Planning Problems for Graph Structured Data in Description Logics

Shqiponja Ahmetaj[1], Diego Calvanese[2], Magdalena Ortiz[1], and Mantas Šimkus[1]

[1] Institute of Information Systems, Vienna University of Technology, Austria
[2] KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Italy

1 Motivation

The complex structure and increasing size of information that has to be managed in today's applications calls for flexible mechanisms for storing such information, making it easily and efficiently accessible, and facilitating its change and evolution over time. The paradigm of *graph structured data* (GSD) [5] has gained popularity recently as an alternative to traditional relational databases that provides more flexibility and thus can overcome the limitations of an a priori imposed rigid structure on the data. Indeed, differently from relational data, GSD do not require a schema to be fixed a priori. This flexibility makes them well suited for many emerging application areas such as managing Web data, information integration, persistent storage in object-oriented software development, or management of scientific data. Concrete examples of models for GSD are RDFS [2], object-oriented data models, and XML.

Here we build on recent work that advocates the use of Description Logics (DLs) for managing change in GSD that happens as the result of (agents or users) executing actions [4]. We consider GSD understood in a broad sense, as information represented by means of a node and edge labeled graph, in which the labels convey semantic information. We identify GSD with the *finite* structures over which DLs are interpreted, and use DL knowledge bases as descriptions of constraints and properties of the data. We express actions using a specially tailored action language in which actions are finite sequences of (possibly conditional) insertions and deletions performed on the extensions of labels. For this setting, the *static verification problem*, which consists on deciding whether the execution of a given action will *preserve* some given integrity constraints on any possible GSD, has been studied in [4]. Here we discuss further problems that can be considered as variants of planning, such as deciding whether there is a sequence of actions that leads a given structure into a state where some property (either desired or not) holds, or whether a given sequence of actions leads every structure into a state where some property necessarily holds. We develop algorithms for variations of these problems, and characterize their computational complexity. This abstract is based on the full paper [1].

2 Planning Problems for GSD

For manipulating GSD we use a specially tailored language [1], in which a basic action can take, for example, the form $(A \oplus C)$ for a concept name A and an arbitrary concept C. Intuitively, when this action is applied to an interpretation (i.e., a GSD instance) $\mathcal{I}$, the content of $C^{\mathcal{I}}$ is added to $A^{\mathcal{I}}$. Similarly $(A \ominus C)$ removes $C^{\mathcal{I}}$ from $A^{\mathcal{I}}$, and there are analogous operations $(p \oplus r)$ and $(p \ominus r)$ on roles. These basic actions β can be combined using action composition $\beta \cdot \alpha$ and conditional action execution $\mathcal{K}\,?\,\alpha_1 \parallel \alpha_2$. Actions may be *non-ground* and contain variables (in the place of individuals), which stand for parameters. The semantics is then given using substitutions in the natural way.

Given a set Act of actions, a finite interpretation $\mathcal{I}$, and a goal KB $\mathcal{K}$, a *plan* is a finite sequence of ground instances of actions from Act, whose execution leads from $\mathcal{I}$ to a state $\mathcal{I}'$ that satisfies $\mathcal{K}$. We allow to introduce fresh values in the data by expanding in $\mathcal{I}'$ the domain of $\mathcal{I}$ with a finite set of domain elements.

Example 1. The following interpretation $\mathcal{I}$ represents (part of) the project database of a research institute. There are two active projects and three employees working in them.

$$\begin{array}{rlll}\mathsf{Empl}^{\mathcal{I}} = \{e_1, e_3, e_7\} & \mathsf{ActivePrj}^{\mathcal{I}} = \{p_1, p_2\} & \mathsf{worksFor}^{\mathcal{I}} = \{(e_1, p_1), (e_3, p_1), (e_7, p_2)\} \\ \mathsf{Prj}^{\mathcal{I}} = \{p_1, p_2\} & \mathsf{FinishedPrj}^{\mathcal{I}} = \{\} & \end{array}$$

We assume constants $\mathsf{p_i}$ with $\mathsf{p_i}^{\mathcal{I}} = p_i$ for projects, and analogously constants $\mathsf{e_i}$ for employees. The KB $\mathcal{K}_1$ expresses constraints on this project database: all projects are active or finished, the domain of worksFor are the employees, and its range the projects.

