PlanetData

Network of Excellence

FP7 – 257641

D31.2 MetaReasons: metatheories list and description

Coordinator: Loris Bozzato, Luciano Serafini (FBK)

1st Quality Reviewer: Giorgos Flouris (FORTH)
2nd Quality Reviewer: Nguyen Quoc Viet Hung (EPFL)

Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	M42
Actual delivery date:	M44
Version:	1.0
Total number of pages:	36
Keywords:	Metatheories list, metadata models

Abstract

In the context of linked data, different models and approaches have been proposed for the management of metadata regarding aspects such as provenance, access control, privacy and trust. In the MetaReasons approach we propose a framework that aims at seamlessly combining and reasoning with several metalevel aspects, each one encoded as a separate "meta-theory" module in the representation of metadata.
Following the previous work in the definition of the MetaReasons architecture, in this deliverable we propose a list of meta-theories representing models that can be included in the framework, for reasoning with metadata for provenance, access control and trust. For each of the models we present its DL based encoding and the extension to the initial sets of rules to reason with the new aspect, together with examples of policies bridging different metatheories. The general representation of metadata provided by MetaReasons allow us to encode additional types of metadata: we present additional metatheories for reasoning with factuality and data quality.

Executive Summary

This deliverable presents the results of the second task (T31.2) of the *MetaReasons* activity (WP31-WP33) in the PlanetData project.

The *MetaReasons* approach has as objective the definition of a unified framework to represent and reason about provenance, access control, privacy and trust meta information of linked data datasets. Activities in this line of work consist of: the definition of the theoretical architecture of the framework and the metatheories encoding different models for the considered aspects (WP31); the implementation of such framework in a working prototype (WP32); the evaluation of the resulting prototype with respect to modelling and scalability criteria (WP33).

Objective of the present task (T31.2, second and last task of WP31) is to provide a study of selected models for provenance, access control and trust aspects, in order to encode and integrate them as metatheories in the proposed architecture, possibly with diverse and helpful representations for such aspects.

In this deliverable, thus, we basically provide a set of metalevel theories (metatheories) encoding different metadata aspects of datasets, including (but not limited to) provenance, access control and trust, that can be imported in MetaReasons architecture in order to reason over such aspects in an unified way.

We first begin by reintroducing the basic ideas and the foundational definitions for the MetaReasons framework together with the associated materialization calculus for reasoning over OWL RL unified knowledge bases.

In the following, each chapter considers a specific metalevel aspect and presents one or more models for its representation. For each model, we will summarize its definition and then propose its encoding as a metalevel theory, together with the additional inference rules to be included in the basic materialization calculus in order to reason about it. By means of examples, we show the possible use and combination of the different metatheories.

The first aspect we consider is *provenance*: we distinguish between two notions of provenance, namely *where provenance* and *how provenance* and we consider a model for each of them. The "where" model we represent is the provenance model introduced in PlanetData deliverable D3.2 (and already considered as initial example metatheory in previous deliverable D31.1), while for representation of "how" provenance we consider the W3C standard PROV-O ontology.

We then consider models for *access control*. We first reintroduce the access control model presented in D31.1 and related to the previous model for provenance. Then, we show how it is possible to include in our framework ROWLBAC, the OWL based encoding of the RBAC access control model.

The following considered metalevel aspect is *trust*: in order to encode a trust metatheory. we considered an available ontology for trust, based on a study about the different elements for trust models representation.

Finally, we show how we can flexibly encode in MetaReasons other metalevel information about datasets. In particular, we present a first theory encoding *data quality* of datasets, using the data quality properties identified in PlanetData deliverable D2.1. We then propose a theory encoding notions of *event factuality* (the effective realization of an event versus its hypothetical or negative occurrence) and *certainty*.

Document Information

IST Project Number	FP7 – 257641	**Acronym**	PlanetData
Full Title	PlanetData		
Project URL	http://www.planet-data.eu/		
Document URL	http://planet-data-wiki.sti2.at/web/D31.2		
EU Project Officer	Leonhard Maqua		

Deliverable	**Number**	D31.2	**Title**	MetaReasons: metatheories list and description
Work Package	**Number**	WP31	**Title**	MetaReasons: Metatheory Development

Date of Delivery	**Contractual**	M42	**Actual**	M44
Status	version 1.0		final ☒	
Nature	Report (R) ☒ Prototype (P) ☐ Demonstrator (D) ☐ Other (O) ☐			
Dissemination Level	Public (PU) ☒ Restricted to group (RE) ☐ Restricted to programme (PP) ☐ Consortium (CO) ☐			

Authors (Partner)	Loris Bozzato, Luciano Serafini (FBK)			
Responsible Author	**Name**	Luciano Serafini	**E-mail**	serafini@fbk.eu
	Partner	Fondazione Bruno Kessler (FBK)	**Phone**	+39 (0461) 314 319

Abstract (for dissemination)	In the context of linked data, different models and approaches have been proposed for the management of meta-data regarding aspects such as provenance, access control, privacy and trust. In the MetaReasons approach we propose a framework that aims at seamlessly combining and reasoning with several metalevel aspects, each one encoded as a separate "meta-theory" module in the representation of metadata. Following the previous work in the definition of the MetaReasons architecture, in this deliverable we propose a list of meta-theories representing models that can be included in the framework, for reasoning with metadata for provenance, access control and trust. For each of the models we present its DL based encoding and the extension to the initial sets of rules to reason with the new aspect, together with examples of policies bridging different metatheories. The general representation of metadata provided by MetaReasons allow us to encode additional types of metadata: we present additional metatheories for reasoning with factuality and data quality.
Keywords	Metatheories list, metadata models

Version Log			
Issue Date	**Rev. No.**	**Author**	**Change**
19/03/2014	0.1	Loris Bozzato, Luciano Serafini (FBK)	First version.
06/05/2014	0.2	Loris Bozzato, Luciano Serafini (FBK)	Addressed feedback from reviewers.
13/05/2014	1.0	Loris Bozzato, Luciano Serafini (FBK)	Final version.

Contents

List of Figures

List of Tables

1 Introduction

In the representation of metadata for management of linked data, several approaches have been proposed to address aspects such as provenance, access control, privacy and trust. In the MetaReasons approach we propose a framework that aims at seamlessly combining and reasoning with several metalevel aspects, each one encoded as a separate "meta-theory" module in the representation of metadata.

In our previous deliverable D31.1 [3] we introduced the framework architecture together with an associated reasoning procedure and two initial metatheories. In this deliverable we propose a list of meta-theories representing models that can be included and combined in our framework, in order to reason with metadata for provenance, access control, trust and additional metalevel aspects.

We summarize in the following the architecture of the MetaReasons framework: an intuitive representation of such architecture is depicted in Figure 1.1. The framework is basically defined over a two layered structure: the upper layer, called *metalevel*, represents the metadata information of datasets while contents of the datasets themselves is represented in the lower layer, called *object level*. In the domain of the metalevel, thus, datasets are seen as atomic individuals and the different meta information is linked to these dataset identifiers. The different metadata aspects (e.g. regarding the provenance and access level of each dataset) are encoded in distinct *metatheories*, metalevel theories that independently describe a particular aspect of the datasets. Each of the metatheories can use its own schema and possibly its own local reasoning formalism: the final representation of the metatheories has however to be reconduced to a single formalism, possibly encoded as one of the OWL2 fragments, in order to enable a unified reasoning over the different aspects. Similarly, formalism and reasoning at object level can be different from dataset to dataset: this modelling allows to decouple reasoning at meta and object level. *Policies*, connecting different metatheories can be defined in the part of the metaknowledge covering all of the single theories for the different aspects. On this architecture, *unified queries* over both the meta-knowledge and the object knowledge can be expressed in standard SPARQL (possibly extended with the primitives defined by the architecture). Such queries can span over the knowledge in the integrated datasets by constraining their properties in the metaknowledge: for example, if we want to retrieve "all of the facts about a certain event, from datasets which have a certain level of trust and for which we have access rights" we can express this in a SPARQL query that selects the datasets matching the meta-level requirements and then run the object-query "all of the facts about a certain event" on the selected datasets.

Considering this framework architecture, in this deliverable we propose a list of meta-theories representing models that can be included in the MetaReasons knowledge bases for reasoning with metadata for provenance, access control and trust. For each of the models we present its DL based encoding and the extension to the initial sets of rules to reason with the new aspect, together with examples of policies bridging different metatheories.

The first aspect we consider is *provenance*: we distinguish between two notions of provenance, namely *where provenance* and *how provenance* and we consider a model for each of them. The "where" model we represent is the provenance model introduced in PlanetData deliverable D3.2 [6] (and already considered as initial example metatheory in previous deliverable D31.1 [3]), while for representation of "how" provenance we consider the W3C standard PROV-O ontology [10].

We then consider models for *access control*. We first reintroduce the access control model presented in D31.1 and related to the previous model for provenance. Then, we show how it is possible to include in our framework ROWLBAC [5], the OWL based encoding of the RBAC access control model.

The following considered metalevel aspect is *trust*: in order to encode a trust metatheory. we considered an available ontology for trust [17], based on a study about the different elements of trust representation in known trust models.

We then show how we can flexibly encode in MetaReasons other metalevel information about datasets: we first propose a theory encoding *data quality* of datasets, using the data quality model and properties identified in PlanetData deliverable D2.1 [11]; finally, we present a metatheory encoding notions of *event factuality* [15] (the effective realization of an event versus its hypothetical or negative occurrence) and certainty.

We remark that the presented metatheories should not be understood as an exhaustive list of encodings for possible models for the considered aspects: instead, we concentrated on encoding few general models that can

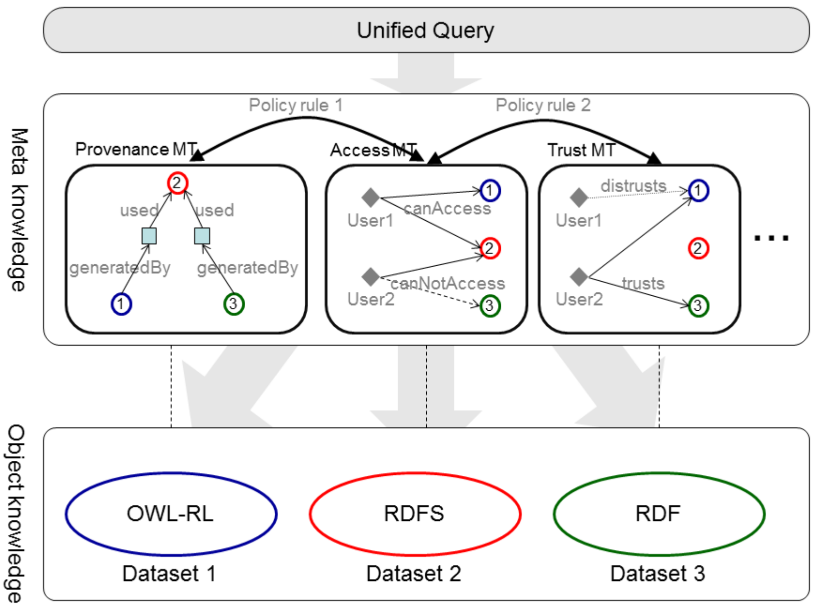

Figure 1.1: MetaReasons architecture.

be representative of the possible encodings of each aspect in our framework (and possibly encode significatively known models, as in the case of metatheories for PROV-O and ROWLBAC).

The deliverable is organized as follows: in the following chapter we introduce some preliminary definitions for language and structure of the MetaReasons framework and the related materialization calculus. We then present the different models and related metatheories for provenance (Chapter 3), access control (Chapter 4), trust (Chapter 5) and further metadata aspects of data quality and event factuality (Chapter 6). Finally, we conclude by giving some outlook on the following phases for implementation of the MetaReasons framework.

2 Preliminaries

In this chapter we reintroduce the basic elements of MetaReasons useful to understand and integrate the following chapters about metalevel theories. In the following, we summarize the basic definitions of the MetaReasons framework and the associated materialization calculus as presented in previous PlanetData deliverable D31.1 [3].

2.1 $\mathcal{SROIQ}$-RL language and normal form

$\mathcal{SROIQ}$-RL language. In the following we assume familiarity with the basic concepts of description logics [1] and with the description logic $\mathcal{SROIQ}$ [8]. We based MetaReasons framework on a restriction of $\mathcal{SROIQ}$ syntax corresponding to OWL RL [13]: we call such language $\mathcal{SROIQ}$-RL. The language is obtained by restricting the form of $\mathcal{SROIQ}$ General Concept Inclusion axioms (GCIs) and concept equivalences as follows: the GCI $C \sqsubseteq D$ is admitted only when C and D are members of two classes of concept expressions called *left-side concept* and *right-side concept*, respectively; the equivalence assertion $E \equiv F$ is admitted only if E and F are both members of the class of left- and right-side concepts. The classes of left- and right-side concepts are defined by the following grammar:

$$C := A \mid \{a\} \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C_1 \mid \exists R.\{a\} \mid \exists R.\top$$
$$D := A \mid \neg C_1 \mid D_1 \sqcap D_2 \mid \exists R.\{a\} \mid \forall R.D_1 \mid \leqslant nR.C_1 \mid \leqslant nR.\top$$

where A is a concept name, R is role name and $n \in \{0,1\}$. A *both-side concept* E, F is a concept expression which is both a left- and right-side concept. TBox axioms can only take the form $C \sqsubseteq D$ or $E \equiv F$. The RBox can contain every role axiom of $\mathcal{SROIQ}$ except for $\mathrm{Ref}(R)$. ABox concept assertions can be only stated in the form $D(a)$, where D is a right-side concept.

Normal form. To facilitate the definition of the calculus, we suppose that $\mathcal{SROIQ}$-RL input axioms are in the following *normal form*:

Table 2.1: $\mathcal{SROIQ}$-RL normal form axioms

$A(a)$	$R(a,b)$	$\neg R(a,b)$	$a = b$	$a \neq b$
$A \sqsubseteq B$	$\{a\} \sqsubseteq B$	$A \sqsubseteq \neg B$	$A \sqcap B \sqsubseteq C$	
$\exists R.A \sqsubseteq B$	$A \sqsubseteq \exists R.\{a\}$	$A \sqsubseteq \forall R.B$	$A \sqsubseteq \leqslant 1 R.B$	
$R \sqsubseteq T$	$R \circ S \sqsubseteq T$	$\mathrm{Dis}(R,S)$	$\mathrm{Inv}(R,S)$	$\mathrm{Irr}(R)$

With $A, B, C \in \mathrm{NC}$ and $R, S, T \in \mathrm{NR}$.

As in [9], we assume that rule chain axioms in input are already decomposed in binary role chains. In deliverable D31.1 [3], we provided a set of rules that transform any $\mathcal{SROIQ}$-RL KB in an "equivalent" KB in normal form. The correctness of this translation can be shown by the following lemma.

