Reasoning over Evolving Graph-structured Data Under Constraints

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*

Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2016)

Panama City, Panama, June 6–10, 2016

Outline

unibz

Motivations for this research

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

1. Graph databases [Mendelzon and Wood 1995], aka 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 knowledge representation has provided important contributions to both settings.
- We combine the two aspects in a novel setting relying on constraints expressed in Description Logics (DLs) for managing evolving GSD.

Outline

1. Description Logics for Graph-structured Data
2. Reasoning in Dynamic Systems
3. Description Logics for Evolving Graph Structured Data
4. Planning
5. Conclusions

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
- Pointer structure in a program

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

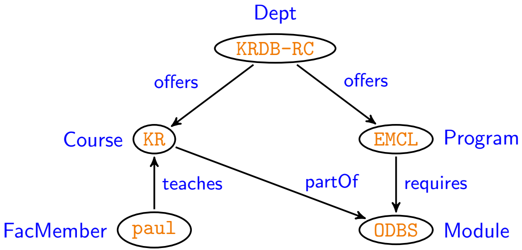
- **modeling languages and constraints**
- query languages
- efficient query answering
- **dealing with evolving GSD**

unibz

Graph-structured data is not really new

A graph-structured database instance =
An edge and node labeled graph =
A finite relational structure with unary and binary relations only =
A **finite** Description Logic 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** in $\mathcal{I}$ **by a path** whose sequence of labels is a word in the language **of** P.

- Path constraints φ come in two forms: $P_\ell \subseteq P_r$ $\quad$ $[P_p](P_\ell \subseteq P_r)$
- Semantics: **set of nodes** satisfying the constraint

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

$$[\![[P_p](P_\ell \subseteq P_r)]\!]^{\mathcal{I}} = \{n \mid \text{for all } n_1 \text{ s.t. } (n,n_1) \in [\![P_p]\!]^{\mathcal{I}}, \text{ if } (n_1,n') \in [\![P_\ell]\!]^{\mathcal{I}} \text{ then } (n_1,n') \in [\![P_r]\!]^{\mathcal{I}}, \text{ for all } n'\}$$

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 $a \in [\![\varphi]\!]^{\mathcal{I}}$, where a is some given node.

Central problem: **implication** of path constraints

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

- **Unrestricted implication:** Does $\Gamma \models \varphi$?
  I.e., for every $\mathcal{I}$, whenever $\mathcal{I} \models \Gamma$ then also $\mathcal{I} \models \varphi$.
- **Finite implication:** Does $\Gamma \models_{\mathit{fin}} \varphi$?
  Same as above, but over **finite** instances.

Similarly for unrestricted and finite implication under pointed semantics.

Implication of path constraint is undecidable

Finite and unrestricted **implication** of path constraints was shown **undecidable** [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 $w_\ell \subseteq w_r$

$\leadsto$ 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 simple labels (i.e., no ε, no inverse labels).

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:

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

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 (positive) regular XPath with nominals [Cate and Segoufin 2008; C., De Giacomo, Lenzerini, et al. 2009] and Propositional Dynamic Logic [Fischer and Ladner 1979].

The DL $\mathcal{ZOI}$

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

  Note: each node 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 $r \in \mathsf{N_R}, \quad a, b \in \mathsf{N_I}, \quad S$: simple role

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

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:

$$\mathcal{K} \quad\longrightarrow\quad \alpha \mid \mathcal{K} \land \mathcal{K}' \mid \mathcal{K} \lor \mathcal{K}' \mid \dot{\neg}\mathcal{K}$$

Semantics of $\mathcal{ZOI}$

We have the standard DL semantics, based on (finite) FO interpretations.

- Roles (i.e., paths) are interpreted as binary relations:
  - Regular expressions are analogous to those in path constraints.
  - Inverse role r^- denotes the inverse of the binary relation denoted by r.
  - Also:
    $$\begin{array}{rcl} \varepsilon^{\mathcal{I}} & = & \{(o,o) \mid o \in \Delta^{\mathcal{I}}\} \\ \nabla^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\ \{(a,b)\}^{\mathcal{I}} & = & \{(a^{\mathcal{I}}, b^{\mathcal{I}})\} \end{array}$$
- Concepts are interpreted as unary relations (i.e., sets of objects):
  - The boolean operators $\sqcap$, $\sqcup$, $\neg$ are as usual.
  - Nominal $\{a\}$ denotes a singleton, i.e., $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$.
  - $\exists P.C$ denotes the starting points of a P-path ending in (an instance of) C.
  - $\forall P.C$ denotes the objects for which all P-paths starting there end in C.
- Inclusions are interpreted as implications (i.e., as set inclusion):
  - $\mathcal{I}$ satisfies $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - Analogously for roles.
- Assertions $C(a)$ and $S(a,b)$ are analogous to facts, but can make use of complex concept and role expressions.
- Booleans in a KB have the usual meaning.

