Logic on MARS: Ontologies for Generalised Property Graphs (Extended Technical Report)

Maximilian Marx and **Markus Krötzsch** and **Veronika Thost**
TU Dresden
Center for Advancing Electronics Dresden (cfaed)
{maximilian.marx,markus.kroetzsch,veronika.thost}@tu-dresden

Abstract

Graph-structured data is used to represent large information collections, called knowledge graphs, in many applications. Their exact format may vary, but they often share the concept that edges can be annotated with additional information, such as validity time or provenance information. Property Graph is a popular graph database format that also provides this feature. We give a formalisation of a generalised notion of Property Graphs, called *multi-attributed relational structures* (MARS), and introduce a matching knowledge representation formalism, *multi-attributed predicate logic* (MAPL). We analyse the expressive power of MAPL and suggest a simpler, rule-based fragment of MAPL that can be used for ontological reasoning on Property Graphs. To the best of our knowledge, this is the first approach to making Property Graphs and related data structures accessible to symbolic AI.

1 Introduction

The representation of human knowledge in its variety and complexity is a central topic of AI. Recently, graph-based representation formats have led to huge advances in the field of data modelling. So-called *Knowledge Graphs* (KGs) are now widely used to capture heterogeneous, multi-faceted knowledge from different sources in a coherent way. Large KGs are deployed in companies such as Google and Facebook, in online communities such as Freebase [Bollacker *et al.*, 2008] and Wikidata [Vrandečić and Krötzsch, 2014], and in research projects such as YAGO2 [Hoffart *et al.*, 2013].

There is no standard definition of what constitutes a KG, and the formats used in practice vary. The basis of KGs typically are directed graphs with labelled nodes and edges. What distinguishes them from plain graphs is their enriched structure that includes additional *annotations* to provide contextual information for every edge or node. Examples include provenance (source information) and temporal validity, but there can be many other types of annotations.

A popular data model for such KGs is the *Property Graph* model, used by the Neo4J graph database [Rodriguez and Neubauer, 2010] and various other database applications [Rudolf *et al.*, 2013]. It allows sets of attribute–value pairs to be associated with the nodes and edges in a directed graph. For example, we can express that Liz Taylor married Richard Burton 1964 in Montreal, and that the marriage ended in 1974 by annotating the fact (or *edge*) $\mathsf{spouse}(\mathsf{taylor}, \mathsf{burton})$ with the annotation $\{\mathsf{start} : 1964, \mathsf{loc} : \mathsf{Montreal}, \mathsf{end} : 1974\}$. We write the annotated fact as $\mathsf{spouse}(\mathsf{taylor}, \mathsf{burton})@\{\mathsf{start} : 1964, \mathsf{loc} : \mathsf{Montreal}, \mathsf{end} : 1974\}$. Such graphs are also known as *attributed graphs*. This model is fundamentally different from annotated logics, e.g., using semiring annotations [Green *et al.*, 2007; Straccia *et al.*, 2010] or requiring functional annotations like aRDF [Udrea *et al.*, 2010], since the same edge may occur with several, distinct annotations (e.g., Taylor was also married to Burton from 1975 to 1976). Wikidata's custom data model supports attributed statements too, but goes beyond Property Graph by allowing attributes with multiple values. Wikidata includes, e.g., the fact $\mathsf{castMember}(\mathsf{Sesame\_Street}, \mathsf{Frank\_Oz})$ with annotation $\{\mathsf{role} : \mathsf{Cookie\_Monster}, \mathsf{role} : \mathsf{Grover}, \mathsf{role} : \mathsf{Bert}\}$.[1] We call such generalised Property Graphs *multi-attributed graphs*.

In spite of the huge practical significance of these data models and the obvious potential of KGs for AI, there is practically no support for using such data in knowledge representation. The popular ontology language OWL focusses on labelled directed graphs as defined by RDF [OWL Working Group, 2009]. Rule languages as in logic programming support predicates of higher arities, but bounded arity relations cannot encode arbitrary sets of attribute–value pairs either. Indeed, we might want to express schema knowledge as sketched by the following rule:

$$\begin{aligned}&\forall x, y, z_1, z_2, z_3.\\ &\qquad \mathsf{spouse}(x, y)@\{\mathsf{start} : z_1, \mathsf{loc} : z_2, \mathsf{end} : z_3\} \\ &\qquad \rightarrow \mathsf{spouse}(y, x)@\{\mathsf{start} : z_1, \mathsf{loc} : z_2, \mathsf{end} : z_3\}\end{aligned} \tag{1}$$

which states that spouse is a symmetric relation, where the inverse statement has the same start and end dates, and location. To the best of our knowledge, this cannot be encoded in any modern ontology language. Yet there are many more complex relationships that one might want to express. In fact, even the underlying data model of a multi-attributed graph lacks proper formalisation. Property Graph and Wikidata are highly implementation bound and have no formal specification. Other approaches, e.g., YAGO2 annotated triples, lack generality.

[1] https://www.wikidata.org/wiki/Q155629 (retr. 2017-02-17)

In this paper, we therefore propose a new, formal data model for generalised Property Graphs, called *multi-attributed relational structure* (MARS), and we develop a logical language, called *multi-attributed predicate logic* (MAPL), for modelling knowledge over such structures. Our main contributions are as follows:

- we provide a logical foundation for knowledge modelling over Property Graphs,
- we study the expressive power of MAPL and show that it has no sound and complete deduction calculus,
- we identify two expressive features of MAPL that are useful for modelling,
- we suggest a rule-based fragment of MAPL, called *MARPL*, for which reasoning is decidable, and
- we give a reasoning procedure for MARPL.

For reasoning in MARPL, we obtain an ExpTime complexity bound, which we show to be tight even if only the underlying facts (corresponding to the input KG) are allowed to vary. This unusually high data complexity illustrates the added power of our approach as compared to traditional rule languages such as Datalog. Under a reasonable assumption (bounding the size of annotations), data complexity drops back to polynomial time.

For now, we treat the annotations as opaque; we do not aim at providing specialty reasoning services for certain datatypes such as time, numbers, coordinates, or provenance information. As each datatype suggests different bodies of related work, we refrain from going into detail.

Our presentation is self-contained, but some proofs and standard definitions have been moved to the appendix of this extended report.

2 Multi-Attributed Relational Structures

We now formalise our notion of a generalised Property Graph, and we use it as the basis for the model theory of a suitably enhanced predicate logic.

We consider a finite set $\mathbf{P}$ of *predicates*, where each $p \in \mathbf{P}$ has an associated arity $\mathsf{ar}(p) \geq 0$. If not otherwise stated, we assume this signature to be fixed and refrain from mentioning it. For the following definition, let $\mathcal{P}_{\mathsf{fin}}(S)$ be the set of all finite subsets of set S.

Definition 1. *A* multi-attributed relational structure *(MARS) $\mathcal{M}$ consists of a non-empty set $\Delta^{\mathcal{M}}$ of* domain elements *and, for each n-ary predicate $p \in \mathbf{P}$, an $(n+1)$-ary relation $p^{\mathcal{M}} \subseteq (\Delta^{\mathcal{M}})^n \times \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{M}} \times \Delta^{\mathcal{M}}\right)$.*

In other words, a MARS behaves like a relational structure (i.e., hypergraph) over a domain $\Delta^{\mathcal{M}}$, where each relation tuple (i.e., hyperedge) is annotated with a finite binary relation over $\Delta^{\mathcal{M}}$. We view this relation as a set of attribute–value pairs. There might be multiple values for each attribute, justifying our terminology. Also note that the same relational tuple may occur with different attribute–value collections within a single MARS. Thus MARS generalise Property Graphs, where attributes are functional and relations are unary and binary. The unary relations can be used to assign attribute–value collections to nodes.

To represent schematic knowledge on (generalised) Property Graphs, we introduce logical formulae with an additional set-valued parameter for each predicate, representing attribute–value pairs. In addition to the set of predicates $\mathbf{P}$, such logical expressions are based on a set $\mathbf{C}$ of *constant symbols*, a set $\mathbf{V}$ of *object variables* and a set $\mathbf{U}$ of *set variables*.

Definition 2. *An* object term *is a symbol $t \in \mathbf{C} \cup \mathbf{V}$. A* set term *is either a variable $U \in \mathbf{U}$ or an expression of the form $\{a_1 : b_1, \ldots, a_n : b_n\}$ for $n \geq 0$, where a_i, b_i are object terms for all $i \in \{1, \ldots, n\}$.*

The formulae of multi-attributed predicate logic *(MAPL) are defined as follows. A* relational atom *is an expression $p(t_1, \ldots, t_n)@S$ where $p \in \mathbf{P}$ is an n-ary predicate, $t_1, \ldots, t_n$ are object terms, and S is a set term. A* set atom *is an expression $(a : b) \in S$, where a, b are object terms and S is a set term. An* atom *is a relational atom or set atom. The set of MAPL formulae is inductively defined to contain all atoms and all formulae of the form $\varphi_1 \wedge \varphi_2$, $\neg\varphi$, $\exists x.\varphi$ and $\exists U.\varphi$, where $\varphi_{(i)}$ are MAPL formulae, $x \in \mathbf{V}$ and $U \in \mathbf{U}$. We define the shortcuts $\vee$, $\rightarrow$, and $\forall$ in the usual way.*

Example 1. (1) *is a MAPL formula that works on* spouse *statements with* start, loc, *and* end *attributes. It does not apply to marriages missing* loc, *but the location may not be known for all* spouse *statements. To support* optional *attributes, we first define a shortcut $\varphi[X, Y]$, where X and Y might be arbitrary set variables, to be the following formula:*

$$\begin{aligned} &\forall z_1.((\mathsf{start} : z_1) \in X \rightarrow (\mathsf{start} : z_1) \in Y) \wedge \\ &\forall z_2.((\mathsf{loc} : z_2) \in X \rightarrow (\mathsf{loc} : z_2) \in Y) \wedge \\ &\forall z_3.((\mathsf{end} : z_3) \in X \rightarrow (\mathsf{end} : z_3) \in Y) \end{aligned} \tag{2}$$

The formula $\Psi[X, Y] = \varphi[X, Y] \wedge \forall W.\forall x, y.(\varphi[X, W] \wedge (x : y) \in Y) \rightarrow (x : y) \in W$ then expresses that Y is the least set that contains the same start, loc, *and* end *values as X, i.e., Y does not contain any other attributes. Using this conditional copying of attributes, we can improve* (1)*:*

$$\begin{aligned} &\forall x, y.\forall U, V.(\mathsf{spouse}(x, y)@U \wedge \psi[U, V] \\ &\qquad \rightarrow \mathsf{spouse}(y, x)@V) \end{aligned} \tag{3}$$

The semantics of MAPL formulae is defined in the natural way by using MARS as the basis for the model theory, i.e., by switching from standard relational structures to MARS in logical interpretations. We first define our interpretation of (object and set) terms. We make a *unique name assumption* (UNA), since this is more natural for working with large knowledge graphs (which are similar to databases). The UNA is particularly useful when formulating meaningful conditions on finite annotations, as discussed in Section 3.