Lemma 1. *For every $\mathcal{SROIQ}$-RL knowledge base $\mathcal{K}$ over a signature Σ, a knowledge base $\mathcal{K}'$ over an extended signature Σ' can be computed s.t. (i) all axioms in $\mathcal{K}'$ are in normal form (ii) the size of $\mathcal{K}'$ is a linear factor of the size of $\mathcal{K}$, (iii) for all axioms α in the signature Σ, $\mathcal{K} \models \alpha$ iff $\mathcal{K}' \models \alpha$.* □

2.2 MetaReasons architecture

We summarize in the following the basic definitions of the MetaReasons architecture.

Syntax. A *meta-vocabulary* is a DL vocabulary Γ composed of a set of atomic concepts NC_Γ, a set atomic roles NR_Γ and a set of individual constants NI_Γ that are mutually disjoint and such that:

– $\mathbf{N} \subseteq \mathrm{NI}_\Gamma$ is a set of *dataset names*;

– $\Gamma \supseteq \bigcup_{i \in \{1,\dots,m\}} \Gamma_i$.

Intuitively, each of the Γ_i represent the sub-vocabulary for each of the m metatheories. The *meta-language* denoted by $\mathcal{L}_\Gamma$, is a DL language defined over meta-vocabulary Γ. We may distinguish local languages $\mathcal{L}_{\Gamma_i}$ based on vocabulary subset Γ_i.

We call *object vocabulary* any DL vocabulary $\Sigma = \mathrm{NC}_\Sigma \uplus \mathrm{NR}_\Sigma \uplus \mathrm{NI}_\Sigma$. The object-language $\mathcal{L}_\Sigma$ is the DL language defined from the Σ vocabulary.

Definition 1 (Unified Knowledge Base, UKB). *An* Unified Knowledge Base (UKB) *is a structure* $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ *such that:*

– $\mathfrak{M} = \langle \mathrm{P}, \{\mathrm{MT}_1, \dots, \mathrm{MT}_m\} \rangle$ *is a structure where:*

  – P *is a DL knowledge base over* $\mathcal{L}_\Gamma$ (policy knowledge base)*;*

  – $\mathrm{MT}_1, \dots, \mathrm{MT}_m$ *are DL knowledge bases* (metatheories) *with* MT_i *defined over* $\mathcal{L}_{\Gamma_i}$ *with metavocabulary* $\Gamma_i \subseteq \Gamma$.

– $\mathfrak{D} = \{\, \mathrm{DS}_n \mid n \in \mathbf{N} \,\}$ *where each* DS_n *for* $n \in \mathbf{N}$ *is a DL knowledge base* (dataset) *with* DS_n *defined over* $\mathcal{L}_{\Sigma_n}$ *for an object vocabulary* Σ_n.

An UKB is a $\mathcal{SROIQ}$-RL UKB iff P, every MT_i with $i \in \{1, \dots, m\}$ and every $\mathrm{DS}_n \in \mathfrak{D}$ are $\mathcal{SROIQ}$-RL knowledge bases.

Semantics. Interpretations for an unified knowledge base follow the two layered structure of its components:

Definition 2 (UKB interpretation). *A* UKB interpretation *is a structure* $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ *such that (i)* $\mathcal{M}$ *is a DL interpretation of* Γ *s.t. for any* $n, m \in \mathbf{N}$ *with* $n \neq m$, $n^{\mathcal{M}} \neq m^{\mathcal{M}}$*; (ii) for every* $n \in \mathbf{N}$, $\mathcal{I}(n) = \emptyset$ *or is a DL interpretation over* Σ_n.

Definition 3 (UKB model). *An UKB interpretation* $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ *is a model for an UKB* $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ *(denoted* $\mathfrak{I} \models \mathfrak{K}$*) iff: (i) for every* $\alpha \in \mathfrak{M}$, $\mathcal{M} \models \alpha$*; (ii) for every* $n \in \mathbf{N}$ *and* $\alpha \in \mathrm{DS}_n$*, if* $\mathcal{I}(n) \neq \emptyset$ *then* $\mathcal{I}(n) \models \alpha$.

The classical reasoning problem of entailment can be adapted to UKB models intuitively by referring to a specific dataset (or to the metalevel). Given an UKB $\mathfrak{K}$, $n \in \mathbf{N}$ and an axiom $\alpha \in \mathcal{L}_{\Sigma_n}$, we say that α is *n-entailed* by $\mathfrak{K}$ (denoted $\mathfrak{K} \models n : \alpha$) if, for every UKB model $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ of $\mathfrak{K}$, it holds that $\mathcal{I}(n) \models \alpha$. Similarly, given $\alpha \in \mathcal{L}_\Gamma$, we say that α is entailed by $\mathfrak{K}$ (denoted $\mathfrak{K} \models \alpha$) if, for every UKB model $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ of $\mathfrak{K}$, it holds that $\mathcal{M} \models \alpha$.

2.3 Derivability metatheory MT_d

A common metatheory that we may include in our system (and that is used by some of the metatheories we present in the chapters that follow) is used to reason with the notion of derivation across datasets: considering meta-properties, the derivation of a fact from a set of facts having some meta-properties induces other meta-properties on the derived fact. For instance, the provenance of a fact α derived from facts β and γ depends on the provenance of β and γ. In order to reason with meta-properties of facts, we need a meta-theory for derivability that maintains information on how facts are derived from other facts: we call such meta-theory MT_d. The only predicate of the meta-theory is derivedFrom, a role linking a dataset ds_i with the set of datasets $ds_1, \dots, ds_k$ containing the facts used to infer the facts in ds_i. derivedFrom is declared to be transitive.

We impose the following semantic condition on the interpretation of derivedFrom: all of the consequences derivable (at instance level) from the contents of combining datasets must be contained in the interpretation of the derived dataset. An *instantiation* of an axiom $\alpha \in \mathcal{L}_\Sigma$ with a tuple $\mathbf{e}$ of individuals in NI_Σ, written $\alpha(\mathbf{e})$, is the specialization of α, viewed as its first order translation in an universal sentence $\forall \mathbf{x}.\phi_\alpha(\mathbf{x})$, to $\mathbf{e}$ (i.e., $\phi_\alpha(\mathbf{e})$); accordingly, $\mathbf{e}$ is void for assertions, a single element e for GCIs, and a pair e_1, e_2 of elements for role axioms[1].

[1]In the following we consider Σ_{ds_i} as containing all of the symbols from the signatures $\Sigma_{ds_1}, \dots, \Sigma_{ds_n}$ of the datasets it is derived from.

Given an UKB $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ such that $\mathrm{MT}_d \in \mathfrak{M}$, an UKB interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ is a model for $\mathfrak{K}$ if it is an UKB model and, if $\mathcal{M} \models \mathsf{derivedFrom}(ds_i, ds_j)$ for $j \in \{1, \ldots, k\}$, then either $\mathcal{I}(ds_i) = \emptyset$ or, for every $\alpha \in \mathrm{DS}_j$ and $\mathbf{x}^{\mathcal{I}(ds_i)} \in \Delta^{\mathcal{I}(ds_i)}$, $\mathcal{I}(ds_i) \models \alpha(\mathbf{x})$.

2.4 Basic materialization calculus

Datalog rules. In defining the reasoning of MetaReasons, we follow the approach described in [9], where the inference method is specified by a set of forward inference rules expressed in datalog. In the following we summarize their basic definitions.

A *signature* is a tuple $\langle \mathbf{C}, \mathbf{P} \rangle$, with $\mathbf{C}$ a finite set of *constants* and $\mathbf{P}$ a finite set of *predicates*, each of which is associated with an arity. We assume a set $\mathbf{V}$ of *variables* and we call *terms* the elements of $\mathbf{C} \cup \mathbf{V}$. A *datalog atom* over $\langle \mathbf{C}, \mathbf{P} \rangle$ is in the form $p(t_1, \ldots, t_n)$ with $p \in \mathbf{P}$ and every $t_i \in \mathbf{C} \cup \mathbf{V}$ for $i \in \{1, \ldots, n\}$, and n is the arity of p. A *rule* is an expression in the form $B_1, \ldots, B_m \to H$ where H and $B_1, \ldots, B_m$ are datalog atoms (the *head* and *body* of the rule). A *ground substitution* σ for $\langle \mathbf{C}, \mathbf{P} \rangle$ is a function $\sigma : \mathbf{V} \to \mathbf{C}$. We define as usual substitutions on atoms and *ground instances* of atoms. A ground rule with empty body ($\to H$ or simply H) is called a *fact*. A *program* P is a finite set of datalog rules and facts.

A *proof tree* for P is a structure $\langle N, E, \lambda \rangle$ with $\langle N, E \rangle$ a finite directed tree and λ a labelling function assigning a ground atom to each node, where: for each $v \in N$, there exists a rule $B_1, \ldots, B_m \to H$ in P and a ground substitution σ s.t. (i) $\lambda(v) = \sigma(H)$ and (ii) v has m child nodes w_i in E, with $\lambda(w_i) = \sigma(B_i)$ for $i \in \{1, \ldots, m\}$.

A ground atom H is a *consequence* of P (denoted $P \models H$) if there exists a proof tree for P with root node r and with $\lambda(r) = H$.

Basic materialization calculus. We summarize and slightly generalize the definitions for the materialization calculus for instance checking in the MetaReasons framework proposed in [3]. Differently to the calculus presented in previous deliverable, we consider the case in which the only imported metatheory is MT_d: the additional deduction rules for the other metatheories presented in this deliverable are then introduced together with each metatheory definition in next chapters.

We instantiate and adapt the general definition of *materialization calculus* given by [9] in order to meet the structure of UKBs. The materialization calculus is composed by the sets of *input translations* I_{rln}, I_{meta}, *deduction rules* P_{rln}, $P_{der-obj}$, and *output translation* O, such that: (i) every input translation I and output translation O is a partial function (defined over axioms in normal form) while deduction rules P are sets of datalog rules; (ii) given an axiom or signature symbol α and $n \in \mathbf{N}$, each $I(\alpha, n)$ is either undefined or a set of datalog facts; (iii) given an axiom α and $n \in \mathbf{N}$, $O(\alpha, n)$ is either undefined or a single datalog fact. We extend the definition of input translations to knowledge bases (set of axioms) S with their signature Σ, with $I(S, n) = \bigcup_{\alpha \in S} I(\alpha, n) \cup \bigcup_{s \in \Sigma, I(s,n)\ \text{defined}} I(s, n)$.

$\mathcal{SROIQ}$-RL input I_{rln} and deduction P_{rln} rules provide the translation and interpretation of $\mathcal{SROIQ}$-RL axioms and signature in normal form. Rules of I_{rln} and P_{rln} are shown in Table 2.2. Note that, w.r.t. the translation of [9], we include an additional parameter identifying the knowledge base of reference. Moreover, ABox assertion predicates have to be divided in their "asserted" (`insta`, `triplea`) and "derived" (`instd`, `tripled`) form, in order to recognize explicit ABox assertions inside datasets that need to be propagated to their "derived" datasets.

The output rules O provide the translation of "output" ABox assertions that can be proved by applying the rules of the final program. It is composed of the following rules:

$$\begin{array}{ll} \text{(o-concept)} & A(a) \mapsto \{\mathtt{instd}(a, A, n)\} \\ \text{(o-role)} & R(a, b) \mapsto \{\mathtt{tripled}(a, R, b, n)\} \end{array}$$

The input rules I_{meta} encode rules specifically needed for the translation of the metalevel layer: they include the following rules assuring the distinction in the interpretation of dataset identifiers and to recognize dataset constants in the metavocabulary.

$$\begin{array}{ll}\text{(im-names)} & n1, n2 \in \mathbf{N}, \text{with } n1 \neq n2 \mapsto \{\texttt{neq}(n1, n2, n)\} \\ \text{(im-ds)} & n1 \in \mathbf{N} \mapsto \{\texttt{ds}(n1, n)\}\end{array}$$

We provide a specific set of rules for the derivability metatheory MT_d: these rules in $P_{der-obj}$ define the propagation of knowledge across "derived" datasets that is implied by derivedFrom assertions in the MT_d metatheory. Note that these rules basically encode the same deduction rules from P_{rln} considering in the body one axiom that might appear in a combined dataset with instances appearing in the target dataset: consequences are then propagated in the "composed" dataset. The set of rules for $P_{der-obj}$ are presented in the lower part of Table 2.2.

Translation process. We introduce the notion of entailment by defining the "translation process" to produce a program that represents the complete input UKB. Let $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ be an input UKB in normal form. Then, the *meta program* for its metalevel part $\mathfrak{M}$ is defined as:

$$PM(\mathfrak{M}) = I_{rln}(\mathfrak{M}, m) \cup I_{meta}(\mathfrak{M}, m) \cup P_{rln}$$

with m a new constant identifying the metalevel knowledge base. Intuitively, the meta program translates the contents of policies and all of the metatheories in datalog atoms and includes all of the rule sets needed for reasoning at the meta level: in our case, we only add P_{rln} rules and further metalevel rules may be added by the additional metatheories imported in $\mathfrak{M}$.

For every $n \in \mathbf{N}$, we define the *local program* for DS_n as:

$$PD(n) = I_{rln}(\mathrm{DS}_n, n) \cup P_{rln} \cup P_{der-obj}$$

This means that in every local program we include the translation for local axioms and the rules for local reasoning plus the rules for knowledge propagation to derived datasets. Finally, the *UKB program* for $\mathfrak{K}$ can be encoded as:

$$PK(\mathfrak{K}) = PM(\mathfrak{M}) \cup \textstyle\bigcup_{n \in \mathbf{N}} PD(n)$$

We say that $\mathfrak{M}$ *entails* an axiom $\alpha \in \mathcal{L}_\Gamma$ (denoted $\mathfrak{M} \vdash \alpha$) if $PM(\mathfrak{M})$ and $O(\alpha, m)$ are defined and $PM(\mathfrak{M}) \models O(\alpha, m)$. We say that $\mathfrak{K}$ *entails* an axiom $\alpha \in \mathcal{L}_\Sigma$ in a dataset DS_n with $n \in \mathbf{N}$ (denoted $\mathfrak{K} \vdash n : \alpha$) if the elements of $PK(\mathfrak{K})$ and $O(\alpha, n)$ are defined and $PK(\mathfrak{K}) \models O(\alpha, n)$.

Correctness. We can show that the presented rules and translation process provide a sound and complete calculus for instance checking (with respect to n-entailment) in $\mathcal{SROIQ}$-RL UKB in normal form. A full proof for the result (in particular, for UKBs including additionally metatheories MT_{wp} for provenance and MT_{ac} for access control) can be found in [3].

