Managing Change in Graph-structured Data Using Description Logics

Diego Calvanese

Research Centre for Knowledge and Data (KRDB)
Free University of Bozen-Bolzano, Italy

Based on joint work with: *S. Ahmetaj, M. Ortiz, M. Šimkus*

28th International Workshop on Description Logics (DL 2016)

Cape Town, South Africa, April 22, 2016

Motivations for this research

Comes from two important "trends" in data and information management:

1. Graph-structured data (GSD)
2. Dealing with dynamic systems, while properly taking into account data

What we are going to do here:

- We argue that research in DLs has provided important contributions to both settings.
- We combine the two aspects in a novel setting based on DLs for the management of evolving GSD.

unibz

Outline

unibz

Outline

unibz

Graph-structured data are everywhere

The data underlying many settings is inherently graph structured:

- Web data
- Social data
- RDF data
- Open linked data
- XML data
- States of a program (pointer structure)

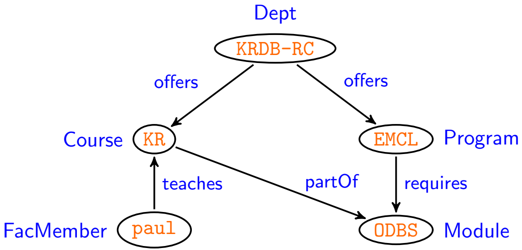
We need formalisms, techniques, and tools to properly manage GSD:

- **modeling languages and constraints**
- query languages
- efficient query answering
- **dealing with evolving GSD**

Graph-structured data is nothing new for us

A graph-structured database instance =
An edge and node labeled graph =
A finite relational structure with unary and binary relations only =
A **finite** DL interpretation

Example:



Path constraints for GSD

The problem of specifying and reasoning over integrity constraints for GSD has been addressed in the database community.

Path constraints [Abiteboul and Vianu 1999; Buneman, Fan, and Weinstein 2000; Grahne and Thomo 2003]

- Make use of regular expressions P, interpreted over GSD instances $\mathcal{I}$:

  $P^{\mathcal{I}}$ = set of **pairs** of nodes connected by a **path in $\mathcal{I}$** whose **labels** spell a word in P.

- Path constraints φ come in two forms: $P_\ell \subseteq P_r$ $\quad$ $[P_p](P_\ell \subseteq P_r)$
- Semantics:

$$(P_\ell \subseteq P_r)^{\mathcal{I}} = \{n \mid \text{for all } n',\ \text{if } (n,n') \in P_\ell^{\mathcal{I}} \text{ then } (n,n') \in P_r^{\mathcal{I}}\}$$

$$([P_p](P_\ell \subseteq P_r))^{\mathcal{I}} = \{n \mid \text{for all } n_1,\ \text{if } (n,n_1) \in P_p^{\mathcal{I}},\ \text{then for all } n',\ \text{if } (n_1,n') \in P_\ell^{\mathcal{I}} \text{ then } (n_1,n') \in P_r^{\mathcal{I}}\}$$

unibz

Reasoning with path constraints

- **Global** semantics: $\mathcal{I} \models \varphi$, if every node is in $\varphi^{\mathcal{I}}$
- **Pointed** semantics: $\mathcal{I}, a \models \varphi$, if $a^{\mathcal{I}} \in \varphi^{\mathcal{I}}$ for some node a

Central problem: implication of path constraints

Given a set Γ of path constraints, and a path constraint φ (and a node a), decide:

- **Unrestricted implication:** Does $\Gamma(,a) \models \varphi$?
  i.e., $\mathcal{I}(,a) \models \varphi$ for every $\mathcal{I}$ such that $\mathcal{I}(,a) \models \Gamma$
- **Finite implication:** Does $\Gamma(,a) \models_{\mathit{fin}} \varphi$?
  same as above, but over **finite** instances

unibz

Implication of path constraint is undecidable

Finite and unrestricted **implication** of path constraints shown **undecidable** by [Buneman, Fan, and Weinstein 2000; Grahne and Thomo 2003]:

- for pointed semantics, and general constraints $[P_p](P_\ell \subseteq P_r)$
- for global semantics, even for prefix-empty, word constraints $w_\ell \subseteq w_r$

$\rightsquigarrow$ Decidability requires both pointed semantics and empty prefixes.