Definition 3. *A MAPL* interpretation *$\mathcal{I}$ is a MARS, which in addition maps each constant $c \in \mathbf{C}$ injectively to an element $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ (UNA). A MAPL* variable assignment *for $\mathcal{I}$ is a function $\mathcal{Z}$ that maps object variables $x \in \mathbf{V}$ to elements $\mathcal{Z}(x) \in \Delta^{\mathcal{I}}$ and set variables $U \in \mathbf{U}$ to finite binary relations $\mathcal{Z}(U) \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$.*

For an object term t, we define $t^{\mathcal{I},\mathcal{Z}} := t^{\mathcal{I}}$ if $t \in \mathbf{C}$ and $t^{\mathcal{I},\mathcal{Z}} := \mathcal{Z}(t)$ if $t \in \mathbf{V}$. For a set variable $U \in \mathbf{U}$, we define $U^{\mathcal{I},\mathcal{Z}} := \mathcal{Z}(U)$. For a set term $S = \{a_1 : b_1, \ldots, a_n : b_n\}$, we define $S^{\mathcal{I},\mathcal{Z}} := \{\langle a_1^{\mathcal{I},\mathcal{Z}}, b_1^{\mathcal{I},\mathcal{Z}}\rangle, \ldots, \langle a_n^{\mathcal{I},\mathcal{Z}}, b_n^{\mathcal{I},\mathcal{Z}}\rangle\}$.

Definition 4. *An interpretation $\mathcal{I}$ and a variable assignment $\mathcal{Z}$ for $\mathcal{I}$ may* satisfy *a formula φ, written $\mathcal{I}, \mathcal{Z} \models \varphi$. This is defined recursively as follows:*

- $\mathcal{I}, \mathcal{Z} \models p(t_1, \dots, t_n)@S$ *if* $\langle t_1^{\mathcal{I},\mathcal{Z}}, \dots, t_n^{\mathcal{I},\mathcal{Z}}, S^{\mathcal{I},\mathcal{Z}} \rangle \in p^{\mathcal{I}}$,
- $\mathcal{I}, \mathcal{Z} \models (a : b) \in S$ *if* $\langle a^{\mathcal{I},\mathcal{Z}}, b^{\mathcal{I},\mathcal{Z}} \rangle \in S^{\mathcal{I},\mathcal{Z}}$,
- $\mathcal{I}, \mathcal{Z} \models (\varphi_1 \wedge \varphi_2)$ *if* $\mathcal{I}, \mathcal{Z} \models \varphi_1$ *and* $\mathcal{I}, \mathcal{Z} \models \varphi_2$,
- $\mathcal{I}, \mathcal{Z} \models \neg\varphi$ *if* $\mathcal{I}, \mathcal{Z} \not\models \varphi^{\mathcal{I}}$,
- $\mathcal{I}, \mathcal{Z} \models \exists x.\varphi$ *if there is* $\delta \in \Delta^{\mathcal{I}}$ *such that* $\mathcal{I}, \mathcal{Z}|_{x \mapsto \delta} \models \varphi$,
- $\mathcal{I}, \mathcal{Z} \models \exists U.\varphi$ *if there is* $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$ *such that* $\mathcal{I}, \mathcal{Z}|_{U \mapsto R} \models \varphi$,

*where $\mathcal{Z}|_{X \mapsto V}$ denotes the assignment $\mathcal{Z}$ modified to map X to V. We may omit $\mathcal{Z}$ for formulae without free variables (*sentences*), writing $\mathcal{I} \models \varphi$; in this case, $\mathcal{I}$ is a* model *of φ.*

As usual, we say that a variable-free MAPL formula is *satisfiable* if it has a model. A variable-free formula φ *entails* ψ, written $\varphi \models \psi$, if all models of φ are also models of ψ. We extend all of these notions to sets of formulae as usual.

Unfortunately, as the next theorem shows, MAPL is already too powerful for practical applications. Recall that *weak second-order logic* (WSO) is a second-order logic that allows set quantifiers to range over finite sets only [Leivant, 1994]. In spite of this restriction, there is no sound and complete calculus for WSO, even without built-in equality.

Theorem 1. *Satisfiability of a sentence of weak second-order logic can be polynomially reduced to satisfiability of a MAPL sentence. In particular, satisfiability for MAPL is not semidecidable (and therefore not decidable).*

Proof. MAPL uses the same connectives as WSO, but further restricts second-order quantification to binary predicate variables. We translate WSO formulae φ into equisatisfiable MAPL formulae φ_{MAPL}. For a k-ary second-order variable X, let $X_1, \dots, X_k$ be fresh set variables. If $\varphi = p(t_1, \dots, t_n)$ is a first-order atom, then $\varphi_{\mathsf{MAPL}} := p(t_1, \dots, t_n)@\{\}$. If $\varphi = X(t_1, \dots, t_k)$ is a second-order atom, then

$$\varphi_{\mathsf{MAPL}} := \exists v.(v : t_1) \in X_1 \wedge \dots \wedge (v : t_k) \in X_k.$$

If $\varphi = \exists X.\psi$ for a second-order variable X, then $\varphi_{\mathsf{MAPL}} := \exists X_1, \dots, X_k.\psi_{\mathsf{MAPL}}$. If $\varphi = \exists x.\psi$ for a first-order variable x, then $\varphi_{\mathsf{MAPL}} := \exists x.\psi_{\mathsf{MAPL}}$. Likewise, $(\neg\psi)_{\mathsf{MAPL}} := \neg\psi_{\mathsf{MAPL}}$ and $(\psi \wedge \psi')_{\mathsf{MAPL}} := \psi_{\mathsf{MAPL}} \wedge \psi'_{\mathsf{MAPL}}$. □

3 Towards a Decidable Fragment of MAPL

Theorem 1 shows that MAPL is hardly suitable as an ontology language, since automated, fully unsupervised processing can access logically entailed information only in incomplete ways. Our goal is to identify a decidable fragment of MAPL that is suitable for speaking about Property Graphs and related structures. In this section, we introduce several practically useful expressive features that MAPL can provide, and that will play an important role in the definition of this fragment.

An important class of formulae are those that can express conditions on attribute–value sets, which we call *specifiers*. For example, we will allow formulae such as $[\mathsf{start} : *, \mathsf{loc} : \mathsf{Montreal}](U)$ to state that U is an annotation that contains zero or more values for attribute start, the value $\mathsf{Montreal}$ for attribute loc, and no other attributes. We show that this and many other specifiers can be expressed in MAPL.

Definition 5. *A* closed specifier *is an expression of the form $[a_1 : b_1, \dots, a_n : b_n]$ where $n \geq 0$, a_i are object terms, and b_i are either object terms or one of the special symbols $*$ and $+$. For interpretation $\mathcal{I}$ and variable assignment $\mathcal{Z}$, define:*

- $[a : b]^{\mathcal{I},\mathcal{Z}} := \{\{a : b\}^{\mathcal{I},\mathcal{Z}}\}$ *if* $b \notin \{*, +\}$,
- $[a : *]^{\mathcal{I},\mathcal{Z}}$ *as the set of all finite binary relations of form* $\{\langle a^{\mathcal{I},\mathcal{Z}}, \delta_1 \rangle, \dots, \langle a^{\mathcal{I},\mathcal{Z}}, \delta_\ell \rangle\}$ *with* $\ell \geq 0$ *and* $\delta_i \in \Delta^{\mathcal{I}}$,
- $[a : +]^{\mathcal{I},\mathcal{Z}}$ *as the set of all finite binary relations of form* $\{\langle a^{\mathcal{I},\mathcal{Z}}, \delta_1 \rangle, \dots, \langle a^{\mathcal{I},\mathcal{Z}}, \delta_\ell \rangle\}$ *with* $\ell \geq 1$ *and* $\delta_i \in \Delta^{\mathcal{I}}$,
- $[a_1 : b_1, \dots, a_n : b_n]^{\mathcal{I},\mathcal{Z}} := \left\{\bigcup_{i=1}^{n} S_i \mid S_i \in [a_i : b_i]^{\mathcal{I},\mathcal{Z}}\right\}$.

Example 2. *Consider $\Omega := [\mathsf{start} : *, \mathsf{loc} : \mathsf{Montreal}]$, the closed specifier from the motivating example above. Then given any interpretation $\mathcal{I}$, we have $\{\mathsf{loc} : \mathsf{Montreal}\}^{\mathcal{I}} \in \Omega^{\mathcal{I}}$ and $\{\mathsf{start} : 1964, \mathsf{loc} : \mathsf{Montreal}\}^{\mathcal{I}} \in \Omega^{\mathcal{I}}$, but $\{\mathsf{start} : 1964, \mathsf{loc} : \mathsf{Montreal}, \mathsf{end} : 1974\}^{\mathcal{I}} \notin \Omega^{\mathcal{I}}$.*

Closed specifiers allow us to describe annotations that use only certain “expected” attributes, which is useful in the open, schema-less world of knowledge graphs, where attributes may be reused in different contexts. However, it is also useful to consider *open specifiers*, which characterise the set of all supersets of a given annotation:

Definition 6. *An* open specifier *is an expression of the form $\Omega = \lfloor a_1 : b_1, \dots, a_n : b_n \rfloor$ where $n \geq 0$, a_i are object terms, and b_i are either object terms or one of the special symbols $*$ and $+$. Given an interpretation $\mathcal{I}$ and a variable assignment $\mathcal{Z}$, we define $\Omega^{\mathcal{I},\mathcal{Z}}$ to be the set $\left\{S \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right) \mid S \supseteq T \text{ for some } T \in \bar{\Omega}^{\mathcal{I},\mathcal{Z}}\right\}$ where $\bar{\Omega}$ is the corresponding closed specifier $[a_1 : b_1, \dots, a_n : b_n]$.*

Example 3. *The open specifier $\lfloor \mathsf{start} : +, \mathsf{loc} : \mathsf{Montreal} \rfloor$ characterises annotations that have at least one value for* start *and a* loc *attribute with value Montreal, e.g., for any interpretation $\mathcal{I}$, we have $\{\mathsf{start} : 1964, \mathsf{loc} : \mathsf{Montreal}, \mathsf{end} : 1974\}^{\mathcal{I}} \in \lfloor \mathsf{start} : +, \mathsf{loc} : \mathsf{Montreal} \rfloor^{\mathcal{I}}$.*

It is natural to define set operations on specifiers. For example, the union $\lfloor \mathsf{loc} : \mathsf{Montreal} \rfloor \cup \lfloor \mathsf{loc} : \mathsf{Perth} \rfloor$ describes annotations with (at least) one of two possible values for loc.