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 $\mathcal{ZOI}$

We can encode path constraints in $\mathcal{ZOI}$, for **empty prefixes** under **pointed semantics** (for individual a):

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

(where $\mathsf{inv}(P_r)$ denotes the role representing the inverse of path P_r)

Lemma

Let Γ be a set of prefix-empty constraints, φ a prefix-empty constraint, and a an individual. Then:

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

unibz

Complexity of path-constraint implication

Satisfiability of $\mathcal{ZOI}$ is ExpTime-**complete**. From this we get:

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

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.

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

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

unibz

Other classes of path constraints

For empty prefixes under pointed semantics, we have used a nominal to encode the inclusion of the left-tail in the right-tail. This does **not** work

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

To express other path constraints, we need to extend the logic:

We can capture all forms of path constraints in $\mathcal{ZOI}$ extended with **role difference** for arbitrary (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 an individual. Then:

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

$\Gamma \models_{(\mathit{fin})} \varphi$ iff $\dot\neg\big(\bigsqcap_{\gamma\in\Gamma} C_\gamma \sqsubseteq C_\varphi\big)$ is not (finitely) satisfiable

Outline

unibz

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

unibz

Reasoning about evolving data and knowledge

However, the knowledge representation community 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]
- Knowledge and Action Bases [Bagheri Hariri, C., Montali, et al. 2013]
- Bounded Situation Calculus [De Giacomo, Lesperance, and Patrizi 2012; C., De Giacomo, Montali, et al. 2016]

Reasoning about evolving data

Actually, there is quite some work also coming from the database community:

- Dynamic relational model [Vianu 1983, 1984]
- Transactional database schemas [Abiteboul and Vianu 1985, 1986, 1987, 1988]
- Temporal deductive databases [Snodgrass 1984; Chomicki and Imielinski 1988]
- Relational and ASM transducers [Abiteboul, Vianu, et al. 1998; Spielmann 2000]
- Data-driven web systems [Deutsch, Sui, and Vianu 2004]
- Business Artifacts [Nigam and Caswell 2003; Bhattacharya et al. 2007]
- Active XML [Abiteboul, Benjelloun, and Milo 2004]
- Artifact systems with arithmetic [Damaggio, Deutsch, and Vianu 2012]
- Data Centric Dynamic Systems [Bagheri Hariri, C., De Giacomo, et al. 2013]

unibz

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 intensional knowledge about the system is fixed vs. changes.

The setting we adopt here

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

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

unibz

Reasoning services of interest

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

- **Verification.**
- Variants of planning:
  - Existence of a plan.
  - Existence of a plan from a given precondition.
  - Conformant planning.
- Variants of bounded planning, i.e., we impose a priori finites bounds on the length or domain of the plan.

unibz

Reasoning services – Verification

Applying an action to a finite DB instance

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

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

Verification (V) problem

Given KB $\mathcal{K}$ and action α, **is α $\mathcal{K}$-preserving?**

I.e., is it the case that, for every finite DB instance $\mathcal{I}$, if $\mathcal{I} \models \mathcal{K}$ then $\alpha(\mathcal{I}) \models \mathcal{K}$?

Outline

Update language for GSD

To make the framework concrete, we consider an update language for GSD that allows for various types of actions:

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

unibz

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 in the conjunction fails), it does nothing.

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:

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

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

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

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

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

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

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 DB instance $\mathcal{I}$:

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

Theorem

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

α is $\mathcal{K}$-preserving
iff
$\mathcal{K} \land \dot{\neg}\mathsf{TR}(\mathcal{K}, \alpha_g)$ 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 $\mathrm{TR}(\mathcal{K}, \alpha_g)$ 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 $(r \oplus P)$ and $(r \ominus P)$, the role P **must be simple**: i.e., a 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:

friendOf $\ominus$ (hasAunt $\cap$ sendsCandyCrushInvitation$^-$)

friendOf $\ominus$ (supports$|_{\{\mathit{Berlusconi}\}}$)

preferredAICollaborators $\oplus$ $\exists(\text{collaboratesWith}|_{\neg\exists\text{projWith}.\{\mathit{Darpa}\}})^*$.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 $(\mathcal{K} \land (A \sqsubseteq \neg A') \land (A(o)))$-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.