Recently, undecidability has been tightened to rather simple (word) constraints, of the forms [C., Ortiz, and Simkus 2016]:

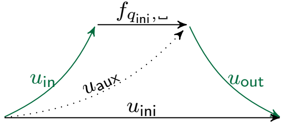
	$[r](r_1 \circ r_2 \subseteq r_3)$	$[r](r_1 \subseteq r_2 \circ r_3)$	(for both semantics)
or:	$r_1 \circ r_2 \subseteq r_3$	$r_1 \subseteq r_2 \circ r_3$	(for global semantics)

where all r are role names (i.e., no ε, no inverse roles).

unibz

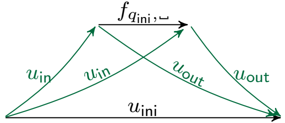
Core ideas of the undecidability proof

- Based on encoding Turing Machine computations.
- Constraints Γ generate the TM computation grid.
- Employs spy point technique known from DLs with nominals [Tobies 2001]: Spy points are connected to all nodes of the domain, and are used to enforce conditions on such nodes.



Creating the arc $f_{q_{\mathsf{ini}},\llcorner\lrcorner}$ for the first tape position

$$u_{\mathsf{ini}} \subseteq u_{\mathsf{aux}} \circ u_{\mathsf{out}}$$
$$u_{\mathsf{aux}} \subseteq u_{\mathsf{in}} \circ f_{q_{\mathsf{ini}},\llcorner\lrcorner}$$



Connecting the new arc to the spy-points

$$u_{\mathsf{in}} \circ f_{q_{\mathsf{ini}},\llcorner\lrcorner} \subseteq u_{\mathsf{in}}$$
$$[u_{\mathsf{in}}](f_{q_{\mathsf{ini}},\llcorner\lrcorner} \circ u_{\mathsf{out}} \subseteq u_{\mathsf{out}})$$

- Conditions to correctly encode the TM computation are then enforced on the grid points, making also use of "diagonals".

Impact on inference over TGDs

The previous result can easily be rephrased in terms of tuple-generating dependencies (TGDs):

- $r_1 \circ r_2 \subseteq r_3$ is equivalent to $r_1(x,y), r_2(y,z) \rightarrow r_3(x,y)$
- $r_1 \subseteq r_2 \circ r_3$ is equivalent to $r_1(x,y) \rightarrow \exists z.r_2(x,z), r_3(z,y)$

Undecidability of TGD entailment and of query answering under TGDs

(Finite) entailment of TGDs, and (finite) entailment of atomic queries under TGDs are undecidable already for TGDs of the forms:

$$r_1(x,y), r_2(y,z) \rightarrow r_3(x,y) \qquad r_1(x,y) \rightarrow \exists z.r_2(x,z), r_3(z,y)$$

unibz

Expressive DLs for constraints on GSD

Expressive DLs are well suited to express constraints on GSD:

- powerful features for structuring the domain into classes (i.e., concepts)
- complex conditions for typing binary relations (i.e., roles)
- when resorting to expressive DLs with regular expressions over roles, we also have a mechanism to navigate the graph

Let us consider one such DL: $\mathcal{ALCOI}b_{reg}$,

also known as $\mathcal{ZOI}$.

$\mathcal{ZOI}$ Is closely related to PDL and (positive) regular XPath.

unibz

The DL $\mathcal{ZOI}$

- The vocabulary of $\mathcal{ZOI}$ has three alphabets:
  - $\mathsf{N_C}$ concept names unary predicates, node symbols
  - $\mathsf{N_R}$ role names binary predicates, edge symbols
  - $\mathsf{N_I}$ individuals constants, node names

  Note: each nodes and edge can be labeled with a **set** of symbols.

- **Concepts**, i.e., node formulas $A \in \mathsf{N_C}, \quad a \in \mathsf{N_I}$

$$C, C' \longrightarrow A \mid \{a\} \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \forall P.C \mid \exists P.C$$

- **Roles**, i.e., path formulas $r \in \mathsf{N_R}, \quad a, b \in \mathsf{N_I}$, S: simple role

$$S, S' \longrightarrow r \mid r^- \mid \{(a,b)\} \mid S \cap S' \mid S \cup S' \mid S \setminus S'$$
$$P, P' \longrightarrow \nabla \mid \varepsilon \mid id(C) \mid S \mid P \cup P' \mid P \circ P' \mid P^*$$

unibz

Formulas in $\mathcal{ZOI}$

An **atomic formula** α of $\mathcal{ZOI}$ corresponds to a TBox or ABox assertion:

- Inclusions between concepts and between simple roles:

$$C_1 \sqsubseteq C_2 \qquad\qquad S_1 \sqsubseteq S_2$$

- Assertions on concepts and on simple roles

$$C(a) \qquad\qquad S(a, b)$$

A $\mathcal{ZOI}$ **knowledge base** is a boolean combination of atomic formulas:

$$\mathcal{K} \longrightarrow \alpha \mid \mathcal{K} \wedge \mathcal{K}' \mid \mathcal{K} \vee \mathcal{K}' \mid \dot{\neg}\mathcal{K}$$

Semantics: standard one for DLs.