Theorem 1. *Given $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ a consistent UKB in normal form, $n \in \mathbf{N}$, and $\alpha \in \mathcal{L}_{\Sigma_n}$ an atomic concept or role assertion, then $\mathfrak{K} \vdash n : \alpha$ iff $\mathfrak{K} \models n : \alpha$.* $\square$

Table 2.2: Materialization calculus input and deduction rules

RL input translation $I_{rln}(S,n)$

(irln-nom)	$a \in \mathrm{NI} \mapsto \{\mathtt{nom}(a,n)\}$	(irln-not)	$A \sqsubseteq \neg B \mapsto \{\mathtt{supNot}(A,B,n)\}$
(irln-cls)	$A \in \mathrm{NC} \mapsto \{\mathtt{cls}(A,n)\}$	(irln-subcnj)	$A_1 \sqcap A_2 \sqsubseteq B \mapsto \{\mathtt{subConj}(A_1,A_2,B,n)\}$
(irln-rol)	$R \in \mathrm{NR} \mapsto \{\mathtt{rol}(R,n)\}$	(irln-subex)	$\exists R.A \sqsubseteq B \mapsto \{\mathtt{subEx}(R,A,B,n)\}$
(irln-inst1)	$A(a) \mapsto \{\mathtt{insta}(a,A,n)\}$	(irln-supex)	$A \sqsubseteq \exists R.\{a\} \mapsto \{\mathtt{supEx}(A,R,a,n)\}$
(irln-triple)	$R(a,b) \mapsto \{\mathtt{triplea}(a,R,b,n)\}$	(irln-forall)	$A \sqsubseteq \forall R.B \mapsto \{\mathtt{supForall}(A,R,B,n)\}$
(irln-ntriple)	$\neg R(a,b) \mapsto \{\mathtt{negtriple}(a,R,b,n)\}$	(irln-leqone)	$A \sqsubseteq \leqslant 1 R.B \mapsto \{\mathtt{supLeqOne}(A,R,B,n)\}$
(irln-eq)	$a = b \mapsto \{\mathtt{eq}(a,b,n)\}$	(irln-subr)	$R \sqsubseteq S \mapsto \{\mathtt{subRole}(R,S,n)\}$
(irln-neq)	$a \neq b \mapsto \{\mathtt{neq}(a,b,n)\}$	(irln-subrc)	$R \circ S \sqsubseteq T \mapsto \{\mathtt{subRChain}(R,S,T,n)\}$
(irln-inst2)	$\{a\} \sqsubseteq B \mapsto \{\mathtt{insta}(a,B,n)\}$	(irln-dis)	$\mathrm{Dis}(R,S) \mapsto \{\mathtt{dis}(R,S,n)\}$
(irln-subc)	$A \sqsubseteq B \mapsto \{\mathtt{subClass}(A,B,n)\}$	(irln-inv)	$\mathrm{Inv}(R,S) \mapsto \{\mathtt{inv}(R,S,n)\}$
(irln-top)	$\top(a) \mapsto \{\mathtt{insta}(a,\mathtt{top},n)\}$	(irln-irr)	$\mathrm{Irr}(R) \mapsto \{\mathtt{irr}(R,n)\}$
(irln-bot)	$\bot(a) \mapsto \{\mathtt{insta}(a,\mathtt{bot},n)\}$		

RL deduction rules P_{rln}

(prln-inst)	$\mathtt{insta}(x,z,n) \to \mathtt{instd}(x,z,n)$
(prln-triple)	$\mathtt{triplea}(x,v,y,n) \to \mathtt{tripled}(x,v,y,n)$
(prln-ntriple)	$\mathtt{negtriple}(x,v,y,n), \mathtt{tripled}(x,v,y,n) \to \mathtt{instd}(x,\mathtt{bot},n)$
(prln-eq1)	$\mathtt{nom}(x,n) \to \mathtt{eq}(x,x,n)$
(prln-eq2)	$\mathtt{eq}(x,y,n) \to \mathtt{eq}(y,x,n)$
(prln-eq3)	$\mathtt{eq}(x,y,n), \mathtt{instd}(x,z,n) \to \mathtt{instd}(y,z,n)$
(prln-eq4)	$\mathtt{eq}(x,y,n), \mathtt{tripled}(x,u,z,n) \to \mathtt{tripled}(y,u,z,n)$
(prln-eq5)	$\mathtt{eq}(x,y,n), \mathtt{tripled}(z,u,x,n) \to \mathtt{tripled}(z,u,y,n)$
(prln-eq6)	$\mathtt{eq}(x,y,n), \mathtt{eq}(y,z,n) \to \mathtt{eq}(x,z,n)$
(prln-neq)	$\mathtt{eq}(x,y,n), \mathtt{neq}(x,y,n) \to \mathtt{instd}(x,\mathtt{bot},n)$
(prln-top)	$\mathtt{instd}(x,z,n) \to \mathtt{instd}(x,\mathtt{top},n)$
(prln-subc)	$\mathtt{subClass}(y,z,n), \mathtt{instd}(x,y,n) \to \mathtt{instd}(x,z,n)$
(prln-not)	$\mathtt{supNot}(y,z,n), \mathtt{instd}(x,y,n), \mathtt{instd}(x,z,n) \to \mathtt{instd}(x,\mathtt{bot},n)$
(prln-subcnj)	$\mathtt{subConj}(y_1,y_2,z,n), \mathtt{instd}(x,y_1,n), \mathtt{instd}(x,y_2,n) \to \mathtt{instd}(x,z,n)$
(prln-subex)	$\mathtt{subEx}(v,y,z,n), \mathtt{tripled}(x,v,x',n), \mathtt{instd}(x',y,n) \to \mathtt{instd}(x,z,n)$
(prln-supex)	$\mathtt{supEx}(y,r,x',n), \mathtt{instd}(x,y,n) \to \mathtt{tripled}(x,r,x',n)$
(prln-supforall)	$\mathtt{supForall}(z,r,z',n), \mathtt{instd}(x,z,n), \mathtt{tripled}(x,r,y,n) \to \mathtt{instd}(y,z',n)$
(prln-leqone)	$\mathtt{supLeqOne}(z,r,z',n), \mathtt{instd}(x,z,n), \mathtt{tripled}(x,r,x_1,n),$ $\mathtt{instd}(x_1,z',n), \mathtt{tripled}(x,r,x_2,n), \mathtt{instd}(x_2,z',n) \to \mathtt{eq}(x_1,x_2,n)$
(prln-subr)	$\mathtt{subRole}(v,w,n), \mathtt{tripled}(x,v,x',n) \to \mathtt{tripled}(x,w,x',n)$
(prln-subrc)	$\mathtt{subRChain}(u,v,w,n), \mathtt{tripled}(x,u,y,n), \mathtt{tripled}(y,v,z,n) \to \mathtt{tripled}(x,w,z,n)$
(prln-dis)	$\mathtt{dis}(u,v,n), \mathtt{tripled}(x,u,y,n), \mathtt{tripled}(x,v,y,n) \to \mathtt{instd}(x,\mathtt{bot},n)$
(prln-inv1)	$\mathtt{inv}(u,v,n), \mathtt{tripled}(x,u,y,n) \to \mathtt{tripled}(y,v,x,n)$
(prln-inv2)	$\mathtt{inv}(u,v,n), \mathtt{tripled}(x,v,y,n) \to \mathtt{tripled}(y,u,x,n)$
(prln-irr)	$\mathtt{irr}(u,n), \mathtt{tripled}(x,u,x,n) \to \mathtt{instd}(x,\mathtt{bot},n)$

Derivability object deduction rules $P_{der-obj}$

(ppo-inst)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{insta}(x,y,n1) \to \mathtt{instd}(x,y,nx)$
(ppo-triple)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{triplea}(x,v,y,n1) \to \mathtt{tripled}(x,v,y,nx)$
(ppo-eq)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{eq}(x,y,n1) \to \mathtt{eq}(x,y,nx)$
(ppo-neq)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{neq}(x,y,n1) \to \mathtt{neq}(x,y,nx)$
(ppo-ntriple)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{negtriple}(x,v,y,n1) \to \mathtt{negtriple}(x,v,y,nx)$
(ppo-subc)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{subClass}(y,z,n1), \mathtt{instd}(x,y,nx) \to \mathtt{instd}(x,z,nx)$
(ppo-not)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{supNot}(y,z,n1), \mathtt{instd}(x,y,nx), \mathtt{instd}(x,z,nx) \to \mathtt{instd}(x,\mathtt{bot},nx)$
(ppo-subcnj)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{subConj}(y_1,y_2,z,n1), \mathtt{instd}(x,y_1,nx), \mathtt{instd}(x,y_2,nx) \to \mathtt{instd}(x,z,nx)$
(ppo-subex)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{subEx}(v,y,z,n1), \mathtt{tripled}(x,v,x',nx), \mathtt{instd}(x',y,nx) \to \mathtt{instd}(x,z,nx)$
(ppo-supex)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{supEx}(y,r,x',n1), \mathtt{instd}(x,y,nx) \to \mathtt{tripled}(x,r,x',nx)$
(ppo-supforall)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{supForall}(z,r,z',n1), \mathtt{instd}(x,z,nx), \mathtt{tripled}(x,r,y,nx) \to \mathtt{instd}(y,z',nx)$
(ppo-leqone)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{supLeqOne}(z,r,z',n1), \mathtt{instd}(x,z,nx), \mathtt{tripled}(x,r,x_1,nx),$ $\mathtt{instd}(x_1,z',nx), \mathtt{tripled}(x,r,x_2,nx), \mathtt{instd}(x_2,z',nx) \to \mathtt{eq}(x_1,x_2,nx)$
(ppo-subr)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{subRole}(v,w,n1), \mathtt{tripled}(x,v,x',nx) \to \mathtt{tripled}(x,w,x',nx)$
(ppo-subrc)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{subRChain}(u,v,w,n1), \mathtt{tripled}(x,u,y,nx), \mathtt{tripled}(y,v,z,nx) \to \mathtt{tripled}(x,w,z,nx)$
(ppo-dis)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m),$ $\mathtt{dis}(u,v,n1), \mathtt{tripled}(x,u,y,nx), \mathtt{tripled}(x,v,y,nx) \to \mathtt{instd}(x,\mathtt{bot},nx)$
(ppo-inv1)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{inv}(u,v,n1), \mathtt{tripled}(x,u,y,nx) \to \mathtt{tripled}(y,v,x,nx)$
(ppo-inv2)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{inv}(u,v,n1), \mathtt{tripled}(x,v,y,n2) \to \mathtt{tripled}(y,u,x,nx)$
(ppo-irr)	$\mathtt{tripled}(nx,\mathsf{derivedFrom},n1,m), \mathtt{irr}(u,n1), \mathtt{tripled}(x,u,x,nx) \to \mathtt{instd}(x,\mathtt{bot},nx)$

3 Metatheories for provenance

3.1 Models for provenance

Inside approaches for modelling provenance information, one distinction that can be defined [16] regards whether the model is interested in tracking the source of information or determine the process by which a given information or object has been obtained. The first type of provenance model is called "where provenance", while the latter "how provenance".

In this chapter we present two metatheories for the representation of provenance information: the first encodes the "where provenance" model presented in [7] and in previous Deliverable D31.1, while the second encodes the "how provenance" model proposed by W3C in the PROV-O ontology [10].

3.1.1 Where provenance: Flouris et. al provenance model

The where provenance model that we consider has been introduced in [7]. The idea of this model is to associate single RDF triples to a *color* representing the source of information from which the triple has been obtained. The color of a triple can be stored either as a fourth component of the triple (using *quadruples*), or by grouping all the triples sharing the same color into a *named-graph* [4] and by associating the color as an attribute of the graph. This second alternative turns out to be more convenient, as information about color for different triples can be factorized in a unique property attached to their graph.

By combining triples coming from different data sources it is possible to logically infer new triples: the goal of the model is to automatically compute the color values for the triples inferred from initial colored triples. To this purpose the model allows to combine colors using the binary operator $+$, so that, for instance, the color of a triple derived from a triple colored with c_1 and one colored with c_2, is a new "derived color" denoted with $c_{(1,2)}$. Formally, colors and combinations of colors are modelled as a color structure $(\mathbf{I}, +)$ where: (i) $\mathbf{I}$ is a set of colors; (ii) $+$ is a binary operator of composition over $\mathbf{I}$ that is idempotent, commutative and associative. In our application we assume that $\mathbf{I}$ contains a set of "primitive" colors from which one is able to derive all the colors of $\mathbf{I}$. Intuitively, primitive colors are associated to initial data sources. These assumptions are acceptable in the provenance setting since the set of initial data sources is always bound, and inferred triples that need to be colored can be derived only from triples coming from initial data sources. As a consequence the set $\mathbf{I}$ is isomorphic to the set of non empty subsets of primitive colors. So if $c_1, \ldots, c_k$ is the set of primitive colors, any element of $\mathbf{I}$ can be denoted with c_A where $A \neq \emptyset$ and $A \subseteq \{1, \ldots, k\}$.

The colors c_1 and c_2 that compose $c_{(1,2)} = c_1 + c_2$ are called *defining colors* for $c_{(1,2)}$. Formally, one triple is colored with the composed color $c_{(1,\ldots,n)}$ iff it has been derived from triples colored by the primitive colors $c_1, \ldots, c_n$.

3.1.2 How provenance: PROV-O provenance model

The *PROV Ontology (PROV-O)* [10] is an encoding in the OWL 2 ontology language for the *PROV model* [12]. Goal of the PROV model is to provide a general formalism to encode and exchange provenance information using common formats as RDF and XML. This provenance model has been produced as the result of the effort of the W3C *Provenance Working Group*: the *PROV Data Model (PROV-DM)* corresponds to the core of the proposed provenance model and PROV-O is in fact a serialization of such model. PROV-O has been declared a W3C Recommendation in April 2013. PROV-O defines an encoding of the PROV model as a lightweight ontology: aside from a small set of axioms (cfr. Appendix A of [10]), PROV-O is in the OWL 2 simple profile of OWL RL, in order to favor its application in a wide area of scenarios.

Following the PROV data model, PROV-O is structured on different layers of specialization for the elements of the model. The goal of this definition is to limit the use only to the level of detail of the representation needed in the domain at hand. The model thus classifies its elements in three categories: *starting point* elements, *expanded* terms and terms for relationship *qualification*.

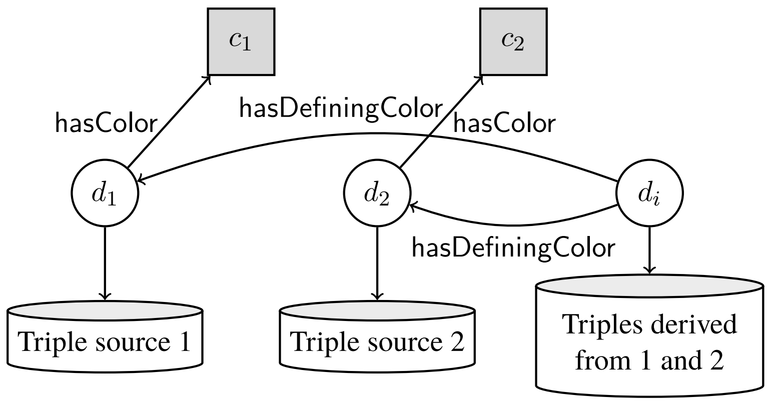

Figure 3.1: Encoding of the provenance model in RDF graphs.