Definition 7. *A* specifier Ω *is a term built from open and closed specifiers using the (binary) operators $\cap$, $\cup$, and $\setminus$. The semantics are extended in the natural way, e.g., $(\Omega_1 \cap \Omega_2)^{\mathcal{I},\mathcal{Z}} := \Omega_1^{\mathcal{I},\mathcal{Z}} \cap \Omega_2^{\mathcal{I},\mathcal{Z}}$. A* specifier atom *is an expression $\Omega(S)$, where Ω is a specifier and S is a set variable.*

Example 4. *Note that specifiers may use object variables. The following specifier defines annotations that give at least a start date x and a location y, but no end date:*

$$\lfloor \mathsf{start} : x, \mathsf{loc} : y \rfloor \setminus \lfloor \mathsf{end} : + \rfloor \tag{4}$$

We may treat arbitrary specifiers as syntactic sugar:

Theorem 2. *For every specifier Ω, there is a MAPL formula $\varphi_\Omega[U]$ with a set variable U such that, for all interpretations $\mathcal{I}$, variable assignments $\mathcal{Z}$, and relations $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$, we have $R \in \Omega^{\mathcal{I},\mathcal{Z}}$ iff $\mathcal{I}, \mathcal{Z}|_{U \mapsto R} \models \varphi_\Omega$.*

Example 5. *Along the lines of Example 4, we can define the current spouse of a person:*

$$\begin{aligned}&\forall x{,}y.\forall U.\big(\lfloor \mathsf{start} : + \rfloor \setminus \lfloor \mathsf{end} : + \rfloor\big)(U)\\ &\qquad \wedge \mathsf{spouse}(x,y)@U \rightarrow \mathsf{currentSpouse}(x,y)@U\end{aligned} \tag{5}$$

This is an important pattern that, in spite of the use of start and end times in our example, is fundamentally different from typical patterns in logics with temporal validity annotations.

Specifiers can express conditions on attribute–value sets. For ontological modelling, it is further useful to construct new attribute–value sets that are used to annotate newly derived facts. For instance, we have constructed a set V with optional parts in Example 1. We want enough power to define interesting types of functions, which compute new annotation sets by evaluating conditions on their input.

Definition 8. *Let* $\mathbf{F}$ *denote a set of* function symbols. *A* function definition $\langle F, \boldsymbol{S}, C\rangle$ *consists of a symbol* $F \in \mathbf{F}$, *a* signature $\boldsymbol{S}$ *that is a list of distinct variables from* $\mathbf{V} \cup \mathbf{U}$, *and a set* C *of* conditionals *of the following form:*

$$\varphi_1, \ldots, \varphi_n \Rightarrow \mathrm{insert}(a_1 : b_1, \ldots, a_m : b_m) \tag{6}$$

where $n \geq 0$, *all* φ_i *are MAPL set atoms or specifier atoms,* $m \geq 1$, *and all* a_j, b_j *are object terms. The part left of* $\Rightarrow$ *is the* precondition *and the part right of* $\Rightarrow$ *is the* action. *We require: (a) all set variables in the precondition must occur in the signature* $\boldsymbol{S}$*; and (b) all variables in the action occur either in the precondition or in the signature* $\boldsymbol{S}$.

A function term *for* $\langle F, \boldsymbol{S}, C\rangle$ *is an expression* $F(\boldsymbol{t})$, *where* $\boldsymbol{t}$ *is a list of MAPL terms of length* $|\boldsymbol{t}| = |\boldsymbol{S}|$, *so that* t_i *is an object term if* S_i *is an object variable, and a set term otherwise. For an interpretation* $\mathcal{I}$ *and variable assignment* $\mathcal{Z}$, *we define* $F(\boldsymbol{t})^{\mathcal{I},\mathcal{Z}}$ *to be the least set of attribute–value pairs such that, for each conditional of form* (6) *in* C *and all variable assignments* $\mathcal{Z}'$ *satisfying* $\mathcal{Z}'(S_i) = t_i^{\mathcal{I},\mathcal{Z}}$ *for all* $i \in \{1, \ldots, |\boldsymbol{S}|\}$, *if* $\mathcal{I}, \mathcal{Z}' \models \varphi_1 \wedge \ldots \wedge \varphi_n$ *then* $\langle a_j^{\mathcal{I},\mathcal{Z}'}, b_j^{\mathcal{I},\mathcal{Z}'}\rangle \in F(\boldsymbol{t})^{\mathcal{I},\mathcal{Z}}$ *for all* $j \in \{1, \ldots, m\}$. *We consider the empty conjunction to be vacuously satisfied in case* $n = 0$.

In words, the set $F(\boldsymbol{t})^{\mathcal{I},\mathcal{Z}}$ simply contains all attribute–value pairs that are entailed by some conditional of F.

Example 6. *Making a copy of optional attribute–value pairs as realised by the formula* $\Psi[X, Y]$ *in Example 1 can be expressed with a function definition* $\langle \mathsf{Copy}, \langle U\rangle, C\rangle$ *with* C *consisting of the following conditionals:*

$$\begin{aligned}\lfloor \mathsf{start} : z_1 \rfloor(U) &\Rightarrow \mathrm{insert}(\mathsf{start} : z_1)\\ \lfloor \mathsf{loc} : z_2 \rfloor(U) &\Rightarrow \mathrm{insert}(\mathsf{loc} : z_2)\\ \lfloor \mathsf{end} : z_3 \rfloor(U) &\Rightarrow \mathrm{insert}(\mathsf{end} : z_3)\end{aligned}$$

Example 7. *We can define binary functions that compute set operations on two annotations sets* U *and* V. *Union is captured with two conditionals* $\lfloor x : y\rfloor(U) \Rightarrow \mathrm{insert}(x : y)$ *and* $\lfloor x : y\rfloor(V) \Rightarrow \mathrm{insert}(x : y)$. *Intersection can be expressed by one conditonal* $\lfloor x : y\rfloor(U), \lfloor x : y\rfloor(V) \Rightarrow \mathrm{insert}(x : y)$. *Set difference is defined using a single conditional*

$$\lfloor x : y\rfloor(U), \big(\lfloor\rfloor \setminus \lfloor x : y \rfloor\big)(V) \Rightarrow \mathrm{insert}(x : y).$$

Arguably, such operations on annotations are essential when defining ontological relations on knowledge graphs. The next result confirms that this can be captured in MAPL.

Theorem 3. *For every function definition* $\langle F, \boldsymbol{S}, C\rangle$ *and corresponding function term* $F(\boldsymbol{t})$, *there is a MAPL formula* $\varphi_{F(\boldsymbol{t})}[U]$ *with a set variable* U *such that, for all interpretations* $\mathcal{I}$, *variable assignments* $\mathcal{Z}$, *and relations* $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$, *we have* $\mathcal{I}, \mathcal{Z}|_{U\mapsto R} \models \varphi_{F(\boldsymbol{t})}$ *iff* $R = F(\boldsymbol{t})^{\mathcal{I},\mathcal{Z}}$.

Theorem 3 allows us to use function terms for annotations within MAPL theories. A relational atom of the form $p(a_1, \ldots, a_n)@F(\boldsymbol{t})$ can be considered as an abbreviation for $\exists U.p(a_1, \ldots, a_n)@U \wedge \varphi_{F(\boldsymbol{t})}[U]$, and similarly for set atoms.

Example 8. *With the function definition* $\langle \mathsf{Copy}, \langle U\rangle, C\rangle$ *of Example 6, formula* (3) *of Example 1 can be written as*

$$\forall x, y.\forall U.\mathsf{spouse}(x, y)@U \rightarrow \mathsf{spouse}(y, x)@\mathsf{Copy}(U) \tag{7}$$

4 MAPL Rules

We now introduce a rule-based fragment of MAPL, where rules are allowed to contain arbitrary specifiers. While still giving us high expressivity, this is sufficient to obtain decidability for fact entailment.

Definition 9. *A* MAPL rule *is a MAPL formula of the form* $\forall \boldsymbol{U}.\forall \boldsymbol{x}.(\varphi \rightarrow \psi)$, *where* φ *is a conjunction of MAPL atoms and specifier atoms;* ψ *is a relational MAPL atom that may use a function term; and* $\boldsymbol{U}$ *and* $\boldsymbol{x}$ *are lists of all set and object variables, respectively, that occur in* φ *or* ψ. *We further require that every set variable or set term* $\{a_1 : b_1, \ldots, a_\ell : b_\ell\}$ *containing object variables must occur in some relational atom in* φ, *and that all variables in* ψ *occur in* φ.

We will tacitly omit the universal quantifiers, and call φ body *and* ψ head *of the rule. A* fact *is a rule where the body* φ *is an empty conjunction (interpreted as true).*

An ontology of multi-attributed rule-based predicate logic *(MARPL) consists of a finite set* $\mathbb{P}$ *of MAPL rules together with a finite set* $\mathbb{F}$ *of function definitions that includes exactly one definition for every function symbol used in* $\mathbb{P}$.

Previous examples such as (1) and (7) have already used rule-shaped formulae. Example 8 therefore showed a MARPL ontology. A slightly more complex example is given next.

Example 9. *MAPL rules allow for elaborate inferences, where we can enrich annotations by information from other statements. The following rule can, for example, infer from a* spouse *statement and a matching* dateOfDeath *statement that the marriage must have ended on the day of the death:* [2]

$$\begin{aligned}&\mathsf{spouse}(x, y)@U \wedge \mathsf{dateOfDeath}(x, z)@V\\ &\qquad\qquad \rightarrow \mathsf{spouse}(y, x)@\mathsf{EndedByDeath}(U, V, z)\end{aligned} \tag{8}$$

where the function definition $\langle \mathsf{EndedByDeath}, \langle U, V, z\rangle, \boldsymbol{C}\rangle$ *with the following conditionals is present:*

$$\begin{aligned}\lfloor \mathsf{start} : x\rfloor(U) &\Rightarrow \mathrm{insert}(\mathsf{start} : x),\\ \lfloor \mathsf{loc} : x\rfloor(U) &\Rightarrow \mathrm{insert}(\mathsf{loc} : x),\\ &\Rightarrow \mathrm{insert}(\mathsf{end} : z, \mathsf{endCause} : \mathsf{Death})\end{aligned} \tag{9}$$

[2] This simplified example is inspired by actual usage on Wikidata, where one may indeed find spouse statements that are annotated with an end cause death.