Example of constraints expressible in $\mathcal{ZOI}$

Complex domain and range restrictions

$$\exists \mathsf{offers}.\mathsf{Course} \sqsubseteq \mathsf{Dept} \qquad \mathsf{Department} \sqsubseteq \forall \mathsf{offers}.(\mathsf{Program} \sqcup \mathsf{Course})$$

$$\exists \mathsf{requires}.\top \sqsubseteq \mathsf{Program} \qquad \top \sqsubseteq \forall \mathsf{requires}.(\exists \mathsf{partOf}^{-*}.\mathsf{Course})$$

Conditions requiring navigation on the graph

$$\begin{aligned}
\mathsf{Course} &\sqsubseteq \exists \mathsf{taughtBy}.\mathsf{FacMember}\\
\exists(\mathsf{partOf}^{-*} \circ \mathsf{requires}).\mathsf{Program} &\sqsubseteq \exists \mathsf{offers}^{-}.\mathsf{Department}\\
\mathsf{Course} \sqcap \exists \mathsf{requires}^{-}.\mathsf{UndergradProgram} &\sqsubseteq\\
&\exists \mathsf{teaches}^{-}.(\exists(\mathsf{memberOf} \circ \mathsf{partOf}^{*}).\mathsf{Institute})
\end{aligned}$$

unibz

Expressing path constraints in $\mathcal{ZOI}$

For **empty prefixes** and **pointed semantics**:

$$\varphi = P_\ell \subseteq P_r \qquad \rightsquigarrow \qquad \mathcal{T}_\varphi = \{a\} \sqsubseteq \forall P_\ell.\exists \mathsf{inv}(P_r).\{a\}$$

Lemma

Let Γ be set of constraints, φ a constraint, all prefix-empty, and $a \in \mathsf{N_I}$. Then:

$$\Gamma \models_{(fin)} \varphi \qquad \text{iff} \qquad (\bigwedge_{\gamma \in \Gamma} \mathcal{T}_\gamma) \wedge \dot{\neg} \mathcal{T}_\varphi \quad \text{is not (finitely) satisfiable}$$

unibz

Complexity of path-constraint implication

From satisfiability of $\mathcal{ZOI}$ in **ExpTime** , we get:

Theorem ([C., Ortiz, and Simkus 2016])

The **implication** of **prefix-empty** path constraints under **pointed semantics**, is decidable in **ExpTime**

Previous known bound: N2ExpTime

What about finite implication?

- Finite model reasoning for $\mathcal{ZOI}$ has not been considered so far.
- However, it turns out that $\mathcal{ZOI}$ has the **finite model property** – Proof needs ideas from PDL and from 2-variable fragment [C., Ortiz, and Simkus 2016].

Theorem ([C., Ortiz, and Simkus 2016])

The **finite implication** of **prefix-empty** path constraints under **pointed semantics**, is decidable in **ExpTime**

Other classes of path constraints

We cannot anymore make use of a nominal to encode the inclusion of the left-tail in the right-tail

- under global semantic, or
- in the presence of a prefix.

To express other path constraints, we need to resort to an extension of $\mathcal{ZOI}$:

We can capture all forms of path constraints in $\mathcal{ZOI}$ extended with **role difference** for non-simple roles:

$$\varphi = [P_p](P_\ell \subseteq P_r) \qquad \rightsquigarrow \qquad C_\varphi = \forall P_p.(\forall (P_\ell \setminus P_r).\bot)$$

Lemma

Let Γ be a set of constraints, φ a constraint, and $a \in \mathsf{N_I}$.
Then:

$$\Gamma, a \models_{(\mathit{fin})} \varphi \quad \text{iff} \quad \left(\textstyle\bigsqcap_{\gamma\in\Gamma} C_\gamma \sqcap \neg C_\varphi\right)(a) \quad \text{is not (finitely) satisfiable,}$$

$$\Gamma \models_{(\mathit{fin})} \varphi \quad \text{iff} \quad \dot{\neg}\left(\textstyle\bigsqcap_{\gamma\in\Gamma} C_\gamma \sqsubseteq C_\varphi\right) \quad \text{is not (finitely) satisfiable}$$