The results relies on the undecidability of implication of path constraints of the simple form seen before.

Lightweight DLs

To simplify verification, we consider a restricted setting based on lightweight DLs of the *DL-Lite* family.

- *DL-Lite* is a family of DLs introduced for the purpose of accessing data through ontologies.
- This family provides a good foundation for ontology-based data access.

Standard *DL-Lite*$_{\mathcal{R}}$ KBs

- Roles P are names r or inverse roles r^-.
- Concepts B are: names A, or
  the projection $\exists P$ of role P on the first component, or
  the projection $\exists P^-$ of role P on the second component.
- In a KB, we can state: inclusions between concepts and roles,
  disjointness between concepts and roles
  ABox assertions $B(a)$ and $P(a, b)$.

A restricted setting based on *DL-Lite*

We consider a generalization of $\mathit{DL\text{-}Lite}_{\mathcal{R}}$ KBs:

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

- Concept and role inclusions and disjointness are as in standard $\mathit{DL\text{-}Lite}_{\mathcal{R}}$.
- In concept assertions $C(a)$, the concept C might be a boolean combination of concept names A, projections $\exists P$, and nominals $\{a'\}$.
- $\dot{\neg}$ may occur only in front of ABox assertions (while $\wedge$ and $\vee$ may be applied freely on KBs).

We need to restrict also the form of actions:

Localized actions

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

unibz

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

Theorem

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

Intuition:

1. Construct as before $\mathcal{K}' = \mathcal{K} \wedge \dot{\neg}\mathsf{TR}(\mathcal{K}, \alpha_g)$.
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 $(B_1 \sqcap \neg B_2)(o)$, where o is fresh,
   and each $\dot{\neg}(r_1 \sqsubseteq r_2)$ by $(r_1 \setminus r_2)(o, o')$, where o, o' are fresh.

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

unibz

Complexity of verification in the *DL-Lite* setting

Theorem

Finite satisfiability of *DL-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 *DL-Lite*$_{\mathcal{R}}^{+}$ KB $\mathcal{K}$ into a *DL-Lite*$_{\mathcal{R}}$ KB $\mathcal{K}'$, s.t., $\mathcal{K}$ is satisfiable iff there exists a $\mathcal{K}'$ that is satisfiable.

Theorem

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

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: $(A_0 \sqsubseteq \neg A'_0) \land \cdots \land (A_n \sqsubseteq \neg A'_n)$, 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.

unibz

Outline

unibz

Reasoning services – Planning

We are given:

- a KB $\mathcal{K}$,
- a finite DB instance $\mathcal{I}$ for $\mathcal{K}$, and
- a finite set *Act* 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 of objects such that

$$(\alpha_1 \circ \cdots \circ \alpha_n)(\mathcal{I}') \models \mathcal{K},$$

where $\mathcal{I}'$ is identical to $\mathcal{I}$, except that the domain is extended by Δ.

Note: extending the DB domain, account for new objects that might be needed in the plan to satisfy the goal constraints in $\mathcal{K}$.

Planning (P) problem and **Domain Bounded Planning (PDb)** problem

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

Reasoning services – Planning under incompleteness

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

Planning Under Incompleteness (PI) and
Length Bounded Planning Under Incompleteness (PILb)

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

Undecidability of unbounded planning

Without bounds, planning is undecidable already for the restricted *DL-Lite*$^{+}_{\mathcal{R}}$ setting.

Theorem (Undecidability of Planning)

Planning (P) and Planning Under Incompleteness (PI) are undecidable already for *DL-Lite*$^{+}_{\mathcal{R}}$ KBs and simple actions.

Intuition: we do not have a bound on the number of objects to be added to the domain to satisfy the goal KB.

unibz

Decidability of bounded planning

Planning under complete information becomes decidable if we bound the domain.

Theorem (Decidability of Domain Bounded Planning)

Domain Bounded Planning (PDb) is PSPACE-complete for $\mathcal{ZOI}$ KBs

Interestingly Planning Under Incompleteness stays undecidable even if we bound the domain.

However, it becomes decidable by bounding the plan length.

Theorem

Length Bounded Planning Under Incompleteness (PILb) is

- EXPTIME-complete for $\mathcal{ZOI}$ KBs, and
- NP-complete for *DL-Lite*$^{+}_{\mathcal{R}}$ KBs and with simple actions.

Outline

Summing up

- By exploiting techniques and tools coming from work in DLs, we obtain strong **decidability and complexity** results for reasoning about **evolving GSD under constraints**.
- This indicates that DLs are well suited not only to manage the structure of data, but also its dynamics.
- This calls for more interaction between the data management and knowledge representation communities.

