}.\varphi(\vec{x}, \vec{y}) \geqslant d$$

Example

$$\mathcal{T} := \{\mathsf{Server} \sqsubseteq \exists\mathsf{hasCPU},\ \exists\mathsf{hasCPU}^- \sqsubseteq \mathsf{CPU},\ \mathsf{func}(\mathsf{hasCPU}^-)\}$$

$$\begin{aligned}\mathcal{A} := \{&\langle\mathsf{Server}(\mathsf{server}_1), 1\rangle,\ \langle\mathsf{hasCPU}(\mathsf{server}_1, \mathsf{cpu}_1), 1\rangle,\\ &\langle\mathsf{OverUtilized}(\mathsf{cpu}_1), 0.6\rangle,\ \langle\mathsf{hasCPU}(\mathsf{server}_1, \mathsf{cpu}_2), 1\rangle,\\ &\langle\mathsf{OverUtilized}(\mathsf{cpu}_2), 0.8\rangle \qquad\qquad \}\end{aligned}$$

$$\begin{aligned} q_1(x) &= \mathsf{CPU(x)}\\ q_2(x,y) &= \mathsf{hasCPU(x,y)} \land \mathsf{OverUtilized(y)}\\ q_3(x) &= \exists y\ \mathsf{hasCPU(x,y)} \land \mathsf{OverUtilized(y)}\end{aligned}$$

$$\begin{aligned} ans(q_1(x), \mathcal{O}) &= \{(\mathsf{cpu}_1, 1),\ (\mathsf{cpu}_2, 1)\}\\ ans(q_2(x,y), \mathcal{O}) &= \{(\mathsf{server}_1, \mathsf{cpu}_1, 0.6),\ (\mathsf{server}_2, \mathsf{cpu}_2, 0.8)\}\\ ans(q_3(x), \mathcal{O}) &= \{(\mathsf{server}_1, 0.8)\}.\end{aligned}$$

Approach for query answering in DL-Lite

1.) Use TBox to reformulate query q into a FOL-query $q_{\mathcal{T}}$, discard the TBox

"Compile TBox information into the query."

2.) View ABox $\mathcal{A}$ as a relational database $\mathcal{I}_{\mathcal{A}}$, where

- $A^{\mathcal{I}_{\mathcal{A}}} = \{a \mid A(a) \in \mathcal{A}\}$
- $r^{\mathcal{I}_{\mathcal{A}}} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$

3.) Evaluate $q_{\mathcal{T}}$ in the DB $\mathcal{I}_{\mathcal{A}}$ using a relational query engine.

The black box rewriting algorithm for fuzzy queries

1. Compute crisp query rewriting $(q(\vec{x}) \rightsquigarrow q_{\mathcal{T}}(\vec{x}))$

2. rewrite query $q_{\mathcal{T}}(\vec{x})$ by

   - degree variables: placeholders for degrees, e.g.: x_d
   - replacing
     - concepts in concept atoms by binary predicates
     - roles in role atoms by ternary predicates

     where new argument is a degree variable
     (to prepare for the tables storing fuzzy information)
   - degree atoms:
     - to capture the fuzzy operators, e.g.: $\Phi_{\otimes}(x_d, x'_d), \Phi_{\ominus}(x_d)$
     - to ensure conditions on degrees, e.g.: $\Phi_{\geq}(x_d, 0.3)$