Given the facts $\mathsf{spouse}(\mathsf{todd}, \mathsf{taylor})@\{\mathsf{start} : \mathsf{1957\text{-}02\text{-}02}\}$ *and* $\mathsf{dateOfDeath}(\mathsf{todd}, \mathsf{1958\text{-}03\text{-}22})@\{\}$*, by using the above rule, we can infer* $\mathsf{spouse}(\mathsf{taylor}, \mathsf{todd})@\{\mathsf{start} : \mathsf{1957\text{-}02\text{-}02}, \mathsf{end} : \mathsf{1958\text{-}03\text{-}22}, \mathsf{endCause} : \mathsf{Death}\}$*, explaining the end of one of Taylor's marriages.*

While specifiers and function definitions are syntactic sugar for MAPL, they add true expressive power to MARPL. Indeed, although MARPL rules use only universal second-order quantifiers explicitly, they are not in the fragment of *universal second-order logic* (USO), since specifiers and function definitions introduce hidden second-order existentials.

Nevertheless, MARPL is not truly a higher-order logic. For the following statement, we consider first-order logic with equality $\approx$ ($\mathrm{FO}_\approx$) under the unique name assumption (UNA):

Theorem 4. *Every MARPL ontology* $\langle \mathbb{P}, \mathbb{F} \rangle$ *can be translated into a set of* $\mathit{FO}_\approx$ *sentences* $\mathsf{M2FO}(\mathbb{P}, \mathbb{F})$*, and every variable-free MAPL atom* φ *into a* $\mathit{FO}_\approx$ *sentence* $\mathsf{M2FO}(\varphi)$*, such that* $\langle \mathbb{P}, \mathbb{F} \rangle \models \varphi$ *iff* $\mathsf{M2FO}(\mathbb{P}, \mathbb{F}) \models \mathsf{M2FO}(\varphi)$ *under the UNA.*

The translation $\mathsf{M2FO}$ is based on the representation of MARS in standard first-order relational structures. For a given MARS predicate symbol p of arity n, we introduce a new $(n+1)$-ary first-order predicate symbol $\bar{p}$. Moreover, we use an auxiliary predicate in of arity $\mathsf{ar}(\mathsf{in}) = 3$. We can now encode a MARS tuple $\langle \delta_1, \ldots, \delta_n, S \rangle \in p^{\mathcal{I}}$, where S is a binary relation $\{\langle \epsilon_1, \tau_1 \rangle, \ldots, \langle \epsilon_m, \tau_m \rangle\}$, using tuples $\langle \delta_1, \ldots, \delta_n, \sigma \rangle \in \bar{p}^{\mathcal{I}}$ and $\langle \epsilon_1, \tau_1, \sigma \rangle \in \mathsf{in}^{\mathcal{I}}, \ldots, \langle \epsilon_m, \tau_m, \sigma \rangle \in \mathsf{in}^{\mathcal{I}}$. That is, we introduce an auxiliary domain element σ to represent the set S. Translated formulae accordingly use first-order variables to represent sets, e.g.,a MAPL atom $(a : b) \in U$ turns into $\mathsf{in}(a, b, u)$, where u is a first-order variable that is specific to U. We simply use lower-case variants of set variables to indicate this switch. As rules are safe and MARPL does not admit counting quantifiers, this mixing of object and set variables is unproblematic. A variable-free relational MAPL atom $\varphi = p(t_1, \ldots, t_n)@\{a_1 : b_1, \ldots, a_m : b_m\}$ therefore becomes

$$\begin{aligned}\mathsf{M2FO}(\varphi) := \exists u. \bar{p}(t_1, \ldots, t_n, u) \wedge \textstyle\bigwedge_{i=1}^{m} \mathsf{in}(a_i, b_i, u) \wedge \\ \forall x, y. \mathsf{in}(x, y, u) \rightarrow \textstyle\bigvee_{i=1}^{n} (x \approx a_i \wedge y \approx b_i)\end{aligned} \tag{10}$$

The translation of MARPL rules uses similar formulae to capture relational and specifier atoms. For every function definition $\langle F, \boldsymbol{S}, C \rangle$, we introduce a fresh $(|\boldsymbol{S}| + 1)$-ary predicate symbol $\bar{F}$. Let $\boldsymbol{s}$ be the signature of F with all set variables replaced by first-order variables. For every conditional $\varphi_1, \ldots, \varphi_n \Rightarrow \mathsf{insert}(a_1 : b_1, \ldots, a_m : b_m)$ in C, we add a rule

$$\bar{F}(\boldsymbol{s}, u) \wedge \bigwedge_{1 \le i \le n} \bar{\varphi}_i \rightarrow \bigwedge_{1 \le j \le m} \mathsf{in}(a_j, b_j, u) \tag{11}$$

where $\bar{\varphi}_i$ denotes the first-order translation of the atom φ_i. In combination with the translated MARPL rules, this yields the required translation $\mathsf{M2FO}(\mathbb{P}, \mathbb{F})$.

This shows that MARPL reasoning is semidecidable. We show in Section 5 that it is decidable in ExpTime, which is the best result that one could expect considering the ExpTime-hardness of Datalog. Yet, surprisingly, this bound is tight for *data complexity*, which is much lower (PTime) for Datalog.

Theorem 5. *Deciding fact entailment for MARPL ontologies is* ExpTime*-hard, even when restricting to classes of ontologies where only the set of facts is allowed to vary.*

Our proof is based on an interesting reduction from the word problem of polynomially space-bounded Alternating Turing Machines, where we use attribute–value sets to represent ATM configurations. The key idea is to compute accepting configurations *backwards* starting from the final configuration. The structure of the polynomial tape can be given in annotations that are part of the input facts. Function definitions can then perform most of the computation needed to determine possible predecessor configurations. The ATM accepts if the initial configuration can be derived in this way.

5 Reasoning with MARPL

We are interested in reasoning with MARPL. A natural reasoning task in this context is *fact entailment*: given a MARPL ontology $\langle \mathbb{P}, \mathbb{F} \rangle$ and a variable-free relational MAPL atom φ, we wish to decide if $\langle \mathbb{P}, \mathbb{F} \rangle \models \varphi$ holds. More general query answering can be reduced to this task. We have already established the semi-decidability of reasoning in MARPL. We now present a decision procedure that, informally speaking, proceeds as follows:

- We consider all annotations of previously derived facts as fixed and final ("immutable"). This makes it easy to evaluate atoms, including all specifier atoms, whether or not they require negation.
- When applying an ontological rule, we create a new annotation as computed from its defining function. This is a finite "initialisation" process that creates new sets.
- We eagerly eliminate duplicates, i.e., relational atoms that agree on the first-order atom and annotation after initialisation.

Exhaustive rule application along these lines yields an algorithm that can be applied directly on MARPL ontologies to compute the least MARS that satisfies them.

Definition 10. *A* ground (MAPL) substitution σ *is a partial mapping from object and set variables to constants and variable-free set terms, respectively. Given a term or formula* φ*, we write* $\varphi\sigma$ *for the term or formula obtained by replacing each variable* χ *in* φ *with* $\sigma(\chi)$*, whenever this is defined.*

Definition 11. *The* MARS chase procedure *takes a MARPL ontology* $\langle \mathbb{P}, \mathbb{F} \rangle$ *and constructs a finite interpretation* $\mathcal{I}$ *as follows. The domain of* $\mathcal{I}$ *is the finite set of constants* $c \in \mathbf{C}$ *in* $\langle \mathbb{P}, \mathbb{F} \rangle$*; and such constants are interpreted as* $c^{\mathcal{I}} := c$*. We initialise* $p^{\mathcal{I}} := \emptyset$ *for every predicate* p*, and repeat the following steps until no more changes occur:*

(A) Let $\varphi \rightarrow p(t_1, \ldots, t_n)@\Psi \in \mathbb{P}$ *be a rule and let* σ *be a ground substitution such that* $\mathcal{I} \models \varphi\sigma$ *but* $\mathcal{I} \not\models p(t_1, \ldots, t_n)@\Psi\sigma$*.*

(B) If Ψ *is a set term or a set variable, set* $T := S\sigma$*. Note that* Ψ *is ground, because* Ψ *is either a variable-free set term or appears in* φ*. Otherwise, if* $\Psi = F(\tau_1, \ldots, \tau_\ell)$*, set* $T := F(\tau_1\sigma, \ldots, \tau_\ell\sigma)$*.*

(C) Set $p^{\mathcal{I}} := p^{\mathcal{I}} \cup \{\langle (t_1\sigma)^{\mathcal{I}}, \ldots, (t_n\sigma)^{\mathcal{I}}, T^{\mathcal{I}} \rangle\}$*.*

The result $\mathcal{I}$ *of this procedure is denoted* $\mathcal{I}(\mathbb{P}, \mathbb{F})$*.*

We start out with a finite set of constants, so there are only finitely many finite sets over this set of constants. Hence, there are only finitely many facts that can be derived, so the algorithm terminates after finitely many steps.

The next result shows that $\mathcal{I}(\mathbb{P},\mathbb{F})$ plays the role of a *least model* for MARPL, similar to the least model of Datalog. While MARPL specifiers introduce a form of negation, the saturation of Definition 11 remains monotone, since specifiers refer to annotations of previously derived ground atoms, whereas rule applications can only derive new ground atoms but never modify existing ones. This mechanism can also be understood as a form of local stratification of negation.

Theorem 6. *For any MARPL ontology $\langle\mathbb{P},\mathbb{F}\rangle$, we have $\langle\mathbb{P},\mathbb{F}\rangle \models \chi$ iff $\mathcal{I}(\mathbb{P},\mathbb{F}) \models \chi$ for any variable-free atom χ.*

Proof. Clearly, the MARS chase terminates after finitely many steps and the resulting interpretation $\mathcal{I}(\mathbb{P},\mathbb{F})$ is finite.

Further note that the MARS chase is monotone in the sense that $\mathcal{I}(\mathbb{P},\mathbb{F})$ can be considered as a set of ground facts, where each rule application may add new facts but not modify or delete previously derived ones.