Outline

Dynamic systems taking into account data

Traditional approach to model dynamic systems: **divide et impera** of

- static, data-related aspects
- dynamic, process/interaction-related aspects

These two aspects traditionally treated separately by different communities:

- Data management community:
  data modeling, constraints, analysis deal (mostly) with static aspects
- (Business) process management and verification community:
  data is abstracted away

Reasoning about evolving data and knowledge

However, the KR community, and also the DL one, traditionally has paid attention to the combination of static and dynamic aspects:

- The combination in a single logical theory is well-known to be difficult [Wolter and Zakharyaschev 1999; Gabbay et al. 2003]
- Reasoning about actions in the Situation Calculus, cf. [Reiter 2001]
- Automated planning, cf. [Ghallab, Nau, and Traverso 2004]
- DL-based action languages [Baader, Lutz, et al. 2005; Baader and Zarriess 2013]
- Data Centric Dynamic Systems [Bagheri Hariri, C., De Giacomo, et al. 2013]
- Knowledge and Action Bases [Bagheri Hariri, C., Montali, et al. 2013]
- Bounded Situation Calculus [De Giacomo, Lesperance, and Patrizi 2012]

Relevant assumptions about the system behaviour

In the dynamic setting, there is a huge variety of different assumptions made, that deeply affect the inference services of interest and their computational properties:

1. System dynamics specified procedurally (e.g., through a finite state machine) vs. declaratively (e.g., through a set of condition-action rules).
2. Simple vs. complex actions.
3. Actions operate on the single instances (i.e., models), as opposed to adopting the functional approach [Levesque 1984].
4. Completely specified initial state vs. incomplete initial state.
5. Deterministic vs. non-deterministic effects of actions.
6. During system execution, new objects may enter the system or not.
7. The intentional knowledge about the system is fixed vs. changes.

unibz

The setting we adopt here

In our setting, we specialize the above options as follows:

- We assume to have available a finite set of parametric actions.
- Actions might be complex, and allow for checking conditions.
- Actions operate on the single instances.
- We assume incomplete information in the initial state, i.e., the initial state is not specified completely, and we are interested in reasoning over all possible initial states.
- Our actions are deterministic.
- Our actions do not incorporate new objects in the system ... but (when relevant) we allow for arbitrarily extending the domain in the initial state.
- The intentional knowledge might change, since it is affected in complex ways by the extensional knowledge.

Reasoning services of interest

We consider several classical reasoning services that are of relevance in this setting:

- Verification.
- Existence of a plan.
- Existence of a plan from a given precondition.
- Conformant planning.
- Variants of the previous three, where we impose a priori a finite bound on the length of the plan.

Reasoning services – Verification

Let $\mathcal{K}$ be a KB, $\mathcal{I}$ a finite interpretation for $\mathcal{K}$, and α a (possibly complex) action.

Then $\alpha(\mathcal{I})$ denotes the interpretation obtained by applying α to $\mathcal{I}$.

Verification (V)

Given $\mathcal{K}$ and α, is α **$\mathcal{K}$-preserving**?
I.e., do we have that, for every finite interpretation $\mathcal{I}$, if $\mathcal{I} \models \mathcal{K}$ then $\alpha(\mathcal{I}) \models \mathcal{K}$?

Reasoning services – Plan existence

Let $\mathcal{K}$ be a KB, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ a finite interpretation for $\mathcal{K}$, and *Act* a finite set of actions.

Plan

A finite sequence $\alpha_1 \circ \cdots \circ \alpha_n$ of actions in *Act* is a **plan** (of length n) for $\mathcal{K}$ from $\mathcal{I}$, if there exists a finite set Δ such that $(\alpha_1 \circ \cdots \circ \alpha_n)(\mathcal{I}') \models \mathcal{K}$, where $\mathcal{I}' = \langle \Delta^{\mathcal{I}} \cup \Delta, \cdot^{\mathcal{I}} \rangle$.

Note: Δ allows for extending the interpretation domain, which might account for new objects needed in the plan.