The black box rewriting algorithm for fuzzy queries

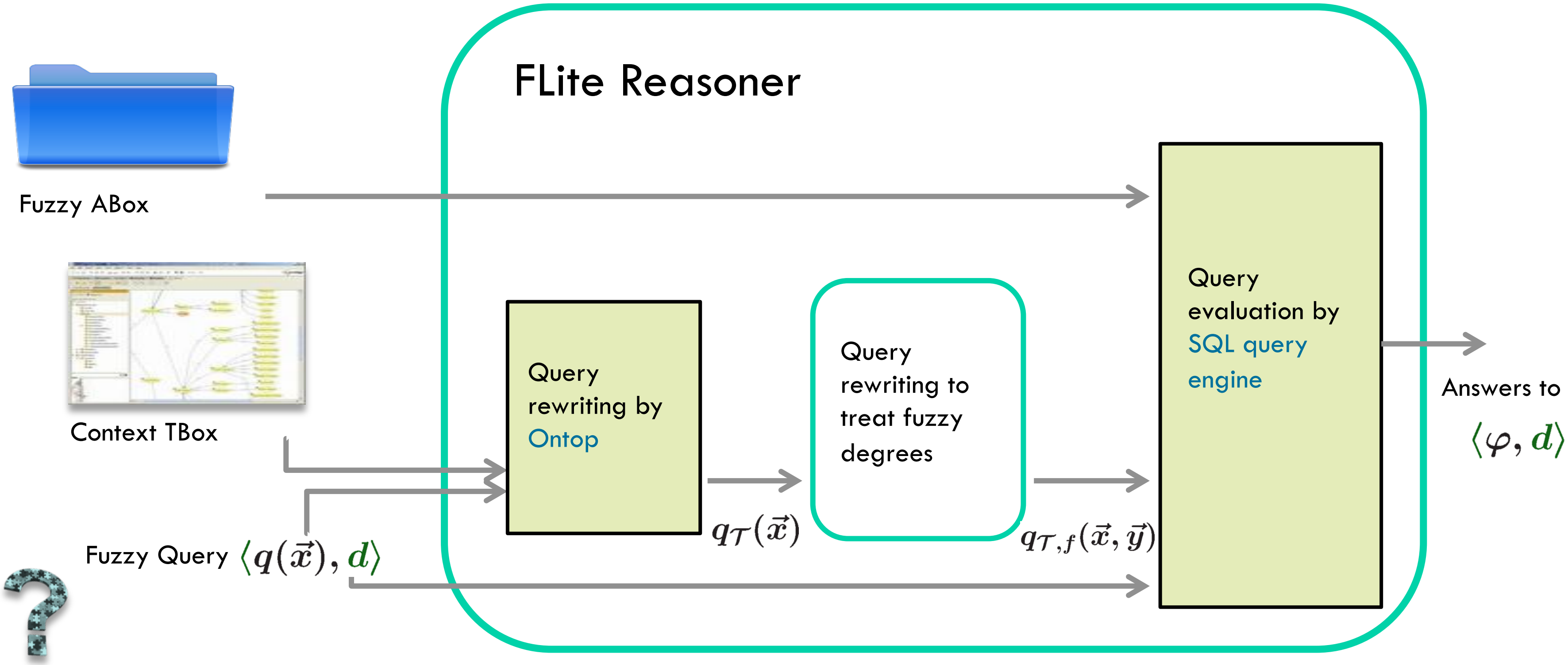
1. Compute crisp query rewriting $(q(\vec{x}) \rightsquigarrow q_{\mathcal{T}}(\vec{x}))$

2. rewrite query $(q_{\mathcal{T}}(\vec{x}) \rightsquigarrow q_{\mathcal{T},f}(\vec{x}, \vec{x_d}))$

3. Evaluate rewritten query $q_{\mathcal{T},f}(\vec{x}, \vec{x_d})$ over the 'fuzzy' database

4. Keep highest degree for each returned tuple

FLite



FLite Reasoner
Fuzzy ABox
Context TBox
Fuzzy Query $\langle q(\vec{x}), d\rangle$
Query rewriting by Ontop
$q_{\mathcal{T}}(\vec{x})$
Query rewriting to treat fuzzy degrees
$q_{\mathcal{T},f}(\vec{x},\vec{y})$
Query evaluation by SQL query engine
Answers to
$\langle \varphi, d\rangle$