For soundness, we show by induction that $\mathcal{I}(\mathbb{P},\mathbb{F}) \models \alpha$ implies $\mathcal{J} \models \alpha$ for every model $\mathcal{J}$ of $\langle\mathbb{P},\mathbb{F}\rangle$, and every ground fact α. Clearly, any fact in $\mathbb{P}$ holds in $\mathcal{J}$. Consider now an applicable rule $\varphi \to \psi \in \mathbb{P}$. Then there is a ground substitution σ such that $\mathcal{I}(\mathbb{P},\mathbb{F}) \models \varphi\sigma$. Hence, by the induction hypothesis, $\mathcal{J} \models \varphi\sigma$. Any fact added to $\mathcal{I}(\mathbb{P},\mathbb{F})$ by applying $\varphi \to \psi$ must also be present in $\mathcal{J}$, since all variables in ψ appear in φ (and are therefore mapped to constants by σ) and $\mathcal{J} \models \varphi \to \psi$.

For completeness, we show that $\mathcal{I}(\mathbb{P},\mathbb{F})$ is indeed a model of $\langle\mathbb{P},\mathbb{F}\rangle$. Assume to the contrary that some rule $\varphi \to \psi \in \mathbb{P}$ does not hold in $\mathcal{I}(\mathbb{P},\mathbb{F})$, i.e., for some variable assignment $\mathcal{Z}$, we have $\mathcal{I}(\mathbb{P},\mathbb{F}),\mathcal{Z} \models \varphi$, but $\mathcal{I}(\mathbb{P},\mathbb{F}),\mathcal{Z} \not\models \psi$. Then $\varphi \to \psi$ is an applicable rule, and its application results in at least one fact being added to $\mathcal{I}(\mathbb{P},\mathbb{F})$ in (A) – contradiction. □

With k constants, one can define 2^{k^2} distinct annotations. If the maximal arity of predicates in an ontology is a, there are at most $|\mathbf{P}| \cdot k^a \cdot 2^{k^2}$ MAPL facts. This yields an upper bound for the number of iterations in the MAPL chase, and thus a matching upper bound for Theorem 5. Note that individual rule applications can also be done in this bound: we can find the required match (A), and compute the new fact (C) in NP.

Theorem 7. *Deciding fact entailment for MARPL ontologies is in* ExpTime.

Our estimation for the upper bound also reveals that the hardness for data complexity is related to the size of annotations, which occurs in the exponent in 2^{k^2}. We can obtain lower bounds by restricting to cases where k is bounded by a constant. A MARPL ontology $\langle\mathbb{P},\mathbb{F}\rangle$ belongs to MARPL$_k$ for $k \geq 0$ iff for any entailed fact $\langle\mathbb{P},\mathbb{F}\rangle \models p(t_1,\ldots,t_n)@S$, the number of attribute–value pairs in S is at most k.

Theorem 8. *For any $k \geq 0$, deciding fact entailment for MARPL$_k$ ontologies is in* PTime *with respect to data complexity, i.e., when considering rules except facts to be fixed.*

Note that, while data complexity generally allows us to vary the set of facts while keeping other rules fixed, not all sets of facts may lead to an ontology in MARPL$_k$. Theorem 8 states that we recover the low data complexity of Datalog if annotation sizes are bounded. This insight is practically relevant since typical Property Graphs do not use very large annotations, but it is not easy to invoke Theorem 8 directly: reasoning is necessary to determine if an ontology (with a given set of facts) is in MARPL$_k$. However, one can find simpler syntactic conditions, e.g., for ontologies without function definitions:

Theorem 9. *An ontology $\langle\mathbb{P},\emptyset\rangle$ is in MARPL$_k$ for every k greater or equal to the largest set term in (facts or rules of) $\mathbb{P}$.*

6 Discussion and Outlook

We presented a first approach towards ontological modelling on Property Graphs and related data formats. Our rule-based formalism MARPL captures key expressive features of our general, second-order logic framework while ensuring approachable reasoning complexity based on practical bottom-up algorithms. Combined complexity agrees with other popular rule languages, such as Datalog, but it may seem problematic that worst-case reasoning time may grow exponentially in the size of the data (which is usually large). However, our hardness proof requires attribute–value sets to grow arbitrarily large, which we think is not to be expected in practical applications. Typical applications rather have bounded-size annotations, for which data complexity drops to PTime.

Nevertheless, our ExpTime data complexity (and even more so the non-semidecidability of our overall framework) underlines that our work is fundamentally different from other scenarios studied in AI. Indeed, we are not aware of any closely related work. Attributed graphs are supported by several database management systems, but even basic "integrity constraints are poorly studied in graph databases" [Angles, 2012]. Somewhat related to our work are recursive graph query languages over advanced graph models, such as SNQL [Martín *et al.*, 2011], GraphQL [He and Singh, 2008], G-Log [Paredaens *et al.*, 1995], and GraphLog [Consens and Mendelzon, 1989]. As MARPL extends Datalog, it subsumes Datalog fragments such as GraphLog, but other algebras sport incomparable expressivity such as the computation of aggregates in SNQL. Maybe closest to the spirit of our work, Moustafa *et al.* proposed an extension of Datalog where set-valued parameters represent edge attributes for social network analysis [2011].

None of these works are truly concerned with knowledge representation or AI, and indeed it seems that this community has largely ignored enriched graph models so far. Our proposal can offer a first bridge between these hitherto disconnected areas. It now remains to explore how the wealth of insights obtained for other ontology languages can be applied to this new field. Language design, expressive extensions, deduction methods, and implementations all are worthwhile areas of future study with a huge potential for applications.

Acknowledgements

This work is partly supported by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (cfaed), the Collaborative Research Center SFB 912 (HAEC), and in Emmy Noether grant KR 4381/1-1 (DIAMOND).

References

[Angles, 2012] Renzo Angles. A comparison of current graph database models. In *Workshops Proceedings of the IEEE 28th International Conference on Data Engineering, ICDE 2012, Arlington, VA, USA, April 1-5, 2012*, pages 171–177, 2012.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proc. 2008 ACM SIGMOD Int. Conf. on Management of Data*, pages 1247–1250. ACM, 2008.

[Consens and Mendelzon, 1989] M. P. Consens and A. O. Mendelzon. Expressing structural hypertext queries in graphlog. In *Proceedings of the Second Annual ACM Conference on Hypertext*, HYPERTEXT '89, pages 269–292, New York, NY, USA, 1989. ACM.

[Green *et al.*, 2007] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40, 2007.

[He and Singh, 2008] Huahai He and Ambuj K. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 405–418, 2008.

[Hoffart *et al.*, 2013] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *J. of Artif. Intell.*, 194:28–61, 2013.

[Leivant, 1994] Daniel Leivant. Higher order logic. In Dov M. Gabbay, Christopher J. Hogger, J. A. Robinson, and Jörg H. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 2, Deduction Methodologies*, pages 229–322. Oxford University Press, 1994.

[Martín *et al.*, 2011] Mauro San Martín, Claudio Gutierrez, and Peter T. Wood. SNQL: A social networks query and transformation language. In *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May 9-12, 2011*, 2011.

[Moustafa *et al.*, 2011] Walaa Eldin Moustafa, Galileo Namata, Amol Deshpande, and Lise Getoor. Declarative analysis of noisy information networks. In *Workshops Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 106–111, 2011.

[OWL Working Group, 2009] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.

[Paredaens *et al.*, 1995] Jan Paredaens, Peter Peelman, and Letizia Tanca. G-log: A graph-based query language. *IEEE Trans. on Knowl. and Data Eng.*, 7(3):436–453, June 1995.

[Rodriguez and Neubauer, 2010] Marko A. Rodriguez and Peter Neubauer. Constructions from dots and lines. *Bulletin of the American Society for Information Science and Technology*, 36(6):35–41, 2010.

[Rudolf *et al.*, 2013] Michael Rudolf, Marcus Paradies, Christof Bornhövd, and Wolfgang Lehner. The graph story of the SAP HANA database. In *Datenbanksysteme für Business, Technologie und Web (BTW), 15. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 11.-15.3.2013 in Magdeburg, Germany. Proceedings*, pages 403–420, 2013.

[Straccia *et al.*, 2010] Umberto Straccia, Nuno Lopes, Gergely Lukacsy, and Axel Polleres. A general framework for representing and reasoning with annotated Semantic Web data. In Maria Fox and David Poole, editors, *Proc. 24th AAAI Conf. on Artificial Intelligence (AAAI'10)*. AAAI Press, 2010.

[Udrea *et al.*, 2010] Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Trans. Comput. Logic*, 11(2):10:1–10:41, 2010.

[Vrandečić and Krötzsch, 2014] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10), 2014.

The appendix provides proofs and additional definitions that were omitted from the main text for lack of space.

A Proofs for Section 3

Before proving Theorems 2 and 3, we show two auxiliary results that we will use in these proofs.

Lemma 1. *There is a MAPL formula* subset$[U,V]$ *with set variables U,V, such that for all interpretations $\mathcal{I}$, variable assignments $\mathcal{Z}$, and relations $R,P \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$, we have $\mathcal{I},\mathcal{Z}|_{U\mapsto R,V\mapsto P} \models \text{subset}[U,V]$ iff $R \subseteq P$.*

Proof. Set $\text{subset}[U,V] := \forall x,y.(x:y) \in U \to (x:y) \in V$.

Let $\mathcal{I}$ be an interpretation, $\mathcal{Z}$ a variable assignment, and $R,P \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$ finite binary relations. Clearly $\mathcal{I},\mathcal{Z}|_{U\mapsto R,V\mapsto P} \models \text{subset}[U,V]$ iff $\langle\delta,\epsilon\rangle \in R \to \langle\delta,\epsilon\rangle \in P$ for all $\delta,\epsilon \in \Delta$, i.e., iff $R \subseteq P$. □

Lemma 2. *For any formula $\psi[X]$ with free set variable X, there is a MAPL formula $\min^X_\psi[X]$ with free variable X such that for all interpretations $\mathcal{I}$, variable assignments $\mathcal{Z}$, and relations $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$, we have $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \min^X_\psi$ iff R is the least relation satisfying $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \psi$.*

Proof. Let $\psi[X]$ be a MAPL formula. Set

$$\min^X_\psi := \psi[X] \land \left(\forall Y.\psi[X/Y] \to \text{subset}[X,Y]\right)$$

where $\psi[X/Y]$ denotes $\psi[X]$ with free occurrences of X replaced by Y, and Y is chosen such that it does not occur as a free variable in ψ.