Planning (P) and **Bounded planning (Pb)**

- Given *Act*, $\mathcal{I}$, and $\mathcal{K}$, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$.
- Given *Act*, $\mathcal{I}$, $\mathcal{K}$, and a bound k, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$ where $|\Delta|$ is at most k.

Reasoning services – Planning with incompleteness

In this variant of planning, we are not given the initial interpretation, but want to check existence of a plan from some interpretation satisfying a given precondition.

Planning with incompleteness (PI) and
Bounded planning with infcompleteness (PIb)

- Given *Act*, $\mathcal{I}$, $\mathcal{K}$, and $\mathcal{K}_{pre}$, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$, for some finite $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}_{pre}$.
- Given *Act*, $\mathcal{I}$, $\mathcal{K}$, $\mathcal{K}_{pre}$, and a bound ℓ, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$ of length at most ℓ, for some finite $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}_{pre}$.

unibz

Outline

1. Motivations
2. DLs for Graph-structured Data
3. Reasoning in Dynamic Systems
4. DLs for Evolving Graph Structured Data
5. Conclusions

Update language for GSD

We consider an update language for GSD that allows for various types of actions:

- **Adding** the result of a concept/role to an atomic concept/role, resp.
- **Removing** the result of a concept/role from an atomic concept/role, resp.
- **Conditional execution / composition / parameters**.

Update Language for GSD – Example

Example

A complex action with input parameters x, y, z that transfers an employee x from a project y to the project z:

$$\alpha = \overbrace{(\mathsf{Employee}(x) \land \mathsf{Project}(y) \land \mathsf{Project}(z) \land \mathsf{worksFor})(x,y)}^{\text{Condition}}\ ?$$
$$\mathsf{worksFor} \ominus \{(x,y)\} \cdot \mathsf{worksFor} \oplus \{(x,z)\} : \varepsilon$$

- α checks if x is an Employee, y and z are Projects, and x worksFor y.
- If **yes**, it removes the worksFor link between x and y, and creates a worksFor link between x and z.
- If **no** (i.e., any of the checks fails), it does nothing.

Recall: We use $\alpha(\mathcal{I})$ to denote the result of applying α to $\mathcal{I}$.

unibz

Result of conditional action – Example

Before being executed, the action in grounded.

Example of execution of a grounded action:

Given:

$$\alpha = (\mathsf{Employee}(e) \wedge \mathsf{Project}(p_1) \wedge \mathsf{Project}(p_2) \wedge \mathsf{worksFor}(e, p_1))\ ?\ \mathsf{worksFor} \ominus \{(e, p_1)\} \cdot \mathsf{worksFor} \oplus \{(e, p_2)\} : \varepsilon$$

$$\mathcal{I} = \{\ \mathsf{Employee}(e),\ \mathsf{worksFor}(e, p_1),\ \mathsf{Project}(p_1),\ \mathsf{Project}(p_2)\ \}$$

Result:

$$\alpha(\mathcal{I}) = \{\ \mathsf{Employee}(e),\ \mathsf{worksFor}(e, p_2),\ \mathsf{Project}(p_1),\ \mathsf{Project}(p_2)\ \}$$

unibz

Solving the verification problem

The verification problem can be reduced to finite (un)satisfiability of a $\mathcal{ZOI}$ KB using a form of **regression**.

Let $\mathcal{K}_{L \leftarrow L'}$ be the KB obtained from $\mathcal{K}$ by replacing each occurrence of L by L'.

Transformation $\mathsf{TR}(\mathcal{K}, \alpha)$ of a KB $\mathcal{K}$ via an action α is defined inductively:

$$
\begin{array}{rcl}
\mathsf{TR}(\mathcal{K}, \epsilon) & = & \mathcal{K} \\
\mathsf{TR}(\mathcal{K}, (A \oplus C) \cdot \alpha) & = & (\mathsf{TR}(\mathcal{K}, \alpha))_{A \leftarrow A \sqcup C} \\
\mathsf{TR}(\mathcal{K}, (A \ominus C) \cdot \alpha) & = & (\mathsf{TR}(\mathcal{K}, \alpha))_{A \leftarrow A \sqcap \neg C} \\
\mathsf{TR}(\mathcal{K}, (r \oplus P) \cdot \alpha) & = & (\mathsf{TR}(\mathcal{K}, \alpha))_{r \leftarrow r \cup P} \\
\mathsf{TR}(\mathcal{K}, (r \ominus P) \cdot \alpha) & = & (\mathsf{TR}(\mathcal{K}, \alpha))_{r \leftarrow r \setminus P} \\
\mathsf{TR}(\mathcal{K}, (\mathcal{K}_1 ? \alpha_1 : \alpha_2)) & = & (\dot{\neg}\mathcal{K}_1 \vee \mathsf{TR}(\mathcal{K}, \alpha_1)) \wedge (\mathcal{K}_1 \vee \mathsf{TR}(\mathcal{K}, \alpha_2))
\end{array}
$$