$$(\mathsf{Prj} \sqsubseteq \mathsf{ActivePrj} \sqcup \mathsf{FinishedPrj}) \wedge (\exists \mathsf{worksFor}.\top \sqsubseteq \mathsf{Empl}) \wedge (\exists \mathsf{worksFor}^{-}.\top \sqsubseteq \mathsf{Prj})$$

The following *goal* KB requires that $\mathsf{p_1}$ is not an active project, and that $\mathsf{e_1}$ is an employee. Consider the following actions α_1 and α_2; here ε stands for the empty action. Action α_1 moves $\mathsf{p_1}$ from the active to the finished projects, and removes the employees that work only for $\mathsf{p_1}$. Action α_2 transfers an employee x from project $\mathsf{p_1}$ to project $\mathsf{p_2}$, if the necessary preliminary checks are successful.

$$\begin{array}{l}\mathcal{K}_g = \neg(\mathsf{p_1}{:}\mathsf{ActivePrj}) \wedge \mathsf{e_1}{:}\mathsf{Empl} \\ \alpha_1 = \mathsf{ActivePrj} \ominus \{\mathsf{p_1}\} \cdot \mathsf{FinishedPrj} \oplus \{\mathsf{p_1}\} \cdot \mathsf{Empl} \ominus \forall \mathsf{worksFor}.\{\mathsf{p_1}\} \cdot \mathsf{worksFor} \ominus \mathsf{worksFor}|_{\{\mathsf{p_1}\}} \\ \alpha_2 = (\mathsf{p_2}{:}\mathsf{Prj} \wedge (x, \mathsf{p_1}){:}\mathsf{worksFor})\ ?\ \big(\mathsf{worksFor} \ominus \{(x, \mathsf{p_1})\} \cdot \mathsf{worksFor} \oplus \{(x, \mathsf{p_2})\}\big) \parallel \varepsilon\end{array}$$

The sequence $\langle \alpha_2', \alpha_1 \rangle$ is a plan for $\mathcal{K}_g$ from $\mathcal{I}$, where α_2' is the result of applying to α_2 the substitution $\sigma : \{x \mapsto \mathsf{e_1}\}$, that is, parameter x takes the value $\mathsf{e_1}$. The interpretation $\mathcal{I}'$ that reflects the resulting status of the data looks as follows (note that $\mathcal{I}' \models \mathcal{K}_1 \wedge \mathcal{K}_g$):

$$\begin{array}{rlll}\mathsf{Empl}^{\mathcal{I}} = \{e_1, e_7\} & \mathsf{ActivePrj}^{\mathcal{I}} = \{p_2\} & \mathsf{worksFor}^{\mathcal{I}} = \{(e_1, p_2), (e_7, p_2)\} \\ \mathsf{Prj}^{\mathcal{I}} = \{p_1, p_2\} & \mathsf{FinishedPrj}^{\mathcal{I}} = \{p_1\} & \end{array}$$

We define the following planning problems:

(P1) Given a set Act of actions, a finite interpretation $\mathcal{I}$, and a goal KB $\mathcal{K}$, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$?

(P2) Given a set Act of actions and a pair $\mathcal{K}_{pre}$, $\mathcal{K}$ of formulae, does there exist a substitution σ and a plan for $\sigma(\mathcal{K})$ from some finite $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$?

(P1) is the classic plan existence problem, formulated in the setting of GSD. (P2) also aims at deciding plan existence, but rather than the full actual state of the data, we have as an input a *precondition* KB, and we are interested in deciding the existence of a plan from some of its models. To see the relevance of (P2), consider the complementary problem: a 'no' instance of (P2) means that, from every relevant initial state, (undesired) goals cannot be reached. For instance, $\mathcal{K}_{pre} = \mathcal{K}_{ic} \wedge x : \mathsf{FinishedPrj}$ and $\mathcal{K} = x : \mathsf{ActivePrj}$

may be used to check whether starting with GSD that satisfies the integrity constraints and contains some finished project p, it is possible to make p an active project again.