Let $\mathcal{I}$ be an interpretation, $\mathcal{Z}$ a variable assignment, and $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$ a finite binary relation satisfying $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \min^X_\psi$. Then $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \psi$. Suppose for a contradiction that R is not minimal, i.e., $\mathcal{I},\mathcal{Z}|_{X\mapsto P} \models \psi$ for some $P \subsetneq R$. Then $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \forall Y.\psi[X/Y] \to \text{subset}[X,Y]$ yields a contradiction. Conversely, if R is the least relation satisfying $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \psi$, for any relation Q satisfying $\mathcal{I},\mathcal{Z}|_{X\mapsto Q} \models \psi$, we have $R \subseteq Q$, yielding $\mathcal{I},\mathcal{Z}|_{X\mapsto R} \models \min^X_\psi$. □

Theorem 2. *For every specifier Ω, there is a MAPL formula $\varphi_\Omega[U]$ with a set variable U such that, for all interpretations $\mathcal{I}$, variable assignments $\mathcal{Z}$, and relations $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$, we have $R \in \Omega^{\mathcal{I},\mathcal{Z}}$ iff $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$.*

Proof. We start by transforming closed specifiers into equivalent MAPL formulae before generalising in turn to open specifiers and then to arbitrary specifier expressions.

Let $\mathcal{I}$ be an interpretation, $\mathcal{Z}$ a variable assignment, and $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$ a finite binary relation.

Consider first $\Omega := [a : b]$, where a, b are object terms, i.e., $b \notin \{*, +\}$. Set $\varphi_\Omega := \min^U_{(a:b)\in U}$. Clearly, $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$ iff R is the least set satisfying $\langle a^{\mathcal{I},\mathcal{Z}}, b^{\mathcal{I},\mathcal{Z}}\rangle \in R$, i.e., iff $R = \{\langle a^{\mathcal{I},\mathcal{Z}}, b^{\mathcal{I},\mathcal{Z}}\rangle\} = \{a : b\}^{\mathcal{I},\mathcal{Z}}$. By definition, this is equivalent to $R \in \{\{a : b\}^{\mathcal{I},\mathcal{Z}}\} = \Omega^{\mathcal{I},\mathcal{Z}}$.

Next, let $\Omega := [a : *]$ for some object term a. Define

$$\varphi_\Omega := \exists Z.\min^Z_{\forall x,y.(x:y)\in U \to (a:y)\in Z}[U,Z] \land \text{subset}[U,Z]$$

Note that the first conjunct is a formula with free variables Z and U. Then $R = \{\langle a^{\mathcal{I},\mathcal{Z}}, \delta_1\rangle, \dots, \langle a^{\mathcal{I},\mathcal{Z}}, \delta_\ell\rangle\}$ for some $\ell \geq 0$ and $\delta_i \in \Delta^{\mathcal{I}}$. Suppose for a contradiction that $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$, but $\langle\gamma,\delta\rangle \in R$ for $\gamma,\delta \in \Delta^{\mathcal{I}}$ such that $\gamma \neq a^{\mathcal{I},\mathcal{Z}}$. Then there is a relation P that is the least set satisfying, for $\epsilon \in \Delta^{\mathcal{I}}$, $\langle\epsilon,\delta\rangle \in R$ only if $\langle a^{\mathcal{I},\mathcal{Z}},\delta\rangle \in P$. But then either $\langle\gamma,\delta\rangle \notin P$, contradicting $R \subseteq P$, or $\langle\gamma,\delta\rangle \in P$, which contradicts the minimality of P. Hence, $R \in \Omega^{\mathcal{I},\mathcal{Z}}$.

Similarly, for $\Omega := [a : +]$, define $\varphi_\Omega := \varphi_{[a:*]} \land \exists x.(a : x) \in U$. Clearly, $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$ iff R is nonempty and $R \in [a : *]^{\mathcal{I},\mathcal{Z}}$, i.e., $R = \{\langle a^{\mathcal{I},\mathcal{Z}}, \delta_1\rangle, \dots, \langle a^{\mathcal{I},\mathcal{Z}}, \delta_\ell\rangle\}$ for some $\ell \geq 1$, $\delta_i \in \Delta^{\mathcal{I}}$.

Finally, let $\Omega := [a_1 : b_1, \dots, a_n : b_n]$ be an arbitrary closed specifier, where a_i are object terms and b_i are either object terms or one of $*$ and $+$, for $i \in \{1, \dots, n\}$. Set

$$\varphi_\Omega := \exists U_1, \dots, U_n.\ \bigwedge_{1\leq i\leq n} \varphi_{[a_i:b_i]}[U/U_i] \land$$

$$\min^U_{\bigwedge_{1\leq i\leq n}\text{subset}[U_i,U]}$$

Then $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$ iff R is the least set satisfying $S_i \subseteq R$ for relations $S_i \in [a_i : b_i]^{\mathcal{I},\mathcal{Z}}$ $(1 \leq i \leq n)$, i.e., $R = \bigcup_{i=1}^n S_i$. Equivalently, $R \in \left\{\bigcup_{i=1}^n S_i \mid S_i \in [a_i : b_i]^{\mathcal{I},\mathcal{Z}}\right\} = \Omega^{\mathcal{I},\mathcal{Z}}$.

We now turn to open specifiers. Consider an arbitrary open specifier $\Omega := \lfloor a_1 : b_1, \dots, a_n : b_n\rfloor$, where a_i are object terms and b_i are either object terms or one of $*$ and $+$, for $i \in \{1, \dots, n\}$. We set $\varphi_\Omega := \exists Z.\varphi_{[a_1:b_1,\dots,a_n:b_n]}[U/Z] \land \text{subset}[Z,U]$. Clearly, we have $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$ iff there is a subset $R' \subseteq R$ such that $R' \in [a_1 : b_1, \dots, a_n : b_n]^{\mathcal{I},\mathcal{Z}}$.

We use induction to show that arbitrary specifiers can be translated into corresponding MAPL formulae.

First, consider $\Omega := \Omega_1 \cap \Omega_2$. Set $\varphi_\Omega := \varphi_{\Omega_1} \land \varphi_{\Omega_2}$. Then $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_\Omega$ iff $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_{\Omega_1}$ and $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_{\Omega_2}$, i.e., iff $R \in \Omega_1^{\mathcal{I},\mathcal{Z}} \cap \Omega_2^{\mathcal{I},\mathcal{Z}} = \Omega^{\mathcal{I},\mathcal{Z}}$.

Similarly, for $\Omega := \Omega_1 \cup \Omega_2$, set $\varphi_\Omega := \varphi_{\Omega_1} \lor \varphi_{\Omega_2}$. Finally, for $\Omega := \Omega_1 \setminus \Omega_2$, define $\varphi_\Omega := \varphi_{\Omega_1} \land \neg\varphi_{\Omega_2}$. □

Theorem 3. *For every function definition $\langle F, \boldsymbol{S}, C\rangle$ and corresponding function term $F(\boldsymbol{t})$, there is a MAPL formula $\varphi_{F(\boldsymbol{t})}[U]$ with a set variable U such that, for all interpretations $\mathcal{I}$, variable assignments $\mathcal{Z}$, and relations $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$, we have $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_{F(\boldsymbol{t})}$ iff $R = F(\boldsymbol{t})^{\mathcal{I},\mathcal{Z}}$.*

Proof. Let $\langle F, \boldsymbol{S}, C\rangle$ be an arbitrary function definition, $F(\boldsymbol{t})$ a corresponding function term, $\mathcal{I}$ an interpretation, $\mathcal{Z}$ a variable assignment, and $R \in \mathcal{P}_{\mathsf{fin}}\left(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}\right)$. First, for each conditional $c := \varphi_1, \dots, \varphi_n \Rightarrow \text{insert}(a_1 : b_1, \dots, a_m : b_m)$ in C with free object variables $\boldsymbol{x}$ in $\varphi_1, \dots, \varphi_n$, we define $\psi_c := \forall\boldsymbol{x}.\bigwedge_{i=1}^n \varphi_i \to \bigwedge_{i=1}^m (a_i : b_i) \in U$. Furthermore, we set $\varphi_{F(\boldsymbol{t})} := \min^U_{\bigwedge_{c\in C}\psi_c}$. Then $\mathcal{I},\mathcal{Z}|_{U\mapsto R} \models \varphi_{F(\boldsymbol{t})}$ iff R is the least set satisfying each ψ_c, i.e., the least set containing all tuples that are entailed by some conditional $c \in C$. □

B Proofs for Section 4

Theorem 4. *Every MARPL ontology $\langle\mathbb{P},\mathbb{F}\rangle$ can be translated into a set of FO$_\approx$ sentences $\mathsf{M2FO}(\mathbb{P},\mathbb{F})$, and every variable-free MAPL atom φ into a FO$_\approx$ sentence $\mathsf{M2FO}(\varphi)$, such that $\langle\mathbb{P},\mathbb{F}\rangle \models \varphi$ iff $\mathsf{M2FO}(\mathbb{P},\mathbb{F}) \models \mathsf{M2FO}(\varphi)$ under the UNA.*

Proof. Our translation is based on the representation of MARS in standard first-order relational structures. For a given MARS predicate symbol p of arity n, we introduce a new $(n+1)$-ary first-order predicate symbol $\bar{p}$. Moreover, we use an auxiliary predicate in of arity $\mathsf{ar}(\mathsf{in}) = 3$. We can now encode a MARS tuple $\langle \delta_1, \ldots, \delta_n, S\rangle \in p^{\mathcal{I}}$, where S is a binary relation $\{\langle \epsilon_1, \tau_1\rangle, \ldots, \langle \epsilon_m, \tau_m\rangle\}$, using tuples $\langle \delta_1, \ldots, \delta_n, \sigma\rangle \in \bar{p}^{\mathcal{I}}$ and $\langle \epsilon_1, \tau_1, \sigma\rangle \in \mathsf{in}^{\mathcal{I}}, \ldots, \langle \epsilon_m, \tau_m, \sigma\rangle \in \mathsf{in}^{\mathcal{I}}$. That is, we introduce an auxiliary domain element σ to represent the set S.

Translated formulae accordingly use first-order variables to represent sets. For example, a MAPL atom $(a : b) \in U$ turns into $\mathsf{in}(a, b, u)$, where u is a first-order variable that is specific to U. We simply use lower-case variants of set variables to indicate this switch. Note that we do not need to distinguish object and set variables in the translation since rules are safe and MARPL does not admit counting quantifiers. Furthermore, we assume all free variables in formulae to be universally quantified.