unibz

Transforming a KB via an action – Example

Example

$$\begin{aligned}\mathcal{K}_1 = \quad & (\text{Project} \sqsubseteq \text{ActiveProject} \sqcup \text{ConcludedProject}) \land \\ & (\text{Employee} \sqsubseteq \text{ProjectEmployee} \sqcup \text{PermanentEmployee}) \land \\ & (\exists \text{worksFor}.\text{Project} \sqsubseteq \text{ProjectEmployee})\end{aligned}$$

$$\begin{aligned}\alpha_1 = \; & \text{ActiveProject} \ominus \{\text{optique}\} \cdot \\ & \text{ConcludedProject} \oplus \{\text{optique}\} \cdot \\ & \text{ProjectEmployee} \ominus \exists \text{worksFor}.\{\text{optique}\}\end{aligned}$$

$$\begin{aligned}\text{TR}(\mathcal{K}_1, \alpha_1) = \; & \\ & (\text{Project} \sqsubseteq (\text{ActiveProject} \sqcap \neg\{\text{optique}\}) \\ & \qquad \sqcup (\text{ConcludedProject} \sqcup \{\text{optique}\})) \land \\ & (\text{Employee} \sqsubseteq (\text{ProjectEmployee} \sqcap \neg\exists \text{worksFor}.\{\text{optique}\}) \\ & \qquad \sqcup \text{PermanentEmployee}) \land \\ & (\exists \text{worksFor}.\text{Project} \sqsubseteq (\text{ProjectEmployee} \sqcap \neg\exists \text{worksFor}.\{\text{optique}\}))\end{aligned}$$

unibz

Reducing verification to unsatisfiability

For a ground action α and a KB $\mathcal{K}$, the transformation $\mathsf{TR}(\mathcal{K}, \alpha)$ correctly captures the meaning of α.

Lemma

For every ground action α and interpretation $\mathcal{I}$:

$$\alpha(\mathcal{I}) \models \mathcal{K} \quad \text{iff} \quad \mathcal{I} \models \mathsf{TR}(\mathcal{K}, \alpha).$$

Theorem

For every action α and KB $\mathcal{K}$

α is $\mathcal{K}$-preserving
iff
$\mathcal{K} \wedge \dot{\neg}\mathsf{TR}(\mathcal{K}, \alpha_g)$ is finitely unsatisfiable

where α_g is obtained from α by replacing each variable with a fresh individual name not occurring in α and $\mathcal{K}$.

Deciding verification

In order to obtain from the previous result decidability of verification, we need to ensure that $\mathsf{TR}(\mathcal{K}, \alpha_g)$ is expressible in $\mathcal{ZOI}$.

Key issue: form of basic actions: $(A \oplus C)$, $(A \ominus C)$, $(r \oplus P)$, $(r \ominus P)$

- We can allow for arbitrary concepts C to be added and removed via $(A \oplus C)$ and $(A \ominus C)$.
- Instead, in basic actions $(r \oplus P)$ and $(r \ominus P)$, the role P **must be simple**: role name, inverse role name, $\{(a, b)\}$, and their boolean combination, but no concatenation or transitive closure.

Complex actions containing these restricted basic actions are called **role-simple**.

Examples of role-simple actions:

$\mathsf{friendOf} \ominus (\ \mathsf{hasAunt} \cap \mathsf{sendsCandyCrushInv}^{-}\ )$

$\mathsf{friendOf} \ominus (\ \mathsf{supports}|\{\mathit{Trump}\}\ )$

$\mathsf{preferredAIColl} \oplus \exists\big(\mathsf{collabWith}|(\neg\exists \mathsf{projWith}.\{\mathit{Darpa}\})\big)^{*}.\mathsf{ExpertAI}$

Complexity of verification

Theorem

For $\mathcal{ZOI}$ KBs and role-simple actions, verification is ExpTime-complete.