Unfortunately, these problems are undecidable already for *DL-Lite* KBs and a quite restricted form of actions. For (P1), the intuition behind this is that we do not know how many fresh objects we need to add to the domain of $\mathcal{I}$. If we put a bound on the number of these fresh objects, we regain decidability. (P2) remains undecidable even if the domain is fixed, but it becomes decidable if we place a bound on the length of plans. In what follows, we assume that the integers k given as bounds are encoded in unary.

(P1$_\mathrm{b}$) Given a set Act of actions, a finite interpretation $\mathcal{I}$, a goal KB $\mathcal{K}$, and a positive integer k, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$ such that at most k elements are added to the domain of $\mathcal{I}$?

(P2$_\mathrm{b}$) Given a set of actions Act, a pair $\mathcal{K}_{pre}, \mathcal{K}$ of formulae, and a positive integer k, does there exist a substitution σ and a plan of length at most k for $\sigma(\mathcal{K})$ from some finite interpretation $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$?

The problem (P1$_\mathrm{b}$) is PSPACE-hard already for settings more restricted than *DL-Lite*, and it can be solved in polynomial space even for the very expressive $\mathcal{ALCHOIQ}br$ (an extension of $\mathcal{ALCHOIQ}$ with further role constructors and Boolean KBs). Note that the problem is not harder than deciding plan existence in standard planning formalisms such as propositional STRIPS [3]. The problem (P2$_\mathrm{b}$) is NEXPTIME-complete for $\mathcal{ALCHOIQ}br$, and the complexity drops to NP-complete if we consider *DL-Lite* and suitably restricted forms of actions.

Next we consider problems that are related to ensuring that plans *always* achieve a goal $\mathcal{K}$, given a possibly incomplete description $\mathcal{K}_{pre}$ of the initial data. They are variants of the so-called *conformant* planning, which deals with incomplete information. The first such problem is to 'certify' that a candidate plan is always a plan for the goal.

(C) Given a sequence P of actions and formulae $\mathcal{K}_{pre}, \mathcal{K}$, is $\sigma(P)$ a plan for $\sigma(\mathcal{K})$ from every finite interpretation $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every substitution σ?

Finally, we are interested in deciding the existence of a plan that always achieves the goal, for every possible state satisfying the precondition. Solving this problem corresponds to the automated *synthesis* of a program for reaching a certain condition. This is formalized via the following problems:

(S) Given a set Act of actions and formulae $\mathcal{K}_{pre}, \mathcal{K}$, does there exist a sequence P of actions from Act such that $\sigma(P)$ is a plan for $\sigma(\mathcal{K})$ from every finite $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every substitution σ?

(S$_\mathrm{b}$) Given a set Act of actions, formulae $\mathcal{K}_{pre}, \mathcal{K}$, and a positive integer k, does there exist a sequence P of actions from Act such that $\sigma(P)$ is a plan for $\sigma(\mathcal{K})$ of length at most k, from every finite $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every substitution σ?

Problem (S) is undecidable already for *DL-Lite*. For $\mathcal{ALCHOIQ}br$, (C) and (S$_\mathrm{b}$) are complete for coNEXPTIME. For *DL-Lite*, (C) is complete for coNP and (S$_\mathrm{b}$) for $\mathrm{NP}^{\mathrm{NP}}$.

3 Conclusions

We believe this work provides powerful tools for analyzing the effects of executing complex actions on GSD in the presence of integrity constraints expressed in DLs. Interesting lines for further research are developing practicable algorithms and identifying meaningful restricted fragments of lower complexity, in particular tractable fragments.

References

1. S. Ahmetaj, D. Calvanese, M. Ortiz, and M. Šimkus. Managing change in graph-structured data using description logics. In *Proc. of AAAI*, 2014. Long version with proofs available at `http://arxiv.org/abs/1404.4274`.
2. D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, W3C, Feb. 2004. `http://www.w3.org/TR/rdf-schema/`.
3. T. Bylander. The computational complexity of propositional STRIPS planning. *AIJ*, 69:165–204, 1994.
4. D. Calvanese, M. Ortiz, and M. Simkus. Evolving graph databases under description logic constraints. In *Proc. of DL*, volume 1014 of *CEUR,* `ceur-ws.org`, pages 120–131, 2013.
5. S. Sakr and E. Pardede, editors. *Graph Data Management: Techniques and Applications*. IGI Global, 2011.