We begin with some simplifications. (S1) We replace each set term $\{a_1 : b_1, \ldots, a_\ell : b_\ell\}$ in a rule by a fresh set variable U and add a new body atom $[a_1 : b_1, \ldots, a_\ell : b_\ell](U)$. (S2) We extend the function definitions $\mathbb{F}$ by a new "copy" function definition $\langle \mathsf{cp}, U, \{(x : y) \in U \Rightarrow \mathrm{insert}(x : y)\}\rangle$. Then we replace all rule heads of the form $p(\ldots)@V$ by the atom $p(\ldots)@\mathsf{cp}(V)$. (S3) If a rule body contains two relational atoms $\alpha_1@U$ and $\alpha_2@U$ with the same set variable U, then we replace one occurrence of U by a fresh set variable U' and add a new body atom $u \approx u'$. Note that the latter is a first-order atom, so the result is partially translated. It is not hard to see that (S1)–(S3) preserve semantics.

Next, each relational atom $p(t_1, \ldots, t_n)@U$ in a rule body is replaced by the corresponding first-order atom $\bar{p}(t_1, \ldots, t_n, u)$. Similarly, body atoms $(a : b) \in U$ are replaced by $\mathsf{in}(a, b, u)$. In either case, U is a variable by (S1).

Towards translating specifier atoms and function definitions, we first establish two auxiliary definitions, where $\psi[u]$ is a formula with free variable u and $\psi[u/v]$ denotes ψ with unbound occurrences of u replaced by v.

$$\begin{aligned} \mathrm{subset}[u, v] &:= \forall x, y. \mathsf{in}(x, y, u) \rightarrow \mathsf{in}(x, y, v) \\ \mathrm{min}^u_\psi &:= \psi[u] \wedge (\forall v. \psi[u/v] \rightarrow \mathrm{subset}(u, v)) \end{aligned}$$

We require that identical sets have identical corresponding first-order variables:

$$\mathrm{subset}[u, v] \wedge \mathrm{subset}[v, u] \rightarrow u \approx v.$$

We continue by transforming specifier atoms, i.e., expressions of the form $\Omega(U)$, where U is a set variable (by (S1)). We write $\Omega[\boldsymbol{x}]$ to say that Ω contains the free object variables $\boldsymbol{x}$ (specifiers never contain set variables). We replace $\Omega(U)$ by the atom $\bar{\Omega}(\boldsymbol{x}, u)$, with $\bar{\Omega}$ a fresh $(|\boldsymbol{x}| + 1)$-ary predicate. Then we add auxiliary formulae depending on Ω, which we obtain by translating the MAPL formulae from the proof of Theorem 2. For example, $\Omega = [a : b]$ with $b \notin \{*, +\}$ translates to

$$\mathrm{min}^u_{\mathsf{in}(a,b,u)} \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \tag{12}$$

whereas for $\Omega = [a : *]$, we add

$$\exists z.\, \mathrm{subset}[u, z] \wedge \mathrm{min}^z_{\forall x,y. \mathsf{in}(x,y,u) \rightarrow \mathsf{in}(a,y,z)} \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \tag{13}$$

Note that the variables $\boldsymbol{x}$ occur among the terms a, b. Similarly, for $\Omega = [a : +]$, we add

$$\exists y. \bar{\Omega}_{[a:*]}(\boldsymbol{x}, u) \wedge \mathsf{in}(a, y, u) \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \tag{14}$$

where $\bar{\Omega}_{[a:*]}$ denotes the translation of $[a : *]$. For $\Omega = [a_1 : b_1, \ldots, a_\ell : b_\ell]$ and $\Omega_i = [a_i : b_i]$, we add

$$\begin{aligned} \exists u_1, \ldots, u_n. \textstyle\bigwedge_{1 \leq i \leq n} \bar{\Omega}_i(\boldsymbol{x}, u_i) \wedge \mathrm{min}^u_{\bigwedge_{1 \leq i \leq n} \mathrm{subset}(u_i, u)} \\ \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \end{aligned} \tag{15}$$

along with the respective formulae for each Ω_i, extended to $(|\boldsymbol{x}| + 1)$-ary predicates, if necessary. For an open specifier $\lfloor a_1 : b_1, \ldots, a_n : b_n \rfloor$, we add

$$\exists z.\, \mathrm{subset}[z, u] \wedge \bar{\Omega}_{[a_1:b_1,\ldots,a_n:b_n]}(\boldsymbol{x}, z) \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u)$$

along with the translation $\bar{\Omega}_{[a_1:b_1,\ldots,a_n:b_n]}$ of the corresponding closed specifier $[a_1 : b_1, \ldots, a_n : b_n]$.

If $\Omega[\boldsymbol{x}] = \Omega_1[\boldsymbol{x_1}] \diamond \Omega_2[\boldsymbol{x_2}]$ with $\diamond \in \{\cap, \cup, \setminus\}$, we create recursive auxiliary formulae for $\bar{\Omega}_1$ and $\bar{\Omega}_2$. The we add:

$$\bar{\Omega}_1(\boldsymbol{x_1}, u) \wedge \bar{\Omega}_2(\boldsymbol{x_2}, u) \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \qquad \text{if } \diamond = \cap \tag{16}$$

$$\bar{\Omega}_1(\boldsymbol{x_1}, u) \vee \bar{\Omega}_2(\boldsymbol{x_2}, u) \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \qquad \text{if } \diamond = \cup \tag{17}$$

$$\bar{\Omega}_1(\boldsymbol{x_1}, u) \wedge \neg\bar{\Omega}_2(\boldsymbol{x_2}, u) \leftrightarrow \bar{\Omega}(\boldsymbol{x}, u) \qquad \text{if } \diamond = \setminus \tag{18}$$

In a similar fashion, we can obtain the translation of function definitions by translating the formulae from the proof of Theorem 3. Consider a function definition $\langle F, \boldsymbol{S}, C\rangle$. For a conditional $c := \varphi_1, \ldots, \varphi_n \Rightarrow \mathrm{insert}(a_1 : b_1, \ldots, a_m : b_m)$ with free object variables $\boldsymbol{x}$, define $\psi_c := \forall \boldsymbol{x}. \bigwedge_{i=1}^n \varphi_i \rightarrow \bigwedge_{i=1}^m \mathsf{in}(a_i, b_i, u)$, and add

$$\mathrm{min}^u_{\bigwedge_{c \in C} \psi_c} \leftrightarrow \bar{F}(\boldsymbol{s}, u)$$

where $\boldsymbol{s}$ is the signature $\boldsymbol{S}$ with set variables replaced by the corresponding first-order variables, and $\bar{F}$ is a fresh $(|\boldsymbol{S}| + 1)$-ary predicate symbol. We denote the set of all formulae obtained by translating an ontology $\langle \mathbb{P}, \mathbb{F}\rangle$ in this fashion by $\mathsf{M2FO}(\mathbb{P}, \mathbb{F})$, including all respective auxiliary formulae.

The correctness of this translation follows by a similar argumentation to the proofs of Theorems 2 and 3. □

Theorem 5 refers to data complexity. For clarity, we give a more formal definition of this notion here.

Definition 12. *For* data complexity, *we consider the following decision problem. Fix a MARPL ontology $\langle \mathbb{P}, \mathbb{F}\rangle$. Then, given any set of MAPL facts $\mathbb{D}$ and a MAPL fact φ, decide whether $\langle \mathbb{P} \cup \mathbb{D}, \mathbb{F}\rangle \models \varphi$ holds.*

Our proof of Theorem 5 uses Alternating Turing Machines, the definition of which we repeat here for convenience.

Definition 13. *A* polynomially space-bounded Alternating Turing Machine *(ATM) $\mathcal{M} = \langle \Sigma, Q, q_s, q_f, \Delta\rangle$ is a polynomially space-bounded non-deterministic Turing Machine, where Q is the disjoint union of $Q_\exists$ and $Q_\forall$, the sets of existential and universal states, respectively. A configuration with state $q \in Q$ is accepting if either $q \in Q_\exists$ is an existential state and there is an accepting successor configuration, or if $q \in Q_\forall$ and all successor configurations are accepting. $\mathcal{M}$ accepts if the initial configuration is accepting.*

Theorem 5. *Deciding fact entailment for MARPL ontologies is* ExpTime*-hard, even when restricting to classes of ontologies where only the set of facts is allowed to vary.*

Proof. To show ExpTime-hardness, we reduce from the word problem for a polynomially space-bounded Alternating Turing Machine $\mathcal{M} = \langle \Sigma, Q, q_s, q_f, \Delta\rangle$ with $Q = Q_\exists \cup Q_\forall$, which is a well known ExpTime-hard problem. We assume that $\mathcal{M}$ is such that it has a unique accepting configuration with unique accepting state q_f (e.g., by ensuring that it deterministically deletes the tape and returns to the first position on accepting runs). We further assume that $\mathcal{M}$ has at most two possible transitions in every configuration, i.e., for all $q \in Q$, $\sigma \in \Sigma$, the set $\{\langle q, \sigma, q', \sigma', d\rangle \in \Delta \mid q' \in Q, \sigma' \in \Sigma, d \in \{r, l\}\}$ is of size ≤ 2 (binary branching); this does not reduce the expressive power of ATMs but it simplifies our encoding.

We construct a MARPL ontology $\langle \mathbb{P}, \mathbb{F}\rangle$ that entails a fact $\mathsf{A}()@S$ where S has a specific form if and only if $\mathcal{M}$ accepts on the empty input tape. We simplify notation and omit empty annotations and empty parameter lists, writing, e.g., $\mathsf{A}@S$ for the accepting fact. Our construction will be such that only the facts in $\mathbb{P}$ depend on $\mathcal{M}$, while other rules of $\mathbb{P}$ and function definitions of $\mathbb{F}$ are the same for every ATM.

We assume that $\mathbb{P}$ contains constants $p_0, \ldots, p_\ell$ for representing each of the polynomially many tape positions available to $\mathcal{M}$ (where ℓ depends on $\mathcal{M}$). Moreover, each symbol $\sigma \in \Sigma$ and each state $q \in Q$ is also used as a constant. We assume the following facts to be present in $\mathbb{P}$:

- $\mathsf{univ}(q)$ for every $q \in Q_\forall$,
- $\mathsf{exist}(q)$ for every $q \in Q_\exists$,
- $\mathsf{Tape}@R$ where $R = \{p_i : p_{i+1} \mid 0 \leq i < \ell\}$,
- $\mathsf{TapeTrans}@R$ where $R = \{p_i : p_j \mid 0 \leq i < j \leq \ell\}$, and
- $\mathsf{transA}(q, \sigma, q_1, \sigma_1, d_1)$ and $\mathsf{transB}(q, \sigma, q_2, \sigma_2, d_2)$ for $\langle q, \sigma, q_1, \sigma_1, d_1\rangle \in \Delta$ and $\langle q, \sigma, q_2, \sigma_2, d_2\rangle \in \Delta$ the two transitions available for q and σ, in an arbitrary but fixed order (if there is only one or none, the according facts are simply omitted).