Starting point elements define the basic building blocks for the PROV-O ontology: these classes and properties can be used to define simple descriptions that can be refined by using elements from the other categories. The starting point classes are Entity, that defines physical or conceptual entities manipulated in the domain, Activity, defining events in which domain entities are processed in some way, and Agent, that defines active elements of the domain that may be responsible for the execution of an activity or the existence of an entity. The three classes are related each other using object properties as shown in Figure 3.2. For example, wasAttributedTo links an entity to the agent which is responsible for actions on it, while wasDerivedFrom produces provenance chains composed of entities that describe the transformation of an entity into another.

Expanded elements define additional terms that can be used to refine and relate elements in the starting point category. Many of the terms in this category are defined as subclasses or subproperties of elements in the starting point category. In particular, both classes and properties are refined from the ones presented in the starting elements For example, Agent is specialized in Person, Organization and SoftwareAgent. Similarly, more general and specialized properties are declared e.g. on entities: a generic wasInfluencedBy generalizes wasDerivedFrom, while properties defining subtypes of wasDerivedFrom are introduced as wasQuotedFrom, wasRevisionOf and hadPrimarySource.

Qualified elements add further information over binary relations asserted across starting point or extended elements. The additional attributes in the relations are added following the *Qualification* pattern: an intermediate class representing the influence of the relation is defined and instances of such class can be annotated with the additional descriptions. This basically correspond to a reification of the properties of interactions to introduce further details about such relations. For example, a wasDerivedFrom relation can be reified as a Derivation: the initial entity of the derivation is linked trough a qualifiedDerivation property and the target as the filler of the entity property in the Derivation element.

3.2 Metatheories for provenance

3.2.1 Where provenance: MT_{wp}

We encode the model for "where" provenance described above in a $\mathcal{SROIQ}$-RL meta-theory called MT_{wp}. Differently from [7] we adapt the representation to extend the granularity from a single triple to datasets. By doing this we have to adapt the notion of composition of provenance information by inference: we do so by explicitly adding information about composition of datasets using the MT_d metatheory.

The basic schema of the encoding is shown in Figure 3.1. Nodes are resource names, squares are color names of type Color. Every dataset is associated with an identifier in $\mathbf{N}$: in the depicted case, the initial datasets are identified by d_1 and d_2. Triples which are derived from the combination of multiple resources are stored in a different graph: in the example, the derived dataset is d_i. While initial datasets are assigned an explicit color (c_1 and c_2), the combined color (corresponding to $c_{(1,2)}$) is implicitly represented by the hasDefiningColor relation from the composed dataset to the defining resources.

Syntax. The meta-vocabulary $\Gamma_{wp} = \mathrm{NC}_{wp} \uplus \mathrm{NR}_{wp} \uplus \mathrm{NI}_{wp}$ contains the following symbols:

- Color $\in \mathrm{NC}_{wp}$, class of colors (i.e. provenance sources identifiers);
- hasColor $\in \mathrm{NR}_{wp}$, role linking a source in $\mathbf{N}$ to its color in Color;
- hasDefiningColor $\in \mathrm{NR}_{wp}$, role linking a source in $\mathbf{N}$ to the sources in $\mathbf{N}$ of which colors it is composed;

Language $\mathcal{L}_{wp}$ is a DL language based on Γ_{wp} such that:

- The domains of hasColor and hasDefiningColor are mutually disjoint.
- hasColor is functional.

Intuitively, hasDefiningColor corresponds to the combination operator $+$ of [7].

Semantics. Given an UKB $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ such that $\mathrm{MT}_{wp} \in \mathfrak{M}$, an UKB interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ is a model for $\mathfrak{K}$ if it is an UKB model and: $\mathcal{M} \models \mathsf{derivedFrom}(ds_i, ds_j)$ iff $\mathcal{M} \models \mathsf{hasDefiningColor}(ds_i, ds_j)$.

We remark that differently from [7] we do not ask for an explicit definition of a new color in Color obtained by composition: thus, the set of combining colors can be obtained by following the paths of hasDefiningColor relations until primitive colors are reached.

Inference rules. The set of rules $P_{prov-meta}$ for reasoning with MT_{wp} at the metalevel is composed by the following two rules:

(ppm-comp1) $\mathtt{tripled}(dx, \mathsf{derivedFrom}, dy, m) \rightarrow \mathtt{tripled}(dx, \mathsf{hasDefiningColor}, dy, m)$
(ppm-comp2) $\mathtt{tripled}(dx, \mathsf{hasDefiningColor}, dy, m) \rightarrow \mathtt{tripled}(dx, \mathsf{derivedFrom}, dy, m)$

3.2.2 How provenance: MT_{hp}

We encode as a metatheory MT_{hp} the W3C ontology for provenance PROV-O [10]. As presented in the W3C recommendation, PROV-O can be reduced to the language of OWL RL (Appendix A in [10]), thus is directly representable in $\mathcal{SROIQ}$-RL. This also implies that the materialization calculus we proposed does not need to be extended to reason with the metatheory encoding PROV-O (the rules dealing with normal form $\mathcal{SROIQ}$-RL axioms can directly reason over such representation). However, in the following we link the representation of PROV-O of derived resources to the definition of MT_d of derived datasets, in order to integrate the reasoning in a way compatible to what has been done for MT_{wp}.

Syntax. In order to give an overview of the structure of PROV-O, in the following we summarize the main elements of the PROV-O ontology (corresponding to the "starting point" elements presented in section 3.1 of [10]) by presenting them as a definition for the metatheory MT_{hp}. Of course, this simplification is considered just for explanatory reasons: the secondary and more detailed classes and properties of the ontology can be naturally included in the metatheory as well.

First of all, we note that, when applied in our scenario, it is natural to consider datasets in $\mathbf{N}$ to be included in (but not exclusively equal to) the Entity class.

The meta-vocabulary $\Gamma_{hp} = \mathrm{NC}_{hp} \uplus \mathrm{NR}_{hp} \uplus \mathrm{NI}_{hp}$ contains the following symbols:

- Entity $\in \mathrm{NC}_{hp}$, class of Entities, physical or conceptual objects manipulated in the activities described by the knowledge base; we require that $\mathbf{N} \subseteq$ Entity;
- Activity $\in \mathrm{NC}_{hp}$, class of Activities, events occurring involving the manipulation of entities;
- Agent $\in \mathrm{NC}_{hp}$, class of Agents, individuals responsible for activities on entities;
- wasDerivedFrom $\in \mathrm{NR}_{hp}$, role linking entities to the entities they have been derived from (e.g. by derivation on datasets);
- used $\in \mathrm{NR}_{hp}$, role linking an activity to the entities it used;

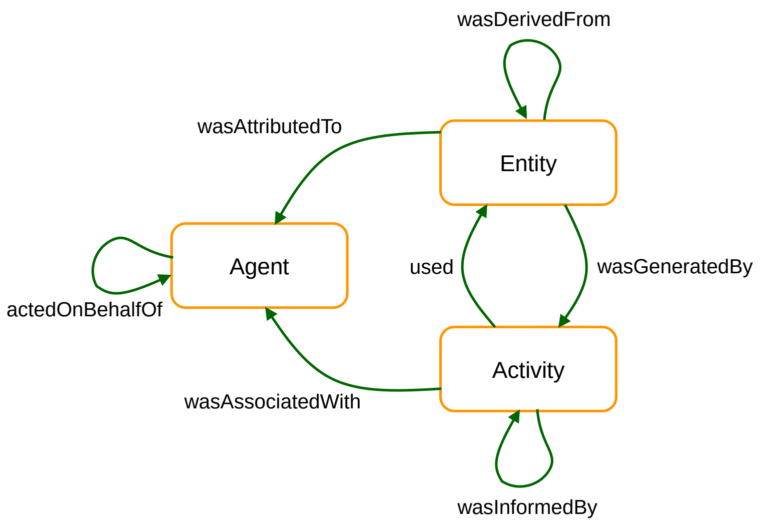

Figure 3.2: PROV-O starting point classes and properties.

– wasGeneratedBy $\in \mathrm{NR}_{hp}$, role linking entities with the activity in which they have been generated;

– wasInformedBy $\in \mathrm{NR}_{hp}$, role linking activities with activities they depend on;

– wasAttributedTo $\in \mathrm{NR}_{hp}$, role linking activities with agents that performed them (i.e. that are responsible for them);

– actedOnBehalfOf $\in \mathrm{NR}_{hp}$, role linking agents to agents that are responsible for the participation on activities;

The intuitive schema of such elements is described in Figure 3.2 (cfr. Figure 1 in [10]).

Language $\mathcal{L}_{hp}$ is a DL language based on Γ_{provo} such that:

– Entity and Agent are disjoint from Activity;

– Domains and range of the properties are defined as informally specified above in Figure 3.2 (e.g. wasDerivedFrom has domain and range in Entity).

We refer to the definition of the "starting point" elements of PROV-O given in Section 4.1 of [10] for the OWL definition of the presented elements and their relation with extended and qualified elements.

Semantics. As we noted above, we only impose the following semantic condition in order to integrate PROV-O with the derivability metatheory MT_d, and thus enable a interaction with the object layer similar to the one of previously presented MT_{wp}.

Given an UKB $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ such that $\mathrm{MT}_{hp} \in \mathfrak{M}$, an UKB interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ is a model for $\mathfrak{K}$ if it is an UKB model and: for $d, d' \in \mathbf{N}$, $\mathcal{M} \models \mathsf{wasDerivedFrom}(d, d')$ iff $\mathcal{M} \models \mathsf{derivedFrom}(d, d')$.

Inference rules. We can then easily extend the calculus to reason with this additional condition by including the following deduction rules in the metalevel program:

$$\begin{array}{ll} \text{(prvm-comp1)} & \mathtt{ds}(dx, m), \mathtt{ds}(dy, m), \mathtt{tripled}(dx, \mathsf{derivedFrom}, dy, m) \rightarrow \mathtt{tripled}(dx, \mathsf{wasDerivedFrom}, dy, m) \\ \text{(prvm-comp2)} & \mathtt{ds}(dx, m), \mathtt{ds}(dy, m), \mathtt{tripled}(dx, \mathsf{wasDerivedFrom}, dy, m) \rightarrow \mathtt{tripled}(dx, \mathsf{derivedFrom}, dy, m) \end{array}$$

3.3 Example

In this section we detail an example of an UKB that includes both metatheories for provenance and we show how the two different aspects of provenance can be combined in reasoning and querying. We basically extend and adapt the example introduced in previous deliverable D31.1 [3] (in turn inspired to the example in [14]) by modelling a system for the management of information about university staff.

Suppose that we have three different datasets:

– d_o, containing the official schema of the university organization, with an assigned provenance color c_1; this dataset has been created by the university system admin ($univ$).

– d_a, containing data about one of the students ($alice$) from the secretary records, with an assigned provenance color c_2; this dataset has been created by the secretary and accepted by $alice$.

– d_b, containing data about one of the professors (bob) from the secretary records, with an assigned provenance color c_2 (the same of previous dataset); this dataset has been created by the secretary and accepted by bob.

We introduce a further dataset d_i that contains the inferences from the combined sources. Thus, in our case $\mathbf{N} = \{d_o, d_a, d_b, d_i\}$.

In the policies, we want to model as visible what is trusted and accepted by the two users $alice$ and bob. In particular we state that:

– *where provenance:* all of the datasets are trusted by the two users. In particular, if a dataset is derived, then it is trusted if it is composed from at least a trusted source.

– *how provenance:* $alice$ and bob data has gone trough an acceptance activity performed by the two (on their own datasets); university organization admin ($univ$) is the creator of d_o, while the secretary (sec) is responsible for the creation of d_a, d_b and d_i.

– *combination:* we include in $alice$ and bob visible datasets the ones that are trusted and accepted.

We can model this situation (together with the information from the datasets) as a $\mathcal{SROIQ}$-RL unified knowledge base $\mathfrak{K}_{ex1} = \langle \mathfrak{M}, \mathfrak{D} \rangle$. The metalevel part $\mathfrak{M} = \langle \mathrm{P}, \{\mathrm{MT}_d, \mathrm{MT}_{wp}, \mathrm{MT}_{hp}\}\rangle$ is defined as:

$$\mathrm{MT}_d = \{\ \mathsf{derivedFrom}(d_i, d_o), \mathsf{derivedFrom}(d_i, d_a), \mathsf{derivedFrom}(d_i, d_b)\ \}$$

$$\begin{aligned}\mathrm{MT}_{wp} = \{\ & \mathsf{hasColor}(d_o, c_1), \mathsf{hasColor}(d_a, c_2), \mathsf{hasColor}(d_b, c_2),\\ & \mathsf{Color}(c_1), \mathsf{Color}(c_2)\ \}\end{aligned}$$

$$\begin{aligned}\mathrm{MT}_{hp} = \{\ & \mathsf{Create} \sqsubseteq \mathsf{Activity}, \mathsf{Accept} \sqsubseteq \mathsf{Activity},\\ & \mathsf{Agent}(alice), \mathsf{Agent}(bob), \mathsf{Agent}(univ), \mathsf{Agent}(sec),\\ & \mathsf{Create}(create\text{-}do), \mathsf{Create}(create\text{-}da), \mathsf{Create}(create\text{-}db), \mathsf{Create}(create\text{-}di),\\ & \mathsf{Accept}(accept\text{-}da), \mathsf{Accept}(accept\text{-}db),\\ & \mathsf{wasAssociatedWith}(accept\text{-}da, alice), \mathsf{wasAssociatedWith}(accept\text{-}db, bob),\\ & \mathsf{wasAssociatedWith}(create\text{-}da, sec), \mathsf{wasAssociatedWith}(create\text{-}db, sec),\\ & \mathsf{wasAssociatedWith}(create\text{-}di, sec), \mathsf{wasAssociatedWith}(create\text{-}do, univ),\\ & \mathsf{used}(create\text{-}do, d_o), \mathsf{used}(create\text{-}da, d_a), \mathsf{used}(create\text{-}db, d_b), \mathsf{used}(create\text{-}di, d_i),\\ & \mathsf{used}(accept\text{-}da, d_a), \mathsf{used}(accept\text{-}db, d_b)\ \}\end{aligned}$$