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] Alberto O. Mendelzon and Peter T. Wood. "Finding Regular Simple Paths in Graph Databases". In: *SIAM J. on Computing* 24.6 (1995), pp. 1235–1258.

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

[3] 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.

[4] 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

[5] 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.

[6] Balder ten Cate and Luc Segoufin. "XPath, Transitive Closure Logic, and Nested Tree Walking Automata". In: *Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 2008, pp. 251–260.

[7] Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, et al. "An Automata-Theoretic Approach to Regular XPath". In: *Proc. of the 12th Int. Symp. on Database Programming Languages (DBPL)*. Vol. 5708. Lecture Notes in Computer Science. Springer, 2009, pp. 18–35.

[8] Michael J. Fischer and Richard E. Ladner. "Propositional Dynamic Logic of Regular Programs". In: *J. of Computer and System Sciences* 18 (1979), pp. 194–211.

References III

[9] 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.

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

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

[12] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning – Theory and Practice*. Elsevier, 2004. ISBN: 9781558608566.

[13] 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.

References IV

[14] 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: 9783642408847. DOI: 10.1007/978-3-642-40885-4_12. URL: http://dx.doi.org/10.1007/978-3-642-40885-4_12.

[15] 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.

[16] 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.

[17] Diego C., Giuseppe De Giacomo, Marco Montali, et al. "On First-Order μ-Calculus over Situation Calculus Action Theories". In: *Proc. of the 15th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press, 2016, pp. 411–420.

References V

[18] Victor Vianu. "Dynamic Constraints and Database Evolution". In: *Proc. of the 2nd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS)*. 1983, pp. 389–399. DOI: 10.1145/588058.588105.

[19] Victor Vianu. "Object Projection Views in the Dynamic Relational Model". In: *Proc. of the 3rd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS)*. 1984, pp. 214–220. DOI: 10.1145/588011.588042.

[20] Serge Abiteboul and Victor Vianu. "Transactions and Integrity Constraints". In: *Proc. of the 4th ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS)*. 1985, pp. 193–204. DOI: 10.1145/325405.325439.

[21] Serge Abiteboul and Victor Vianu. "Deciding Properties of Transactional Schemas". In: *Proc. of the 5th ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS)*. 1986, pp. 235–239. DOI: 10.1145/6012.15417.

References VI

[22] Serge Abiteboul and Victor Vianu. "A Transcation Language Complete for Database Update and Specification". In: *Proc. of the 6th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 1987, pp. 260–268. DOI: 10.1145/28659.28688.

[23] Serge Abiteboul and Victor Vianu. "Procedural and Declarative Database Update Languages". In: *Proc. of the 7th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 1988, pp. 240–250. DOI: 10.1145/308386.308448.

[24] Richard T. Snodgrass. "The Temporal Query Language TQuel". In: *Proc. of the 3rd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS)*. 1984, pp. 204–213. DOI: 10.1145/588011.588041.

[25] Jan Chomicki and Tomasz Imielinski. "Temporal Deductive Databases and Infinite Objects". In: *Proc. of the 7th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 1988, pp. 61–73. DOI: 10.1145/308386.308416.

unibz

References VII

[26] Serge Abiteboul, Victor Vianu, et al. "Relational Transducers for Electronic Commerce". In: *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 1998, pp. 179–187.

[27] Marc Spielmann. "Verification of Relational Transducers for Electronic Commerce". In: *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 2000, pp. 92–103.

[28] Alin Deutsch, Liying Sui, and Victor Vianu. "Specification and Verification of Data-driven Web Services". In: *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 2004, pp. 71–82.

[29] A. Nigam and N. S. Caswell. "Business Artifacts: An approach to Operational Specification". In: *IBM Systems J.* 42.3 (2003), pp. 428–445.

References VIII

[30] K. Bhattacharya et al. "Towards Formal Analysis of Artifact-Centric Business Process Models". In: *Proc. of the 5th Int. Conf. on Business Process Management (BPM)*. Vol. 4714. Lecture Notes in Computer Science. Springer, 2007, pp. 288–234.

[31] Serge Abiteboul, Omar Benjelloun, and Tova Milo. "Positive Active XML". In: *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*. 2004, pp. 35–45. DOI: 10.1145/1055558.1055564.

[32] Elio Damaggio, Alin Deutsch, and Victor Vianu. "Artifact Systems with Data Dependencies and Arithmetic". In: *ACM Trans. on Database Systems* 37.3 (2012), p. 22. DOI: 10.1145/2338626.2338628.

[33] 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.

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

unibz