We encode configurations of $\mathcal{M}$ in finite annotations of the following form

$$\{p_0 : \sigma_0, p_1 : \sigma_1, \ldots, p_\ell : \sigma_\ell, \mathsf{head} : p_h, \mathsf{state} : q\}$$

where $p_0, \ldots, p_\ell$ and p_h (the head position) are position constants, $\sigma_0, \ldots, \sigma_\ell \in \Sigma$ are the current symbols at these positions, $q \in Q$ is the current state, and state and head are auxiliary constants.

To encode the word problem, we compute the set of all accepting configurations of $\mathcal{M}$ *backwards*, starting from the unique final configuration, marked by the following fact:

$$\mathsf{A}@\{p_0 : \_, \ldots, p_\ell : \_, \mathsf{head} : p_0, \mathsf{state} : q_f\}$$

where $\_ \in \Sigma$ is the blank tape symbol. Further sets for A are derived recursively using rules. We show the more complicated case of a rule for a universal state, which requires two accepting successor configurations to accept.

The body of the rule is as follows:

$$\mathsf{Tape}@R \wedge \mathsf{TapeTrans}@T \wedge \mathsf{A}@S_1 \wedge \mathsf{A}@S_2 \wedge \tag{19}$$

$$\lfloor \mathsf{head} : x_1^h, \mathsf{state} : x_1^q, x_{\mathrm{prev}}^h : x_1^\sigma \rfloor(S_1) \wedge \tag{20}$$

$$\lfloor x_1^h : x_{\mathrm{prev}}^h \rfloor(R) \wedge \tag{21}$$

$$\lfloor \mathsf{head} : x_2^h, \mathsf{state} : x_2^q, x_{\mathrm{prev}}^h : x_2^\sigma \rfloor(S_2) \wedge \tag{22}$$

$$\lfloor x_2^h : x_{\mathrm{prev}}^h \rfloor(R) \wedge \tag{23}$$

$$\mathsf{transA}(x_{\mathrm{prev}}^q, \sigma_{\mathrm{prev}}, x_1^q, x_1^\sigma, l) \wedge \tag{24}$$

$$\mathsf{transB}(x_{\mathrm{prev}}^q, \sigma_{\mathrm{prev}}, x_2^q, x_2^\sigma, l) \wedge \tag{25}$$

$$\mathsf{univ}(x_{\mathrm{prev}}^q) \tag{26}$$

Part (19) selects two accepting configurations S_1 and S_2, and the tape adjacency relation R. (20) fetches state and head position of S_1 and the symbol x_1^σ at another position x_{prev}^h (intended to be the previous head position), which by (21) is the right neighbour of the head position. (22) and (23) do the same for S_2. (24) and (25) find suitable transitions that might have led to S_1 and S_2 and (26) checks that the state they came from is indeed universal. Note that this body is specific to the case where both accepting configuration are obtained my moving the head to the left. Similar cases are needed for all combinations of left and right movement.

If this body condition is satisfied, then it might be that there is an accepting configuration that can yield the configurations S_1 and S_2 in two according configuration steps. However, we have to be careful, since S_1 and S_2 may not coincide on all tape positions. The according head is simply $\mathsf{A}@\mathsf{Merge}(S_1, S_2, T, x_{\mathrm{prev}}^h, x_{\mathrm{prev}}^q, \sigma_{\mathrm{prev}})$, where the function Merge is defined by the following conditionals:

$$\Rightarrow \mathsf{insert}(\mathsf{head} : x_{\mathrm{prev}}^h, \mathsf{state} : x_{\mathrm{prev}}^q, x_{\mathrm{prev}}^q : \sigma_{\mathrm{prev}}) \tag{27}$$

$$\begin{aligned}&\lfloor x_l : x_{\mathrm{prev}}^h \rfloor(T) \wedge \\ &\quad \lfloor x_l : y \rfloor(S_1) \wedge \lfloor x_l : y \rfloor(S_2) \Rightarrow \mathsf{insert}(x_l : y)\end{aligned} \tag{28}$$

$$\begin{aligned}&\lfloor x_{\mathrm{prev}}^h : x_r \rfloor(T) \wedge \\ &\quad \lfloor x_r : y \rfloor(S_1) \wedge \lfloor x_r : y \rfloor(S_2) \Rightarrow \mathsf{insert}(x_r : y)\end{aligned} \tag{29}$$

Merge therefore defines the head, state and head symbol of the predecessor configuration (27) and copies all tape symbols to the left (28) and to the right (29) of the head tape position, respectively. However, symbols are only copied when they agree between S_1 and S_2. Therefore Merge yields a complete configuration only if this is the case for all non-head positions. Deriving incomplete configurations is not a problem, as they will never lead to a complete configuration that would be relevant for checking acceptance.

This completes the definition of one generic acceptance propagation rule for the case of universal states. The other universal cases (with different head movements) are similar. The case of existential states is simpler: we merely need one accepting successor configuration and do not have to check agreement of two configurations in this case. However, we have to provide multiple rules since either transA or transB might lead to the accepting configuration, in addition to the two possible directions of movement that we need to take into account as before.

With the according rules in place, the resulting ontology is such that it entails the fact $\mathsf{A}@S$ where S encodes the initial configuration iff $\mathcal{M}$ accepts, i.e.,

$$\mathsf{A}@\{p_0 : \_, \ldots, p_\ell : \_, \mathsf{head} : p_0, \mathsf{state} : q_s\}$$

iff $\mathcal{M}$ accepts on the empty input tape. This can be checked directly from the constructions. □

C Proofs for Section 5

Theorem 7. *Deciding fact entailment for MARPL ontologies is in* ExpTime.

Proof. Consider a MARPL ontology $\langle \mathbb{P}, \mathbb{F} \rangle$. Let K be the set of all constants appearing in $\mathbb{P}$ and $\mathbb{F}$. For any annotation S, we have $S \subseteq K \times K$, i.e., there are at most $|\mathcal{P}(K \times K)| = 2^{|K| \times |K|} = 2^{|K|^2}$ distinct annotations. Any MAPL fact φ is of the form $\varphi = p(\boldsymbol{t})@S$, hence there are at most $\sum_{p \in \mathbf{P}} |K|^{\mathsf{ar}(p)} \cdot 2^{|K|^2} \leq |\mathbf{P}| \cdot |K|^{\max_{p \in \mathbf{P}} \mathsf{ar}(p)} \cdot 2^{|K|^2}$ MAPL facts, i.e., exponentially many facts in the size of K. The MARS chase procedure therefore terminates after at most exponentially many iterations.

Any single iteration can also be computed in exponential time. Clearly, a suitable rule and substitution in step A can be found in NP, as can the values of $(t_i\sigma)^{\mathcal{I}}$ in step C. Computing the value of $T^{\mathcal{I}}$ may require the application of a function definition $\langle F, \boldsymbol{S}, C \rangle$. This amounts to checking, for every conditional $c \in C$, whether $\mathcal{I}$ entails the precondition of c. Note that c may contain free object variables. Indeed, for each conditional, let f_c be the number of free object variables in c. There are $|K|^{f_c}$ ground instances of each conditional. Computing $T^{\mathcal{I}}$ therefore requires checking at most $\sum_{c \in C} |K|^{f_c}$, i.e., polynomially many, instances of preconditions. □

For the following statement, we restrict our notion of data complexity to ontologies in MARPL$_k$ by restricting the decision problem in Definition 12 to sets of facts $\mathbb{D}$ for which $\langle \mathbb{P} \cup \mathbb{D}, \mathbb{F} \rangle \in$ MARPL$_k$.

Theorem 8. *For any $k \geq 0$, deciding fact entailment for MARPL$_k$ ontologies is in* PTime *with respect to data complexity, i.e., when considering rules except facts to be fixed.*

Proof. Consider a fixed MARPL ontology $\langle \mathbb{P}, \mathbb{F} \rangle$, and a given set of MAPL facts $\mathbb{D}$, such that $\langle \mathbb{P} \cup \mathbb{D}, \mathbb{F} \rangle \in$ MARPL$_k$. Let K be the set of all constants appearing in $\mathbb{P} \cup \mathbb{D}$ and $\mathbb{F}$. Then, for any entailed MAPL fact with annotation S, we have $|S| \leq k$. As in the proof of Theorem 7, we obtain $\sum_{p \in \mathbf{P}} |K|^{\mathsf{ar}(p)} \cdot 2^{k^2}$ as an upper bound on the number of entailed MAPL facts, which is polynomial if $\langle \mathbb{P}, \mathbb{F} \rangle$ and k are constant.

Note that a single iteration of the MARS chase procedure can now also be done in PTime. In particular, for any rule $\forall \boldsymbol{U}.\forall \boldsymbol{x}.\varphi \rightarrow \psi$, there are at most $|K|^{|\boldsymbol{x}|} \cdot 2^{|\boldsymbol{U}| \cdot k^2}$ candidates for the substitution σ in step 1. Recall that for any MAPL fact, we have $|\boldsymbol{x}| = |\boldsymbol{U}| = 0$, i.e., $\boldsymbol{x}$ and $\boldsymbol{U}$ depend only on $\langle \mathbb{P}, \mathbb{F} \rangle$. Hence, the number of candidates is polynomial in $|K|$. There are polynomially many rules in $\mathbb{P}$, so we can find an applicable rule and a suitable substitution in polynomially many steps. □

Theorem 9. *An ontology $\langle \mathbb{P}, \emptyset \rangle$ is in MARPL$_k$ for every k greater or equal to the largest set term in (facts or rules of) $\mathbb{P}$.*

Proof. Let $\langle \mathbb{P}, \emptyset \rangle$ an ontology in MARPL$_k$. We show that application of rules in $\mathbb{P}$ does not result in facts with annotation size greater than k. Clearly, this holds for all facts in $\mathbb{P}$. Consider a rule $\varphi \rightarrow \psi \in \mathbb{P}$. Then ψ must be of the form $p(t_1, \ldots, t_n)@S$ for some n-ary predicate symbol $p \in \mathbf{P}$, object terms $t_1, \ldots, t_n$, and a set term U. If $S \in \mathbf{U}$ is a set variable, then S occurs in φ, so, by the induction hypothesis, we obtain $|S| \leq k$. Otherwise, $S = \{a_1 : b_1, \ldots, a_\ell : b_\ell\}$ for some object terms a_i, b_i $(1 \leq i \leq \ell)$. Then $|S| = \ell \leq k$, since $\langle \mathbb{P}, \emptyset \rangle$ is in MARPL$_k$. □