Then, we can include the following axioms in P to define the policies for where provenance[1]:

$$\begin{gathered}\mathsf{isTrustedColor}(c_1, alice) \quad \mathsf{isTrustedColor}(c_2, alice)\\ \mathsf{isTrustedColor}(c_1, bob) \quad \mathsf{isTrustedColor}(c_2, bob)\\ \exists \mathsf{hasColor}.(\exists \mathsf{isTrustedColor}.\{alice\}) \sqsubseteq \mathsf{TrustedAliceDataset}\\ \exists \mathsf{hasColor}.(\exists \mathsf{isTrustedColor}.\{bob\}) \sqsubseteq \mathsf{TrustedBobDataset}\\ \exists \mathsf{hasDefiningColor}.\mathsf{TrustedAliceDataset} \sqsubseteq \mathsf{TrustedAliceDataset}\\ \exists \mathsf{hasDefiningColor}.\mathsf{TrustedBobDataset} \sqsubseteq \mathsf{TrustedBobDataset}\end{gathered}$$

Similarly, we can introduce this set of policies to describe accepted datasets, using the knowledge from the how provenance metatheory:

$$\begin{gathered}\exists \mathsf{usedBy}.(\mathsf{Accept} \sqcap \exists \mathsf{wasAssociatedWith}.\{alice\}) \sqsubseteq \mathsf{AcceptedAliceDataset}\\ \exists \mathsf{usedBy}.(\mathsf{Accept} \sqcap \exists \mathsf{wasAssociatedWith}.\{bob\}) \sqsubseteq \mathsf{AcceptedBobDataset}\end{gathered}$$

[1]We remark that while in this example we are defining policies for individual users, this modelling can be easily extended to user classes.

where usedBy is defined as the inverse of used. We can simply combine the two notions with the following axioms in the policies:

$$\begin{array}{l}\mathsf{TrustedAliceDataset} \sqcap \mathsf{AcceptedAliceDataset} \sqsubseteq \mathsf{VisibleAliceDataset} \\ \mathsf{TrustedBobDataset} \sqcap \mathsf{AcceptedBobDataset} \sqsubseteq \mathsf{VisibleBobDataset}\end{array}$$

At the object layer, $\mathfrak{D} = \{\mathrm{DS}_o, \mathrm{DS}_a, \mathrm{DS}_b, \mathrm{DS}_i\}$ contains:

– $\mathrm{DS}_o, = \{Student \sqsubseteq Person,\ Professor \sqsubseteq Person,\ Person \sqsubseteq Agent, \dots\}$;

– $\mathrm{DS}_a, = \{Student(alice), \dots\}$;

– $\mathrm{DS}_b, = \{Professor(bob), \dots\}$;

– $\mathrm{DS}_i = \emptyset$.

We can now consider the formal interpretation of $\mathfrak{K}_{ex1}$. A model $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ for $\mathfrak{K}_{ex1}$ must respect the semantic conditions given for models of MT_d, MT_{wp} and MT_{hp}. Note that by these conditions we can derive the following facts about d_i and its associated dataset DS_i:

– In the metalevel, for MT_{wp} we have

$$\mathcal{M} \models \mathsf{hasDefiningColor}(d_i, d_o) \quad \mathcal{M} \models \mathsf{hasDefiningColor}(d_i, d_a) \quad \mathcal{M} \models \mathsf{hasDefiningColor}(d_i, d_b)$$

For MT_{hp} we have that

$$\mathcal{M} \models \mathsf{wasDerivedFrom}(d_i, d_o) \quad \mathcal{M} \models \mathsf{wasDerivedFrom}(d_i, d_a) \quad \mathcal{M} \models \mathsf{wasDerivedFrom}(d_i, d_b)$$

– In the object level, the condition on derivedFrom in MT_d implies that $\mathcal{I}(d_i) \models Person(alice), Agent(alice)$ and $\mathcal{I}(d_i) \models Person(bob), Agent(bob)$.

With respect to policies, it is easy to verify that, from the point of view of $alice$:

$$\begin{array}{ll}\mathcal{M} \models \mathsf{TrustedAliceDataset}(d_o), & \mathcal{M} \models \mathsf{TrustedAliceDataset}(d_a), \\ \mathcal{M} \models \mathsf{TrustedAliceDataset}(d_b), & \mathcal{M} \models \mathsf{TrustedAliceDataset}(d_i), \\ \mathcal{M} \models \mathsf{AcceptedAliceDataset}(d_a) & \mathcal{M} \models \mathsf{VisibleAliceDataset}(d_a)\end{array}$$

and similarly for bob. We remark that such derivable information can be used to obtain a “projection” of the data content of $\mathfrak{K}_{ex1}$ with respect to the visibility of the datasets for a specific user.

4 Metatheories for access control

4.1 Models for access control

In this chapter we encode as metatheories two models for reasoning with access control information for datasets. The first model consists of the approach presented in [14] and already introduced in the MetaReasons framework with the previous PlanetData deliverable D31.1. The second metatheory shows how the well-known encoding of the RBAC model in OWL, ROWLBAC [5], can be easily included in the framework.

4.1.1 Papakonstantinou et al. access model

The first model we encode as our metatheory for access control is presented in [14]. The approach is similar to previous model for provenance given in [7]: the idea is to add access control *tokens* to triples in order to identify their accessibility properties.

The model is said to represent an *abstract* access control model: tokens and operators on them are abstract, in the sense that they model the propagation of access control information and their properties without committing to specific representations of tokens and operations. The idea behind the abstract formulation of the model is that it is desirable to abstract from *how* access control information is computed and only model the formal properties of combination operations: the actual accessibility values can then be computed by defining a suitable mapping to a *concrete* implementation.

As in previous model, the goal of the model is to describe the propagation of access control information from explicit to inferred triples. The access control *label* is represented as the fourth component of quadruples. An initial assignment of access control tokens to labels for the explicit RDF triples can be defined by *access control authorizations*, sets of SPARQL queries over the initial data that provide policies to assign a specific token. Propagation of access control information to inferred triples is defined by the binary operator $\odot$, that is used to compose atomic tokens to complex labels, similarly to the $+$ operator from [7]. Additionally, an unary operator $\otimes$ can be used to propagate the same label to triples representing e.g. instances of a class that are intended to have the same access control properties of the higher layer.

Formally, the abstract access control model is defined as a structure $\mathcal{M} = \langle \mathcal{L}, \perp, \odot, \otimes \rangle$ where: (i) $\mathcal{L}$ is a set of *abstract access tokens* including the *default token* $\perp$, a particular token that is associated to triples not in the scope of an authorization rule; (ii) $\odot$ is the binary *inference operator*; (iii) $\otimes$ is the unary *propagation operator*. The inference operator $\odot$ is commutative and associative, while the propagation operator $\otimes$ is idempotent: $\odot$ is not required to be idempotent since it might be useful to define the case in which $a_1 \odot a_1 \neq a_1$.

4.1.2 ROWLBAC access model

The ROWLBAC ontology has been defined by the need to adapt a known model for access control (*Role Based Access Control*, *RBAC*) to the scenario of Semantic Web data and ontologies. Goal of such encoding is not only to incorporate RBAC in SW data, but also to use an ontology based representation for reasoning and exchange of access control data. The motivation for choosing the RBAC model is due to its diffusion among access control models: support for the different versions of RBAC in OWL can then have immediate application in practical cases. Clearly, this OWL encoding also permit to integrate such representation of access control with a domain ontology representing the application scenario at hand.

In the paper, two different approaches for the translation of RBAC to OWL are proposed: the two approaches basically differ in the representation of roles. However, both solutions can be used to represent the features of different versions of RBAC, that is *flat* RBAC, *hierarchical* RBAC, *constrained* and *symmentric* RBAC [5].

The basic elements of RBAC (actions, subjects and objects) are represented in the same way in both approaches. An *action* is a possible activity that an actor (*subject*) can perform with respect to a specific entity (*object*). A generic Action class represents all possible actions: actions have exactly one subject, instance of the Subject class, and one object, individual of the Object class. The control about such actions is introduced by defining two basic subclasses of Action representing permitted and prohibited actions for a subject, namely

PermittedAction and ProhibitedAction. Naturally, permitted and prohibited actions define a partition for the set of all actions.

As anticipated above, the two approaches differ in the representation of the roles of a subject, that is the possible characterization of a subject (e.g. dataset editor, administrator or generic user). The first approach represents *roles as classes* (subclasses of a class Role) to which individual subjects can belong. Role hierarchies can then be easily introduced by defining class hierarchies across role classes. In particular, in a role class inclusion, the most specific role can represent the dominant role between the two: this basically encodes the fact that going into more specific roles, a subject have more specific privileges. For each role, we have its *active* version (subclass of the ActiveRole class), basically stating that, among all possible roles of a subject, that actor is currently taking the asserted role. Each active role class is linked to the original role class using an activeForm property. A subject can activate a role if it is one of its possible roles: thus each active role class is a subclass of its associated role class. In order to define specific roles in this approaches, these are modelled as follows: each role e.g. Citizen is defined as subclass of Role and a corresponding active role class e.g. ActiveCitizen is defined as subclass of ActiveRole. Role and active role are linked by an activeForm property.

Hierarchical roles can be added by simply defining subclass relations across previously defined roles e.g. if Resident is a role, PermanentResident is a Resident and TemporaryResident is a Resident. Static separation across roles can be simply declared by asserting disjointness between two role classes e.g. Citizen owl:disjointWith Resident. Dynamic separation between roles (which holds between two roles when a subject can not have both simultaneously active) can be declared similarly by asserting disjointness across the active version of the roles. Permissions and prohibitions are simply associated using OWL classes expressions to define classes of permitted and prohibited actions: for example, we can state that only citizens (i.e. individuals that have an active Citizen role) are allowed to vote with the axioms:

$$\mathsf{PermittedVoteAction} \sqsubseteq \mathsf{PermittedAction} \qquad \mathsf{PermittedVoteAction} \equiv \mathsf{Vote} \sqcap \forall \mathsf{subject}.\mathsf{ActiveCitizen}$$

The second and alternative approach for the modelling of roles consist in representing *roles as values*. Roles are defined as elements of a generic Role class and subjects are linked to these roles by two properties, role and activeRole, defining possible and active roles for such actors. In particular, role is an object property connecting elements in subject to elements in Role, while activeRole is defined as a subproperty of role. Hierarchical relation across roles is definable by declaring a suitable subRole property as a transitive relation between two Role individuals. Static and dynamic disjunction of roles can be defined similarly, by introducing symmetric and transitive roles ssod and dsod across different Role elements. Association of roles to permissions can be defined by directly associating role individuals with permitted and prohibited actions: this is encoded using two object properties, namely permitted and prohibited, linking a Role with an Action, e.g. Citizen permitted Vote.

As it is noted in [5], in the first approach standard DL reasoners can be used to answer most of the typical access request queries for subjects and classes of subjects by simply relying on subsumption reasoning. The second approach, while more concise, can not use such mechanism to decide whether a general action is necessarily permitted or not for a given subject. Moreover, both approaches represent a static description of the domain: non-monotonic state changes like role deactivations have to be handled outside OWL reasoning.

4.2 Metatheories for access control

4.2.1 Papakonstantinou et al. model: MT_{ac}

We propose a representation as a metatheory, that we denote with MT_{ac}, for the access control model presented in [14]. We note that the model can be defined almost analogously to the previous metatheory MT_{wp} and it is closely related to the database representation suggested in the original paper. MT_{ac} represents the abstract model from [14], but a concrete model can be defined from it, in particular by fixing concrete values for abstract tokens and mapping to access or deny values for specific classes of users: this information can be easily encoded in the policies knowledge base. In our setting, access control authorizations can be still realized as (SPARQL) queries: on the base of the contents of a dataset, an authorization can assign a particular token to the dataset individual at the metalevel. Moreover, the additional operator for knowledge propagation $\otimes$ can be omitted in our current formulation: we basically assume that this kind of propagation is implicit in the inference "local" to a single dataset (i.e. all of the elements of the dataset share the same label).

Syntax. The meta-vocabulary $\Gamma_{ac} = \mathrm{NC}_{ac} \uplus \mathrm{NR}_{ac} \uplus \mathrm{NI}_{ac}$ contains the following symbols:

– $\mathsf{AccessToken} \in \mathrm{NC}_{ac}$, class of access tokens with default access token $\mathsf{dat} \in \mathrm{NI}_{ac}$;

– $\mathsf{hasLabel} \in \mathrm{NR}_{ac}$, role linking a dataset in $\mathbf{N}$ to its token in $\mathsf{AccessToken}$;

– $\mathsf{isComposedOf} \in \mathrm{NR}_{ac}$, role linking a dataset in $\mathbf{N}$ to the datasets in $\mathbf{N}$ of the tokens it is composed.

Language $\mathcal{L}_{ac}$ is a DL language based on Γ_{ac} such that:

– the domains of $\mathsf{hasLabel}$ and $\mathsf{isComposedOf}$ are mutually disjoint;

– $\mathsf{hasLabel}$ is functional.

Semantics. Given an UKB $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ such that $\mathrm{MT}_{ac} \in \mathfrak{M}$, an UKB interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ is a model for $\mathfrak{K}$ if it is an UKB model and $\mathcal{M} \models \mathsf{derivedFrom}(ds_i, ds_j)$ iff $\mathcal{M} \models \mathsf{isComposedOf}(ds_i, ds_j)$.

Inference rules. The set of rules $P_{ac-meta}$ for reasoning with MT_{ac} at the metalevel is composed by the following rules:

(pam-comp1) $\texttt{tripled}(dx, \mathsf{derivedFrom}, dy, m) \rightarrow \texttt{tripled}(dx, \mathsf{isComposedOf}, dy, m)$
(pam-comp2) $\texttt{tripled}(dx, \mathsf{isComposedOf}, dy, m) \rightarrow \texttt{tripled}(dx, \mathsf{derivedFrom}, dy, m)$

4.2.2 ROWLBAC model: MT_{rbac}

We encode as a metatheory MT_{rbac} the first approach presented in [5] (corresponding to have access control roles represented as classes), but also the second approach (considering access control roles as values) can be encoded similarly.

Syntax. In the following we summarize the elements and definition for the ROWLBAC ontology: in our scenario we note that it is natural to assume that datasets identifiers in $\mathbf{N}$ are to be considered as instances of the Object class.

The meta-vocabulary $\Gamma_{rbac} = \mathrm{NC}_{rbac} \uplus \mathrm{NR}_{rbac} \uplus \mathrm{NI}_{rbac}$ contains the following symbols:

– $\mathsf{Action}, \mathsf{PermittedAction}, \mathsf{ProhibitedAction} \in \mathrm{NC}_{rbac}$, classes representing possible actions in the domain and their subclasses for permitted and prohibited actions.

– $\mathsf{Subject}, \mathsf{Object} \in \mathrm{NC}_{rbac}$, classes representing subject and objects of actions (e.g. actor that accesses a data source and the dataset accessed);

– $\mathsf{Role}, \mathsf{ActiveRole} \in \mathrm{NC}_{rbac}$, classes representing a possible role of an actor and whether this role is currently active.

– $\mathsf{subject}, \mathsf{object} \in \mathrm{NR}_{rbac}$, roles linking actions to their subject and objects.