The FLite reasoner

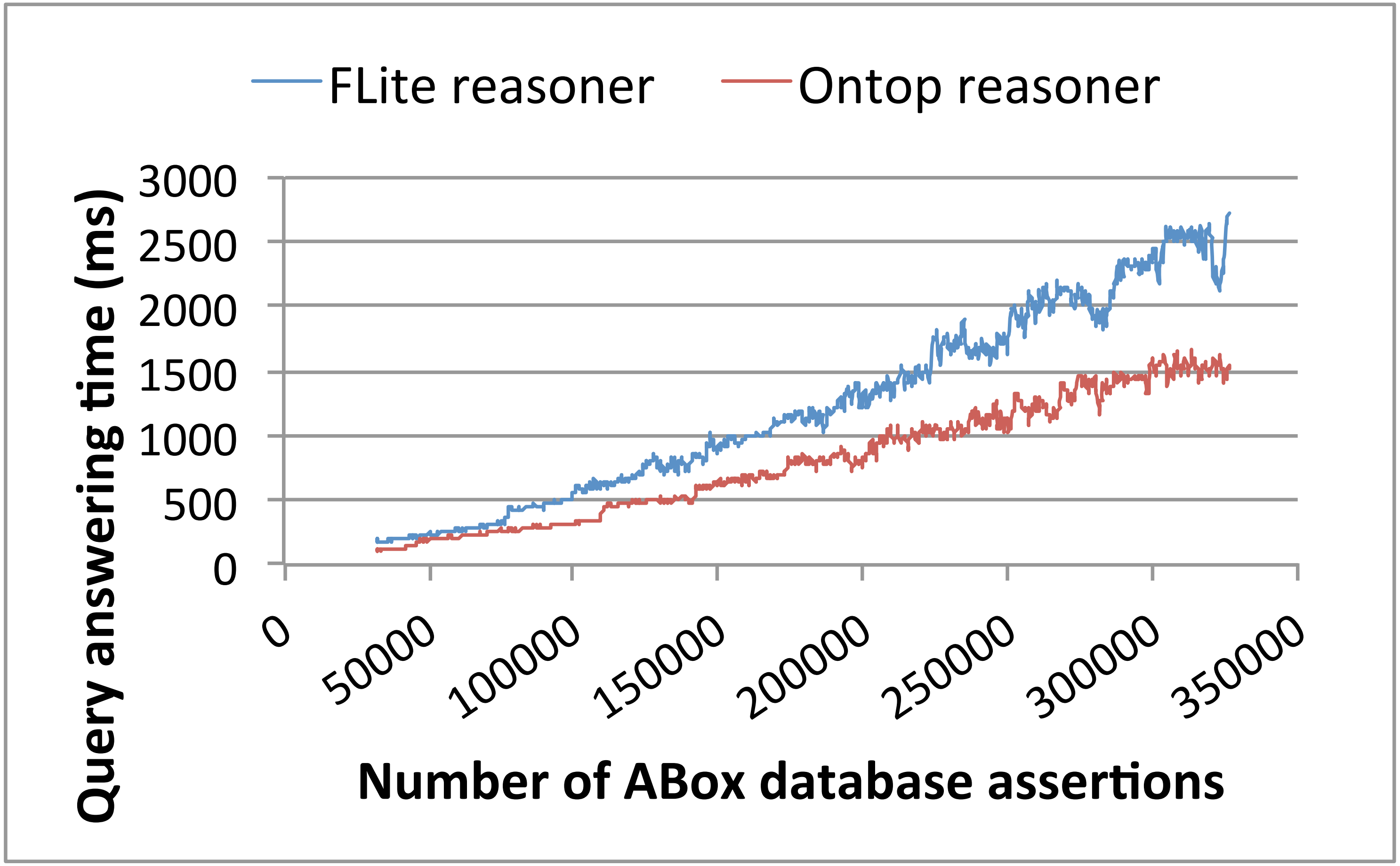
Our implementation: FLite

- implements the blackbox rewriting approach for Gödel semantics
- employs Ontop to compute the crisp rewriting of the query
- stores fuzzy ABox in database

Empirical test:

- Test ontology: TBox with 311 axioms (178 concepts, 39 roles)
  ABox (/database) with 10 tables (4 fuzzy ones)
- Test queries: fuzzy conjunctive queries
  (with 13 atoms, 9 fuzzy ones $\leadsto$ 9 degree variables)
- Test set-up: measure running times of FLite and of Ontop

Performance measurement



Temporal fuzzy queries over fuzzy DL-Lite ontologies

Recall:

- temporal operators in the TBox cause **undecidability**
- temporal information given by **sequence of (fuzzy) ABoxes**
- navigate "backwards" on the sequence

Temporal queries use the **operators from LTL**:

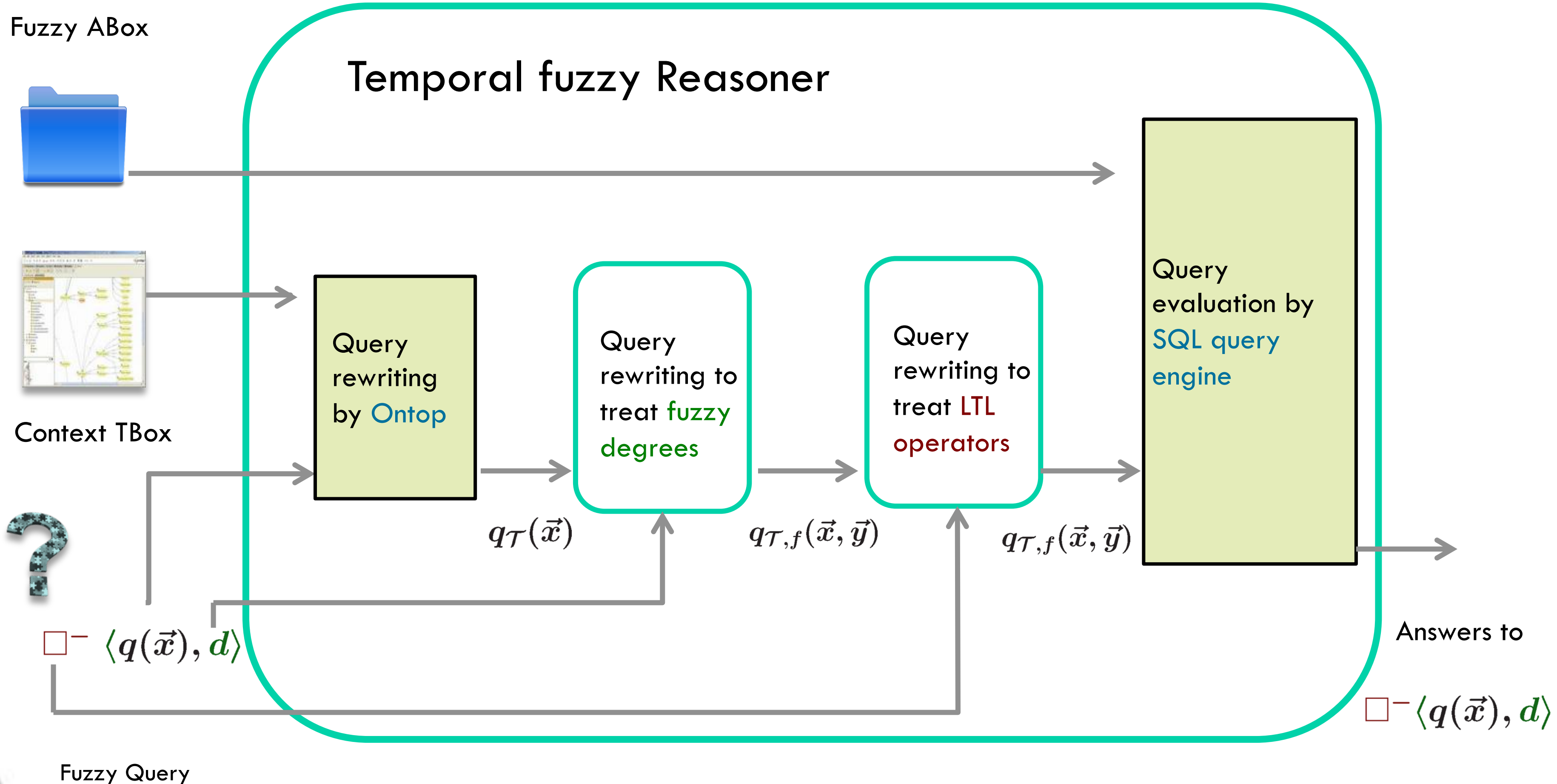
$\Box^-$ 'always in the past'

$\Diamond^-$ 'eventually in the past'

$\bigcirc^-$ 'previous'

Temporal query: query "wrapped" in LTL operators $\bigcirc^- \, \Box^- \, \langle\, q(\vec{x}), d \rangle$

FLite and QuAnTOn combined



Conclusions

- introduced a decidable variant of temporal fuzzy query answering
- proposed an pragmatic approach to temporal fuzzy query answering that enriches crisp query rewritings for DL-Lite$_R$
- method implemented on top of OnTop reasoner
- first evaluation suggest: acceptable overhead for fuzzy information Temporal fuzzy QA?