- The lower bound follows from the fact that a KB $\mathcal{K}$ is finitely satisfiable iff $(A' \oplus \{o\})$ is not $(\mathcal{K} \wedge (A \sqsubseteq \neg A') \wedge (o : A))$-preserving, where A, A', and o are fresh.
- For the upper bound:
  - Observe that the KB TR$(\mathcal{K}, \alpha)$ might be exponential in α, since conditional actions lead to duplication of $\mathcal{K}$.
  - However, the resulting KB can be put in disjunctive normal form, with exponentially many conjunctions of atoms, each of polynomial size.
  - Hence, once can run an exponential number of checks on polynomial-size KBs, each of which takes at most exponential time.
  - The resulting algorithm runs in single exponential time.

unibz

Complexity of verification

When actions are not role-simple, i.e., contain role concatenation, or transitive closure, verification becomes undecidable.

Theorem

Deciding whether α is $\mathcal{K}$-preserving is **undecidable**, even when

- $\mathcal{K}$ consists of a single fact $r(a, b)$, and
- α is just a sequence of basic actions of the form

$$(r \oplus P) \qquad\qquad (r \ominus P)$$

with P a sequence of one or two symbols.

unibz

A restricted setting based on *DL-Lite*

We restrict the setting so as to simplify verification.

A *DL-Lite*$^+_\mathcal{R}$ KB is a KB satisfying the following conditions:

- Concept and role inclusions and disjointness are those allowed in standard *DL-Lite*$_\mathcal{R}$.
- In concept assertions $C(a)$, the concept C might be a boolean combination of concept names A, unqualified existentials $\exists r$, and nominals $\{a'\}$.
- $\dot\neg$ may occur only in front of ABox assertions (while $\wedge$ and $\vee$ may be applied freely on formulae).

Localized actions

A localized action is one where in $\mathcal{K}?\,\alpha_1{:}\alpha_2$, the KB $\mathcal{K}$ is a boolean combination of ABox assertions (hence, it may not contain concept or role inclusions).

unibz

Verification for $\mathit{DL\text{-}Lite}_{\mathcal{R}}^{+}$ KBs and localized actions

Theorem

Verification for $\mathit{DL\text{-}Lite}_{\mathcal{R}}^{+}$ KBs and localized actions can be reduced in linear time to finite unsatisfiability of $\mathit{DL\text{-}Lite}_{\mathcal{R}}^{+}$ KBs.

Intuition:

1. Construct as before $\mathcal{K}' = \mathcal{K} \wedge \dot{\neg}\mathsf{TR}(\mathcal{K}, \alpha^*)$.
2. Push $\dot{\neg}$ inside so that it occurs in front of inclusions and assertions only.
3. Replace each $\dot{\neg}(B_1 \sqsubseteq B_2)$ by $o : B_1 \sqcap \neg B_2$, where o is fresh,
   and each $\dot{\neg}(r_1 \sqsubseteq r_2)$ by $(o, o') : r_1 \setminus r_2$, where o, o' are fresh.

We obtain a $\mathit{DL\text{-}Lite}_{\mathcal{R}}^{+}$ KB that we can check for unsatisfiability.

Complexity of verification in the *DL-Lite* setting

Theorem

Finite satisfiability of $\mathit{DL\text{-}Lite}^{+}_{\mathcal{R}}$ KBs is NP-complete.

- NP-hardness is immediate.
- Membership in NP: we define a non-deterministic polynomial time rewriting procedure that transforms a $\mathit{DL\text{-}Lite}^{+}_{\mathcal{R}}$ KB $\mathcal{K}$ into a $\mathit{DL\text{-}Lite}_{\mathcal{R}}$ KB $\mathcal{K}'$, s.t., $\mathcal{K}$ is satisfiable iff there exists a $\mathcal{K}'$ that is satisfiable.

Theorem

Verification for $\mathit{DL\text{-}Lite}^{+}_{\mathcal{R}}$ KBs and localized actions is coNP-complete.

unibz

Intractability in a very restricted setting

coNP-hardness does **not** depend on intractability of *DL-Lite*$^{+}_{\mathcal{R}}$!

Theorem

Verification is coNP-hard already when:

- KBs consist of a conjunction of concept disjointness assertions: $(A_0 \sqsubseteq \neg A'_0) \wedge \cdots \wedge (A_n \sqsubseteq \neg A'_n)$, and
- actions are localized ground sequences of basic actions of the forms $(A \oplus C)$ and $(A \ominus C)$.

The proof is by a reduction of non-3-colorability.