Language $\mathcal{L}_{rbac}$ is a DL language based on Γ_{rbac} such that:

– $\mathsf{subject}$ and object are functional properties;

– $\mathsf{PermittedAction}$ and $\mathsf{ProhibitedAction}$ are disjoint.

Moreover, we ask that the metatheory MT_{rbac} contains the following axioms:

$$\mathsf{PermittedAction} \sqsubseteq \mathsf{Action} \qquad \mathsf{ProhibitedAction} \sqsubseteq \mathsf{Action} \qquad \mathsf{ActiveRole} \sqsubseteq \mathsf{Role}$$

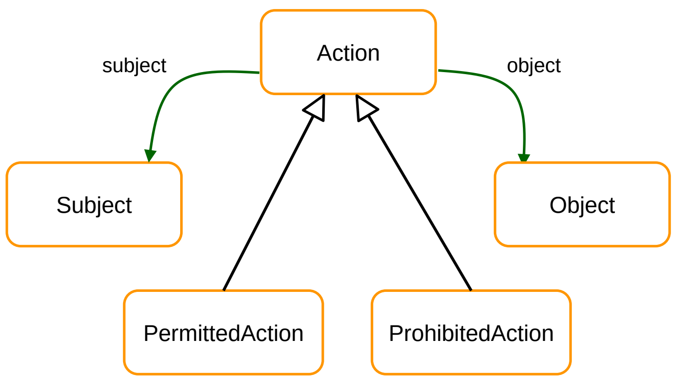

Figure 4.1: ROWLBAC structure about actions.

The intuitive organization of such elements (describing also the concept inclusions and range/domain for each property) is shown in Figure 4.1.

Semantics. Given an UKB $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ such that $\mathrm{MT}_{rbac} \in \mathfrak{M}$, an UKB interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ is a model for $\mathfrak{K}$ if it is an UKB model and for every $n \in \mathbf{N}$, then $n^{\mathcal{M}} \in \mathsf{Object}^{\mathcal{M}}$.

Inference rules. As ROWLBAC is encodable as a OWL RL ontology, we do not have to extend the current rules for $\mathcal{SROIQ}$-RL for reasoning with such metatheory. However we need the following metalevel rule to assure that all the dataset identifiers are included in the Object class.

$$\text{(prbac-objects)} \quad \mathtt{ds}(dx, m) \rightarrow \mathtt{instd}(dx, \mathsf{Object}, m)$$

4.3 Example

We modify our previous example UKB for university staff information in order to combine one of previous provenance theories, MT_{wp}, together with the RBAC metatheory MT_{rbac}. An example of combination for MT_{wp} and MT_{ac} can be found in [3].

Suppose we maintain the same set of datasets from previous example: that is $\mathbf{N} = \{d_o, d_a, d_b, d_i\}$ and the object level knowledge is unchanged.

In the policies, we want now to model that a dataset is visible from the two users $alice$ and bob if they have rights to access it to read its content. In particular w.r.t. *access control* information we want to be able to state that $alice$ and bob have rights to read their own information in d_a and d_b, while $univ$ and sec can edit all of the available datasets.

We can model this situation with the $\mathcal{SROIQ}$-RL UKB $\mathfrak{K}_{ex2} = \langle \mathfrak{M}, \mathfrak{D} \rangle$. The metalevel part $\mathfrak{M} = \langle \mathrm{P}, \{\mathrm{MT}_d, \mathrm{MT}_{wp}, \mathrm{MT}_{rbac}\} \rangle$ now contains the following metatheories:

$$\begin{aligned}
\mathrm{MT}_d &= \{\ \mathsf{derivedFrom}(d_i, d_o), \mathsf{derivedFrom}(d_i, d_a), \mathsf{derivedFrom}(d_i, d_b)\ \} \\
\mathrm{MT}_{wp} &= \{\ \mathsf{hasColor}(d_o, c_1), \mathsf{hasColor}(d_a, c_2), \mathsf{hasColor}(d_b, c_2), \\
&\qquad \mathsf{Color}(c_1), \mathsf{Color}(c_2)\ \} \\
\mathrm{MT}_{rbac} &= \{\ \mathsf{ReadAlice} \sqsubseteq \mathsf{Action}, \mathsf{ReadBob} \sqsubseteq \mathsf{Action}, \\
&\qquad \mathsf{PermittedReadAlice} \sqsubseteq \mathsf{PermittedAction}, \mathsf{PermittedReadBob} \sqsubseteq \mathsf{PermittedAction}, \\
&\qquad \mathsf{Edit} \sqsubseteq \mathsf{Action}, \mathsf{PermittedEdit} \sqsubseteq \mathsf{PermittedAction} \\
&\qquad \mathsf{ActiveAliceReader} \sqsubseteq \mathsf{ActiveRole}, \mathsf{ActiveBobReader} \sqsubseteq \mathsf{ActiveRole}, \mathsf{ActiveEditor} \sqsubseteq \mathsf{ActiveRole} \\
&\qquad \mathsf{ReadAlice} \sqcap \exists \mathsf{subject}.\mathsf{ActiveAliceReader} \sqcap \exists \mathsf{object}.\{d_a\} \sqsubseteq \mathsf{PermittedReadAlice} \\
&\qquad \mathsf{ReadBob} \sqcap \exists \mathsf{subject}.\mathsf{ActiveBobReader} \sqcap \exists \mathsf{object}.\{d_b\} \sqsubseteq \mathsf{PermittedReadBob} \\
&\qquad \mathsf{Edit} \sqcap \exists \mathsf{subject}.\mathsf{ActiveEditor} \sqsubseteq \mathsf{PermittedEdit} \\
&\qquad \mathsf{ActiveAliceReader}(alice), \mathsf{ActiveBobReader}(bob), \mathsf{ActiveEditor}(univ), \mathsf{ActiveEditor}(sec)\ \}
\end{aligned}$$

For simplicity we only model permitted actions, but prohibited actions can be added similarly. We consider axioms in P describing the policies for where provenance to be analogous to previous example:

$$\begin{array}{l}\mathsf{isTrustedColor}(c_1, alice) \quad \mathsf{isTrustedColor}(c_2, alice)\\ \mathsf{isTrustedColor}(c_1, bob) \quad \mathsf{isTrustedColor}(c_2, bob)\\ \exists\mathsf{hasColor}.(\exists\mathsf{isTrustedColor}.\{alice\}) \sqsubseteq \mathsf{TrustedAliceDataset}\\ \exists\mathsf{hasColor}.(\exists\mathsf{isTrustedColor}.\{bob\}) \sqsubseteq \mathsf{TrustedBobDataset}\\ \exists\mathsf{hasDefiningColor}.\mathsf{TrustedAliceDataset} \sqsubseteq \mathsf{TrustedAliceDataset}\\ \exists\mathsf{hasDefiningColor}.\mathsf{TrustedBobDataset} \sqsubseteq \mathsf{TrustedBobDataset}\end{array}$$

Similarly, we can introduce the following set of policies to define read-accessible datasets using the MT_{rbac} metatheory:

$$\begin{array}{l}\exists\mathsf{isObjectOf}.\mathsf{PermittedAliceRead} \sqsubseteq \mathsf{ReadableAliceDataset}\\ \exists\mathsf{isObjectOf}.\mathsf{PermittedBobRead} \sqsubseteq \mathsf{ReadableBobDataset}\end{array}$$

where isObjectOf is defined as the inverse of object. As in our previous example, we can combine as follows the two notions with the following axioms in the policies:

$$\begin{array}{l}\mathsf{TrustedAliceDataset} \sqcap \mathsf{ReadableAliceDataset} \sqsubseteq \mathsf{VisibleAliceDataset}\\ \mathsf{TrustedBobDataset} \sqcap \mathsf{ReadableBobDataset} \sqsubseteq \mathsf{VisibleBobDataset}\end{array}$$

With respect to what is now derivable from $\mathfrak{K}_{ex2}$, consider the situation in which we have a query with a new read request from $alice$: $Q = \{\mathsf{ReadAlice}(read01), \mathsf{subject}(read01, alice), \mathsf{object}(read01, d_a)\}$ and we want to verify that the accessed dataset is visible. It is easy to verify that, by considering $\mathfrak{K}'_{ex2} = \langle \mathfrak{M} \cup Q, \mathfrak{D} \rangle$

$$\begin{array}{ll}\mathfrak{K}'_{ex2} \models \mathsf{TrustedAliceDataset}(d_o) & \mathfrak{K}'_{ex2} \models \mathsf{TrustedAliceDataset}(d_a)\\ \mathfrak{K}'_{ex2} \models \mathsf{TrustedAliceDataset}(d_b) & \mathfrak{K}'_{ex2} \models \mathsf{TrustedAliceDataset}(d_i)\\ \mathfrak{K}'_{ex2} \models \mathsf{ReadableAliceDataset}(d_a) & \mathfrak{K}'_{ex2} \models \mathsf{VisibleAliceDataset}(d_a)\end{array}$$

5 Metatheories for trust

5.1 Models for trust

Among models for the representation of trust in Semantic Web data, we consider the study and ontology proposed in [17]. This work defines a model (then represented as an OWL ontology) that aims at unifying all the different characteristics and notions related to trust, as devised by an analysis of known models. Given its general scope, we chose to encode this model in our framework as it can stand as a good example of how trust metadata (with all of its different factors) can be represented in general.

The idea behind the original work [17] is that there is no single definition of trust and for the aspects related to it: this caused the existence of many different models and formalizations with very different definitions and vocabulary, so that they hardly can be related. This has led the authors of [17] to first survey thirteen relevant computational trust models and classify them with respect to their common factors: using this analysis, a comprehensive ontology is defined in order to ease the collaboration of such distinct models.

The first distinction the paper devises is about the definition and features of *trust*. As there is no unique definition of trust, the work at least identifies the following common properties of different notions of trust. It is concluded that "trust" can be in general represented as a relation that is: not symmetric, not distributive, not associative and not inherently transitive.

To compile the typical features of trust models (in order to encode them as an ontology) several models of trust (from a significative selection of works) have been considered and classified using the trust factors they take into consideration. In the following we summarize the factors that have been identified and used for the classification of models.

First of all, all reviewed models have a notion of *identity* of the individuals (or *principals*) involved in the trust evaluations. Identity has different declinations: some of the models only require a locally unique identity which may be maintained for an amount of time sufficient for the evaluation.

Most models consider trust with respect to a specific *action*. In other words, trust relation across two individuals is the trust in the action an individual wants to perform. Action component is not always necessary: this is the case of *general trust*, in which trust is basically only a binary relation across one trustor and trustee (i.e. only depends on their features).

One of the reasons for representing trust is that it is important inside one interaction to assess the potential risks and benefits of such exchanges. Thus, in different models we can find concepts called risk, benefit or value: all of these factors are associated with an action and provide a quantification of profits and losses in its successful or unsuccessful execution. Note that different notion of risk can be associated to the failure of an action, for example the "risk of not getting an information". In the presented model, these notions are combined under the concept of *business value*, since all these features provide a value for the impact of the action (positive or not). The model then distinguish between *benefit*, *risk* and *importance* of an action as subtypes of the business value.

Another decision factor in trust is the knowledge about the *competence* of a principal in performing a particular action. However, such aspect is not very common, as in the case of automated agents it is of difficult assessment. One way to represent this is a measure of its experience, that is the principal competence in providing relevant information.

Some of the models require a token that demonstrates the access and authority to perform an action. In the field of security such token is also called a *capability* (which in particular defines a token assigned to a peer to access a resource): capability is introduced in the presented model using this particular meaning.

The factor of *confidence* models the level of certainty of the source of trust information and input factors. This factor can take different forms as a confidence value in the cooperability with the trust principal or as a certainty or rating in the source for input factors.

Trust computation may depend on the particular situation (*context*) in which it is evaluated. Similarly, there can be a notion of reputation across principals or in general *history*. This basically corresponds to a record of previous interaction between principals. Such form of reputation appear in models in two forms: subjective reputation, when this is computed by trustors, or external reputation, when such reputation is obtained from a third party. In the model, history represents subjective reputation.

Trust models are said to be *open models* if trust information from a *third party* is involved in trust computation. Such information can take different forms: one of these is, as discussed above, an external *reputation*, that is an external version for the interaction history. Otherwise, third party can provide a form of *recommendation* as an influencing factor. Similarly, it can provide information in the form of *credentials*, regarding a particular subject and his capabilities.

5.2 Metatheory for trust: MT_{tr}

Using the previous considered factors, in [17] a model (encoded as an OWL ontology) is then presented in order to give a unified vision of all such trust features.

In the following, we encode this model (based on the UML model presented in the original paper) as a metatheory MT_{tr} that can be imported in our framework. First of all, we note that in our scenario, differently from other aspects, we may not restrict the notion of "the object of trust" solely to datasets: depending on the domain to be modelled, trust might be differently connected to the other aspects e.g. of provenance and access classes in order to represent, for instance, the trust of a particular information source or of an action for a particular class of users. Thus we do not classify dataset identifiers in $\mathbf{N}$ e.g. as trustee, but we leave this possibility to the modeller, that can enforce different notions of trust by means of external policies.

In the following we thus encode the previously considered factors as a metalevel theory: note that *identity* in our case is naturally represented by the individual name for principals. In the implementation such unique identifier is in fact realized by (unique) URIs: if deemed necessary, a further (datatype) property associating an identifier to such elements can be introduced.

On the other hand, following the original model, most of the factors are represented as individuals of a particular class e.g. History or Capability: concrete values and interpretation for such elements are again dependent from the notion of trust to be modelled and the domain at hand.

The meta-vocabulary $\Gamma_{tr} = \mathrm{NC}_{tr} \uplus \mathrm{NR}_{tr} \uplus \mathrm{NI}_{tr}$ contains the following symbols:

- Principal, Trustor, Trustee $\in \mathrm{NC}_{tr}$, classes representing individuals involved in the trust representation and their role;
- Trusts $\in \mathrm{NC}_{tr}$, classes representing the reification of the trust relation across trustor and trustee; hasTrustor, hasTrustee $\in \mathrm{NR}_{tr}$ link elements in Trusts to the two involved principals.
- Action $\in \mathrm{NC}_{tr}$, class representing actions related to the trust relation; hasAction $\in \mathrm{NR}_{tr}$ links a trust relation to its action.
- BusinessValue, Risk, Benefit, Importance $\in \mathrm{NC}_{tr}$, classes representing business value of an action and their subclasses; hasRisk, hasBenefit, hasImportance $\in \mathrm{NR}_{tr}$ link an action to specific business values.
- Competence, Capability, Confidence $\in \mathrm{NC}_{tr}$, class representing competence, capability and confidence factors of a trust relation; hasCompetence, hasCapability, hasConfidence $\in \mathrm{NR}_{tr}$ link a trust relation to values in such factors.
- Context, History $\in \mathrm{NC}_{tr}$, classes representing context and history values of a trust relation; hasContext, hasHistory $\in \mathrm{NR}_{tr}$ link a trust relation to values in such factors.
- ThirdPinfo, Reputation, Recommendation, Credential $\in \mathrm{NC}_{tr}$, classes representing third party information of a trust and their subtypes; hasThirdPinfo $\in \mathrm{NR}_{tr}$ links a trust relation to values in such factors.

Language $\mathcal{L}_{tr}$ is a DL language based on Γ_{tr} such that: hasTrustor, hasTrustee, hasAction, hasRisk, hasBenefit, hasImportance, hasCompetence, hasConfidence, hasContext are functional properties. Moreover, we ask that the metatheory MT_{tr} contains the following axioms:

$$\mathsf{Trustor} \sqsubseteq \mathsf{Principal} \qquad \mathsf{Trustee} \sqsubseteq \mathsf{Principal}$$
$$\mathsf{Risk} \sqsubseteq \mathsf{BusinessValue} \quad \mathsf{Benefit} \sqsubseteq \mathsf{BusinessValue} \quad \mathsf{Importance} \sqsubseteq \mathsf{BusinessValue}$$
$$\mathsf{Reputation} \sqsubseteq \mathsf{ThirdPinfo} \quad \mathsf{Recommendation} \sqsubseteq \mathsf{ThirdPinfo} \quad \mathsf{Credential} \sqsubseteq \mathsf{ThirdPinfo}$$

The intuitive organization of such elements is shown in Figure 5.1 (which basically corresponds to the UML model presented in Figure 2 of [17]).

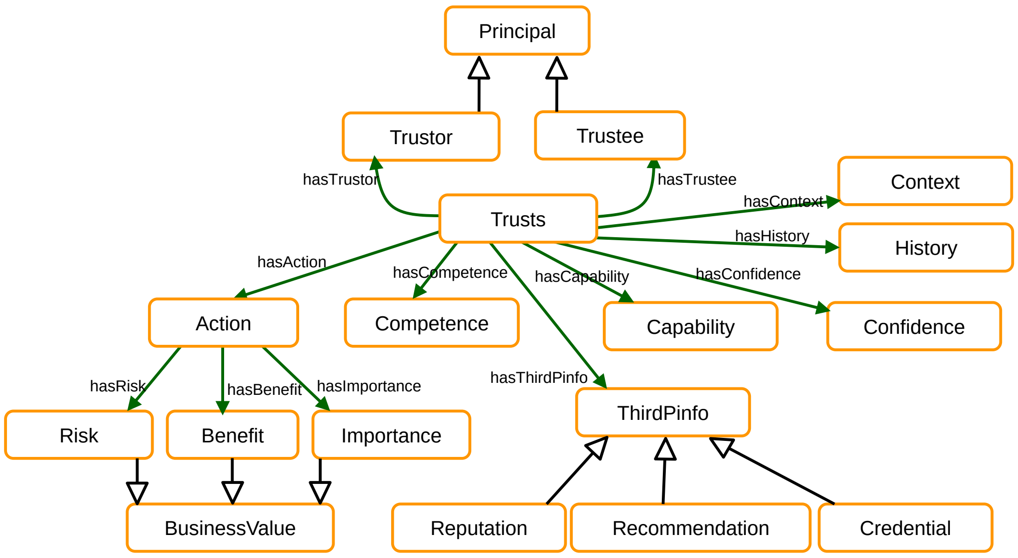

Figure 5.1: Trust ontology structure.

5.3 Example

We can consider another modification to our previous example UKB in order to combine the presented contents of the RBAC metatheory MT_{rbac} with an instantiation of the trust metatheory MT_{tr} (substituting the previous notion for trust provided by the where provenance model).

Suppose that in the policies, we want now to model that the datasets contained in the UKB are trusted by $alice$ and bob because these have been published by the official sources. In particular w.r.t. *trust* information we want to state that $alice$ and bob trust $univ$ and sec in the action of publishing the datasets in $\mathbf{N}$.

We can model this situation with the unified knowledge base $\mathfrak{K}_{ex3} = \langle \mathfrak{M}, \mathfrak{D} \rangle$ containing in the metalevel part $\mathfrak{M} = \langle \mathrm{P}, \{\mathrm{MT}_d, \mathrm{MT}_{tr}, \mathrm{MT}_{rbac}\}\rangle$ the following metatheory:

$$\begin{aligned}
\mathrm{MT}_{tr} = \{\ & \mathsf{Trustor}(alice), \mathsf{Trustor}(bob),\\
& \mathsf{Trustee}(univ), \mathsf{Trustee}(sec),\\
& \mathsf{Trusts}(alice\text{-}trusts\text{-}sec\text{-}da), \mathsf{Trusts}(alice\text{-}trusts\text{-}sec\text{-}db), \mathsf{Trusts}(alice\text{-}trusts\text{-}sec\text{-}di),\\
& \mathsf{Trusts}(bob\text{-}trusts\text{-}sec\text{-}da), \mathsf{Trusts}(bob\text{-}trusts\text{-}sec\text{-}db), \mathsf{Trusts}(bob\text{-}trusts\text{-}sec\text{-}di),\\
& \mathsf{Trusts}(alice\text{-}trusts\text{-}univ\text{-}do), \mathsf{Trusts}(bob\text{-}trusts\text{-}univ\text{-}do)\\
& \mathsf{Action}(publish\text{-}do), \mathsf{Action}(publish\text{-}da), \mathsf{Action}(publish\text{-}db), \mathsf{Action}(publish\text{-}di),\\
& \mathsf{hasAction}(alice\text{-}trusts\text{-}univ\text{-}do, publish\text{-}do), \mathsf{hasAction}(bob\text{-}trusts\text{-}univ\text{-}do, publish\text{-}do),\\
& \mathsf{hasAction}(alice\text{-}trusts\text{-}sec\text{-}da, publish\text{-}da), \mathsf{hasAction}(alice\text{-}trusts\text{-}sec\text{-}db, publish\text{-}db),\\
& \mathsf{hasAction}(alice\text{-}trusts\text{-}sec\text{-}di, publish\text{-}di), \mathsf{hasAction}(bob\text{-}trusts\text{-}sec\text{-}da, publish\text{-}da),\\
& \mathsf{hasAction}(bob\text{-}trusts\text{-}sec\text{-}db, publish\text{-}db), \mathsf{hasAction}(bob\text{-}trusts\text{-}sec\text{-}di, publish\text{-}di)\ \}
\end{aligned}$$

In the policies we can define the relations between the publication action of a dataset and the dataset itself, similarly to the access control metatheory MT_{rbac}:

$$\mathsf{object}(publish\text{-}do, d_o) \quad \mathsf{object}(publish\text{-}da, d_a) \quad \mathsf{object}(publish\text{-}db, d_b) \quad \mathsf{object}(publish\text{-}di, d_i)$$

We consider axioms in P that describe the policies for access control (i.e. defining ReadableAliceDataset and ReadableBobDataset) as analogous to previous example. We can then introduce the following set of policies to define trusted datasets using the MT_{tr} metatheory.

$$\begin{array}{l} \exists \mathsf{isObjectOf}.(\mathsf{Action} \sqcap \exists \mathsf{isActionOf}.(\exists \mathsf{hasTrustor}.\{alice\})) \sqsubseteq \mathsf{TrustedAliceDataset} \\ \exists \mathsf{isObjectOf}.(\mathsf{Action} \sqcap \exists \mathsf{isActionOf}.(\exists \mathsf{hasTrustor}.\{bob\})) \sqsubseteq \mathsf{TrustedBobDataset} \end{array}$$

As above, we can then combine the two notions with these policies:

$$\begin{array}{l} \mathsf{TrustedAliceDataset} \sqcap \mathsf{ReadableAliceDataset} \sqsubseteq \mathsf{VisibleAliceDataset} \\ \mathsf{TrustedBobDataset} \sqcap \mathsf{ReadableBobDataset} \sqsubseteq \mathsf{VisibleBobDataset} \end{array}$$

Is it easy to verify that the consequences derivable from $\mathfrak{K}_{ex3}$ are analogous to our previous example.

6 Metatheories for further aspects

In this chapter we briefly introduce two metatheories encoding in the MetaReasons framework two further aspects for dataset metadata: the first encodes a notion of quality of datasets, while the second allows to differentiate datasets on the base of the factuality and certainty of their contents.

6.1 Metatheory for data quality

6.1.1 Model for data quality

The model we want to encode as a representation for data quality is the set of data quality measures proposed in PlanetData deliverable D2.1 [11] regarding a conceptual model and best practices for assessing the quality of linked data and data streams. Deliverable D2.1 provides a model and several measures and indicators about the quality of linked data in order to establish the reliability of a given set of data (possibly w.r.t. a given task).

The foundational idea of the presented model is that of “fitness for use”: there is not a single fixed method to assess the quality of information, but a wide set of policies can be defined on the base of the task at hand. That is, the interpretation of “quality” of data depends on the user of the data and the task in which such data might be used for. The presented model is thus *multi-faceted*: it provides values for the different dimensions of quality that the user might take into consideration in its evaluation for the fitness of a particular piece of information.

We note that this model fits naturally in our framework: MetaReasons shares with this model the idea of having different metalevel dimensions that can be used in policy reasoning and filtering in a unified way. We can also say that MetaReasons follows the “fitness to use” idea, since the modeller can impose different policies over and across the imported metatheories.

The model proposed in D2.1 aims at unifying the several quality dimensions studied in literature and be flexible enough to accomodate new assessment metrics: the model is composed of indicators, scoring functions and assessment metrics.

A *Data Quality Indicator* is an feature of a dataset that can be used in the assessment of the quality of dataset contents. Indicators can depend on the data itself, the data provider, external ratings or background information on the dataset.

A *Scoring Function* is a function assessing a data quality indicator and can be used by a user in deciding the suitability of the data for the intended task. Scoring functions depend on the value type of the assessed indicator and can vary from simple comparisons, to set functions or statistical functions.

Assessment Metrics are operators or procedures to measure an information quality dimension: they are based and combine different quality indicators and provide an assessment score using a scoring function. Different metrics can be define to measure a specific dimension.

In the following we review the instantiation of the model for linked data datasets presented in Section 2.1 of [11] and discuss how it can be represented as a MetaReasons metatheory.

6.1.2 Metatheory for data quality: MT_{dq}

In this section we propose how to encode the presented set of indicators and metrics of the model instantiation to linked datasets as a metatheory MT_{dq}.

First of all, we remark that since the proposed metrics are basically defined as numeric operations over indicators, such metrics can not be directly computed using logical reasoning provided by our DL based representation: thus with this encoding we only propose a way to represent these elements and results of their (external) computations.

Note also that (the value of) metrics associated to a dataset are naturally representable using OWL datatype properties: in order to formally include datatype properties in our formalization and be able to reason with them, we should introduce datatypes in $\mathcal{SROIQ}$-RL. In this first formalization of MT_{dq}, we consider datatype values as individuals, thus mapping these properties to DL roles. We note however that in an implementation of the framework based on RDF like the one we will implement in the activities of WP32 (and we briefly discuss in the conclusions) these can be easily represented: moreover, as we plan to represent deduction rules using SPARQL queries, these can be easily adapted to extend policies e.g. to exclude datasets with a metric inferior to a specified value.

Accessibility. Accessibility represent measures that describe the ability to get access to datasets.

Access Methods indicators define different possible means to access the dataset: These can be encoded as dataset classes Sparql, Bulk, $\mathsf{Sample} \in \mathrm{NC}_{dq}$: they represent the possibility to access a SPARQL endpoint, bulk files and example sources for a dataset. Using these indicators, an assessment metric for *RDF accessibility*, defined as weighted sum of the accessibility methods (which weights can be assigned by the user). This metric can be encoded as the role $\mathsf{RDFAccessibility} \in \mathrm{NR}_{dq}$ linking a dataset in $\mathbf{N}$ to its (individual) value for the computed metric.

Another notion is that of *reachability*: for a given dataset, we can measure the number of links going from a dataset to another and then evaluate the number of incoming and outgoing triples (in and outdegree of the dataset). These indicators can be encoded with a reification of the external link indicator $\mathsf{ExternalLinks} \in \mathrm{NC}_{dq}$, with roles $\mathsf{source}, \mathsf{target}, \mathsf{value} \in \mathrm{NR}_{dq}$ to encode the two related datasets and the value of the indicator, and two roles $\mathsf{indegree}, \mathsf{outdegree} \in \mathrm{NR}_{dq}$ to encode the in and out degree of a given dataset in $\mathbf{N}$. The two proposed metrics for in-degree reachability and PageRank reachability of a dataset can be represented using roles $\mathsf{LDSInDegreeReachability}, \mathsf{LDSPageRankReachability} \in \mathrm{NR}_{dq}$.

Availability can be defined as the percentage of time a given service (i.e. in our case, data providing service) is able to provide a response when it receives a request. The indicators for such measure are representable as classes $\mathsf{HttpGet}$, $\mathsf{SparqlQuery}$, $\mathsf{HttpHead} \in \mathrm{NC}_{dq}$, representing the availability of the resource with an HTTP get request, a SPARQL query or an HTTP head request. The model also proposes singular metrics *avail_sample*, *avail_sparql* and *avail_bulk* which provide a measure of the availability of different access methods in a given time interval. Similarly to accessibility, we can define a general metric for availability that evaluates the previous indicators as a role $\mathsf{LDSAvailability} \in \mathrm{NR}_{dq}$, linking a dataset in $\mathbf{N}$ to its value for the weighted sum of previous availability indicators.

Differently from availability, *response time* regards the time within which an access method can provide with the requested data. The indicators for such measure are representable as roles $\mathsf{timeSparql}$, $\mathsf{timeSample}$ $\mathsf{timeBulk} \in \mathrm{NR}_{dq}$. They represent the response time of the resource with a SPARQL query or a request for a sample or bulk data. An assessment metric $\mathsf{LDSResponseTime} \in \mathrm{NR}_{dq}$ can be defined linking a dataset in $\mathbf{N}$ to its computed metric: as in previous metrics, it is left to the user to configure the importance of the response time for each access method.

Robustness defines the capability of a provider to respond to a number of data requests: it can be measured from the data provider itself or estimated by users using the response time metrics. We can add to our metatheory this measure as a role $\mathsf{LDSRobustness} \in \mathrm{NR}_{dq}$, assessing the number of requests supported in a given time period.

Interpretability and understandability. Different measures for interpretability of data are defined, useful to assess the capability of the dataset interoperability.

First of all about*format interpretability*, we have different indicators depending on the serialization in which a dataset is available: we can encode them as the dataset classes RdfXml, Turtle, $\mathsf{Ntriples}$, $\mathsf{Nquads} \in \mathrm{NC}_{dq}$ A metric for interoperability can be introduced similarly to previous accessibility metrics as $\mathsf{LDSFormatInterpretability} \in \mathrm{NR}_{dq}$ representing a linear combination of the indicators.