Complexity of planning and conformant planning

Planning (P) and Planning with incompleteness (CI)

1. Given *Act*, $\mathcal{I}$, and $\mathcal{K}$, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$.
2. Given *Act*, $\mathcal{I}$, $\mathcal{K}$, and $\mathcal{K}_{pre}$, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$, for some finite $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}_{pre}$.

- Undecidable in general, even for *DL-Lite*$^{+}_{\mathcal{R}}$ KBs and simple actions.
- (1) is PSPACE-complete, when a bound on the number of fresh values is given.
- (2) is EXPTIME-complete, when a bound on the lenght of the plan is given. It is NP-complete for *DL-Lite*$^{+}_{\mathcal{R}}$.

Outline

Summing up

Main observations

- By exploiting DL techniques and tools, one can obtain strong **decidability and complexity** results for reasoning about the **evolving GSD under constraints**.
- This is an indication that the capabilities of DLs in managing the structure of data can be extended also towards managing the dynamics of data.

Further work

- Investigate further useful fragments with **lower complexity**.
- Can we extend the **update language** while preserving decidability?
  - while loops
  - richer queries than concepts and roles
- Can we consider other forms of **constraints**
  - keys
  - identification constraints

Thank you for your attention!

References I

[1] Serge Abiteboul and Victor Vianu. "Regular Path Queries with Constraints". In: *J. of Computer and System Sciences* 58.3 (1999), pp. 428–452.

[2] Peter Buneman, Wenfei Fan, and Scott Weinstein. "Path Constraints in Semistructured Databases". In: *J. of Computer and System Sciences* 61.2 (2000), pp. 146–193. ISSN: 0022-0000. DOI: 10.1006/jcss.2000.1710. URL: http://www.sciencedirect.com/science/article/pii/S0022000000917100.

[3] Gösta Grahne and Alex Thomo. "Query Containment and Rewriting using Views for Regular Path Queries under Constraints". In: *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 2003, pp. 111–122.

References II

[4] Diego C., Magdalena Ortiz, and Mantas Simkus. "Verification of Evolving Graph-structured Data under Expressive Path Constraints". In: *Proc. of the 19th Int. Conf. on Database Theory (ICDT)*. Vol. 48. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, 15:1–15:19.

[5] Stephan Tobies. "Complexity Results and Practical Algorithms for Logics in Knowledge Representation". PhD thesis. LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[6] Frank Wolter and Michael Zakharyaschev. "Temporalizing Description Logic". In: *Frontiers of Combining Systems*. Ed. by D. Gabbay and M. de Rijke. Studies Press/Wiley, 1999, pp. 379–402.

[7] Dov Gabbay et al. *Many-dimensional Modal Logics: Theory and Applications*. Elsevier Science Publishers, 2003.

[8] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

unibz

References III

[9] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning – Theory and Practice*. Elsevier, 2004. ISBN: 978-1-55860-856-6.

[10] Franz Baader, Carsten Lutz, et al. "Integrating Description Logics and Action Formalisms: First Results". In: *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI)*. 2005, pp. 572–577.

[11] Franz Baader and Benjamin Zarriess. "Verification of Golog Programs over Description Logic Actions". In: *Proc. of the 9th Int. Symp. on Frontiers of Combining Systems (FroCoS)*. Vol. 8152. Lecture Notes in Computer Science. Springer, 2013, pp. 181–196. ISBN: 978-3-642-40884-7. DOI: 10.1007/978-3-642-40885-4_12. URL: http://dx.doi.org/10.1007/978-3-642-40885-4_12.

[12] Babak Bagheri Hariri, Diego C., Giuseppe De Giacomo, et al. "Verification of Relational Data-Centric Dynamic Systems with External Services". In: *Proc. of the 32nd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*. 2013, pp. 163–174.

unibz

References IV

[13] Babak Bagheri Hariri, Diego C., Marco Montali, et al. "Description Logic Knowledge and Action Bases". In: *J. of Artificial Intelligence Research* 46 (2013), pp. 651–686. ISSN: 1076-9757. DOI: 10.1613/jair.3826.

[14] Giuseppe De Giacomo, Yves Lesperance, and Fabio Patrizi. "Bounded Situation Calculus Action Theories and Decidable Verification". In: *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*. 2012, pp. 467–477.

[15] Hector J. Levesque. "Foundations of a Functional Approach to Knowledge Representation". In: *Artificial Intelligence* 23 (1984), pp. 155–212.

unibz