Human+Machine Interpretability refers to the availability of versions of the dataset that are interpretable by machines or readable by humans. Different versions of the same dataset can be published using different formats for the consumption by humans and machines by dereference mechanism. We specialize $\mathsf{HttpGet}$ to represent the availability of dereferenced different verions for the resource: $\mathsf{HttpGetHTML}$, $\mathsf{HttpGetXHTMLRDFa}$, $\mathsf{HttpGetRDFXML}$, $\mathsf{HttpGetNTriples}$, $\mathsf{HttpGetNQuads}$, $\mathsf{HttpGetTurtle} \in \mathrm{NC}_{dq}$ and all such concepts are defined as included in of $\mathsf{HttpGet}$ in MT_{dq}. Further classes can be added to represent additionalMetaReasons formats. Combining these indicators (as a weighted sum), a metric $\mathsf{LDSHumanMachineInterpretability} \in \mathrm{NR}_{dq}$ can be introduced.

Another metric refers to *vocabulary understandability*: the idea is that a relevant factor in understanding the data is to know the vocabulary and schema used for its definition. Thus a dataset that uses a known schema is more likely to be understood by its users. We can include the reification of the proposed indicator to address the number of triples of a dataset that appear as part of a given vocabulary: $\mathsf{TriplesInVocabulary} \in \mathrm{NC}_{dq}$ again using roles $\mathsf{source}, \mathsf{vocabulary}, \mathsf{value} \in \mathrm{NR}_{dq}$ to encode the dataset, the reference vocabulary and the value of the indicator. With this, an assessment of the understanding of the dataset by the evaluating agent can be defined with the metric $\mathsf{LDSVocabularyUnderstandability} \in \mathrm{NR}_{dq}$.

With respect to *internationalization*, we can introduce a metric quantifying the portion of the dataset that is labelled with a particular language tag: this metric can be represented again as a reified relation $\mathsf{LDSIntUnderstandability} \in \mathrm{NC}_{dq}$ using roles $\mathsf{source}, \mathsf{languageLabel}, \mathsf{value} \in \mathrm{NR}_{dq}$ to encode the datasets, the language tag and the value of the measure.

Timeliness. Timeliness regards the level of update of a knowledge source: it can be divided in *freshness*, the measure of time from the last update to the resource, or *newness*, measuring the time in which the resource has been created. Both metrics can be also interpreted "contextually", by evaluating both metrics with respect to a specific task. The two measures can be quantified in seconds from the last modification or creation of a resource (obviously provided by some external computation): we thus only represent the two metrics as the roles $\mathsf{LDSNewness}, \mathsf{LDSFreshness} \in \mathrm{NR}_{dq}$. Another encoding option can consist in representing only the information about the creation and last modification date of each dataset and leave the computation of freshness and newness to the user queries.

Openness. Openness refers to the availability of a resource under an open usage license. We can represent this with a generic class $\mathsf{isOpen} \in \mathrm{NC}_{dq}$. Alternatively, we can introduce a role from a dataset to its license (or set of) with a role $\mathsf{licence} \in \mathrm{NR}_{dq}$, possibly classifying as open or closed the target licenses.

Verifiability. Under verifiability the model groups measures for assessing the possibility to check the data for correctness.

Traceability refers to the ability to follow the history of a dataset, together with its processes and related actors involved in its creation and publishing. The model in [11] does not explicitly provide a measure for traceability: in fact, we note that reasoning on traceability can be performed e.g. using the proposed metatheory MT_{hp} for PROV-O. Thus, we may only introduce a class for datasets $\mathsf{isTraceable} \in \mathrm{NC}_{dq}$, representing the fact that the dataset has some provenance information in the metalevel.

Accountability refers to the possibility to account some entity for a set of data. Similarly to traceability, we can choose to represent this information using the provenance metatheories: otherwise we can again introduce a class $\mathsf{isAccountable} \in \mathrm{NC}_{dq}$, representing the fact that the dataset has some provenance information stating that the dataset is coming from a known (highly accountable) source.

Consistency. As described in the model definition, *consistency* of a dataset states that it does not contain conflicting values. From the logical point of view, it is essential to know if a resource is not consistent, as this can influence the kind of reasoning that is possible to draw from it. Thus, it might be useful to rule out datasets that are recognized as inconsistent before attempting a reasoning activity over a set of datasets: we propose to include the class for datasets $\mathsf{isConsistent} \in \mathrm{NC}_{dq}$ providing a sharp measure for this aspect.

Completeness. Completeness of a dataset clearly depends to the task in which such piece of knowledge should be employed and from the set of attributes and objects known as needed for such task. Considering the scenario in which an UKB is modelled considering a specific task (and an external actor can define the set of attributes and objects for the task), we can represent the completeness metric of [11] (quantified over all of the interesting properties) as the role $\mathsf{LDSCompleteness} \in \mathrm{NR}_{dq}$. If needed, completeness with respect to the intensional and extensional part of a dataset can be distinguished as separate metrics.

Conciseness. The measure of conciseness regards the presence of redundant data or attributes in the dataset. *Intensional conciseness* refers to the measure of the number of unique attributes w.r.t. the number of overall number of attributes in a target schema, while *extensional conciseness* refers to the number of unique attributes w.r.t. the attributes in the whole set of data. The two measures can be defined as a metric (dataset property) in our metatheory: similarly to completeness, we propose to represent the conciseness metric proposed in [11] using the role LDSConciseness $\in \mathrm{NR}_{dq}$.

Structuredness. Two measures are proposed to assess the level of structure use of a dataset: *coverage* and *coherence*. In practice, coverage of a dataset w.r.t a class in a vocabulary define the level of usage of the concept in the dataset. The sum of this property with respect to the complete set of classes of a vocabulary basically defines the coherence: this can be defined as a reified property Coherence $\in \mathrm{NC}_{dq}$ using roles source, vocabulary, value $\in \mathrm{NR}_{dq}$ to encode the datasets, the reference vocabulary and the value of the measure. Alternatively, we can define this as a simple dataset attribute, by considering all of the vocabularies used by the considered dataset.

Other metrics. We summarize here the last metrics for data quality proposed by the model and we discuss how they can be represented in MT_{dq}.

Relevancy measures the level of usefulness of a resource to the task at hand: for this reason, such measure is clearly dependent from the considered task but also on factors as user, domain of knowledge, context and data representation. In [11] it is noted that approaches for assessing relevancy used in search engines, involving hyperlink analysis and information retrieval methods, can be adapted to the scenario of linked data. We can give the possibility to represent this (application defined) value introducing a role LDSRelevancy $\in \mathrm{NR}_{dq}$.

Validity refers to the conformance of the dataset with respect to a set of constraints or requirements. We note that some of such constraints can be encoded in the policies of the UKB, while other assessments may require external computation. We can represent this value defining a concept isValid $\in \mathrm{NC}_{dq}$.

Finally, the *reputation* dimension define the opinion users have of a dataset. As suggested in [11], the measure representing the PageRank accessibility of the dataset can be used as an estimate of reputation of a resource. Otherwise, reputation can be measured by assigning explicit user ratings about the data or their data source. We can represent this value introducing a role reputation $\in \mathrm{NR}_{dq}$: different means of reputation assessment (e.g. ratings from different rating services) can be represented as subtypes of such role.

6.2 Metatheory for factuality

6.2.1 Model for factuality

In the representation of data about events, it is useful to determine whether the described event has actually happened or not or if this fact is not defined. In the context of recognition of events from text, this aspect is called *event factuality* [15]. An event is said to be *factual* if it correspond to a real situation in the world, *counterfactual* if it describes a situation that has not happened and *non factual* if this can not be verified (e.g. the information is hypothetical or describes an event that can happen in the future). This aspect is strictly paired to the *certainty* value of the event data: for example, if an event is uncertain, then it can be neither factual nor counterfactual.

To model the two aspects and their correlation, we consider a simplification of the factuality model proposed in [15]. In the original paper, the two aspects are considered as a modality (certainty) and polarity (factuality) of an event information. In our case we only consider two levels of modality, namely *certain* and *uncertain*, and the three levels of polarity given by *factual*, *counterfactual* and *non factual*.

As introduced above, we want to model that an event can be factual or counterfactual only if it is certain. If an event is uncertain, then the event can only be non factual. Thus, in our model the relations across factuality and certainty can be summarized by the table below:

	Factual	Counterfactual	Non factual
Certain	●	●	●
Uncertain			●

6.2.2 Metatheory for factuality: MT_{fc}

We encode the presented model as a metatheory MT_{fc} for representation of factuality and certainty: the metatheory basically includes symbols to assert the factuality and certainty values of the datasets in $\mathbf{N}$ and axioms stemming from the presented relations across the two aspects.

The meta-vocabulary $\Gamma_{fc} = \mathrm{NC}_{fc} \uplus \mathrm{NR}_{fc} \uplus \mathrm{NI}_{fc}$ contains the following symbols:

- $\mathsf{Factual}, \mathsf{Counterfactual}, \mathsf{NonFactual} \in \mathrm{NC}_{fc}$, classes representing the possible factuality value of a dataset in $\mathbf{N}$.
- $\mathsf{Certain}, \mathsf{Uncertain} \in \mathrm{NC}_{fc}$, classes representing the certainty value of a dataset in $\mathbf{N}$.

Language $\mathcal{L}_{fc}$ is a DL language based on Γ_{fc} such that:

- $\mathsf{Factual}$, $\mathsf{Counterfactual}$ and $\mathsf{NonFactual}$ are mutually disjoint.
- $\mathsf{Certain}$ and $\mathsf{Uncertain}$ are mutually disjoint.

Moreover, the metatheory MT_{fc} contains the following axioms, encoding the relations across certainty and factuality:

$$\mathsf{Factual} \sqsubseteq \mathsf{Certain} \qquad \mathsf{Counterfactual} \sqsubseteq \mathsf{Certain} \qquad \mathsf{Uncertain} \sqsubseteq \mathsf{NonFactual}$$

The current version of the metatheory does not codifies rules to infer the factuality and certainty of derived datasets (i.e. datasets in a $\mathsf{derivedFrom}$ relation in MT_d). Such rules can be defined by the modeller and included in the policies: for example, if we want to define that “all datasets that are derived from at least an uncertain source are uncertain”, we can include in policies P the axiom $\exists \mathsf{derivedFrom}.\mathsf{Uncertain} \sqsubseteq \mathsf{Uncertain}$.

7 Conclusions

In this deliverable we presented the result of the second step in the definition of the formal structure for the MetaReasons framework: the definition of suitable encodings for metadata regarding (but not limited to) aspects of provenance, access control and trust.

After a brief introduction to the main ideas and basic definition of the MetaReasons framework, we described for each of the considered aspects one or more known models for its representation. For each model, we then provided its encoding as a metalevel theory, possibly extending with additional inference rules the basic calculus of the framework in order to reason over the newly added theory. By means of examples, we shown the possible use and combination of the different metatheories.

The next phase in the framework development will consist in its implementation over state of the art tools for storage and inference over RDF data. As already mentioned in our previous deliverable [3], the basic materialization calculus (possibly extended with the metatheory-dependent rules) will constitute the formal base for the implementation of a forward reasoning procedure over RDF data for our framework. Similarly to the CKR framework [2], the MetaReasons architecture can be implemented by representing the metalevel and the datasets as distinct RDF named graphs. Inference inside (and across) named graphs will be implemented as SPARQL based forward rules. In particular, following the same approach of [2], we plan to use an extension of the Sesame framework that we developed, called *SPRINGLES*[1], which provides methods to demand an inference materialization over multiple graphs: rules are encoded as SPARQL queries and it is possible to customize their evaluation strategy. The ruleset will encode the rules of the materialization calculus and the evaluation strategy will follow the calculus translation process. In this implementation phase, we will also provide (where not already available, as can be the case of PROV-O [10]) an OWL-RL based representation of the metatheory schemas we presented in this work and describe their possible combinations. The final evaluation phase for the assessment of the final prototype will also serve as a test for the practical applicability of the presented models in real data.

[1]*SParql-based Rule Inference over Named Graphs Layer Extending Sesame*.

Bibliography

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] Loris Bozzato and Luciano Serafini. Materialization Calculus for Contexts in the Semantic Web. In *DL2013*, CEUR-WP. CEUR-WS.org, 2013.

[3] Loris Bozzato and Luciano Serafini. D31.1 MetaReasons: theoretical architecture description. Deliverable D31.1, PlanetData, January 2014. http://planet-data-wiki.sti2.at/web/D31.1.

[4] J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW'05*. ACM, 2005.

[5] Timothy W. Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi S. Sandhu, William H. Winsborough, and Bhavani M. Thuraisingham. R*OWL*BAC: representing role based access control in *OWL*. In *SACMAT 2008*, pages 73–82, 2008.

[6] Giorgos Flouris. D3.2 Provenance Management and Propagation Through SPARQL Query and Update Languages. Deliverable D3.2, PlanetData, September 2013. http://planet-data-wiki.sti2.at/web/D3.2.

[7] Giorgos Flouris, Irini Fundulaki, Panagiotis Pediaditis, Yannis Theoharis, and Vassilis Christophides. Coloring RDF Triples to Capture Provenance. In *ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 196–212. Springer, 2009.

[8] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Procs. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006.

[9] Markus Krötzsch. Efficient Inferencing for OWL EL. In *JELIA 2010*, volume 6341 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2010.

[10] Deborah McGuinness, Timothy Lebo, and Satya Sahoo. PROV-O: The PROV ontology. W3C recommendation, W3C, April 2013. http://www.w3.org/TR/2013/REC-prov-o-20130430/.

[11] Pablo N. Mendes, Christian Bizer, Zoltán Miklos, Jean-Paul Calbimonte, Alexandra Moraru, and Giorgos Flouris. D2.1 Conceptual model and best practices for high-quality metadata publishing. Deliverable D2.1, PlanetData, March 2012. http://planet-data-wiki.sti2.at/web/D2.1.

[12] Paolo Missier and Luc Moreau. PROV-DM: The PROV data model. W3C recommendation, W3C, April 2013. http://www.w3.org/TR/2013/REC-prov-dm-20130430/.

[13] Boris Motik, Achille Fokoue, Ian Horrocks, Zhe Wu, Carsten Lutz, and Bernardo Cuenca Grau. OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C, October 2009. http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/.

[14] Vassilis Papakonstantinou, Maria Michou, Irini Fundulaki, Giorgos Flouris, and Grigoris Antoniou. Access control for RDF graphs using abstract models. In Vijay Atluri, Jaideep Vaidya, Axel Kern, and Murat Kantarcioglu, editors, *SACMAT*, pages 103–112. ACM, 2012.

[15] Saurí Roser and Pustejovsky James. From structure to interpretation: A double-layered annotation for event factuality. In *Proceedings of the 2nd Linguistic Annotation Workshop*, 2008.

[16] Wang Chiew Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.

[17] Lea Viljanen. Towards an ontology of trust. In *TrustBus 2005*, volume 3592 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2005.