From transparency of knowledge graphs to a general framework for defining assessment measures

Jennie Andersen

N° d'ordre NNT : 2024ISAL0047

THESE de DOCTORAT DE L'INSA LYON,
membre de l'Université de Lyon

Ecole Doctorale N° 512
École doctorale Informatique et Mathématiques de Lyon

Spécialité de doctorat :
Informatique

Soutenue publiquement le 05/06/2024, par :

Jennie Andersen

From transparency of knowledge graphs to a general framework for defining assessment measures

Devant le jury composé de :

Frédérique LAFOREST	Professeure	INSA Lyon	Présidente
Mathieu D'AQUIN	Professeur	Université de Lorraine	Rapporteur
Clément JONQUET	Directeur de Recherche	INRAE	Rapporteur
Fatiha SAÏS	Professeure	Université Paris-Saclay	Examinatrice
Hala SKAF-MOLLI	Professeure	Université de Nantes	Examinatrice
Philippe LAMARRE	Professeur	INSA Lyon	Directeur de thèse
Sylvie CAZALENS	Maîtresse de Conférences	INSA Lyon	Co-encadrante

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Département de la Formation par la Recherche et des Études Doctorales (**FEDORA**)

Bâtiment INSA direction, 1er étage
37, av. J. Capelle
69621 Villeurbanne Cédex
fedora@insa-lyon.fr

Référence : TH1102_ANDERSEN Jennie

L'INSA Lyon a mis en place une procédure de contrôle systématique via un outil de détection de similitudes (logiciel Compilatio). Après le dépôt du manuscrit de thèse, celui-ci est analysé par l'outil. Pour tout taux de similarité supérieur à 10%, le manuscrit est vérifié par l'équipe de FEDORA. Il s'agit notamment d'exclure les auto-citations, à condition qu'elles soient correctement référencées avec citation expresse dans le manuscrit.

Par ce document, il est attesté que ce manuscrit, dans la forme communiquée par la personne doctorante à l'INSA Lyon, satisfait aux exigences de l'Etablissement concernant le taux maximal de similitude admissible.

INSA LYON
Campus LyonTech La Doua
20, avenue Albert Einstein - 69621 Villeurbanne cedex - France
Tél. +33 (0)4 72 43 83 83 - Fax +33 (0)4 72 43 85 00
www.insa-lyon.fr

membre de UNIVERSITÉ DE LYON

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
ED 206 CHIMIE	**CHIMIE DE LYON** https://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne directeur@edchimie-lyon.fr
ED 341 E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE MODÉLISATION** http://e2m2.universite-lyon.fr Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**Mme Sandrine CHARLES** Université Claude Bernard Lyon 1 UFR Biosciences Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69622 Villeurbanne CEDEX e2m2.codir@listes.univ-lyon1.fr
ED 205 EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** http://ediss.universite-lyon.fr Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Laboratoire ICBMS - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
ED 34 EDML	**MATÉRIAUX DE LYON** http://ed34.universite-lyon.fr Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 stephane.benayoun@ec-lyon.fr
ED 160 EEA	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** https://edeea.universite-lyon.fr Sec. : Philomène TRECOURT Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 philippe.delachartre@insa-lyon.fr
ED 512 INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 direction.infomaths@listes.univ-lyon1.fr
ED 162 MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** http://edmega.universite-lyon.fr Sec. : Philomène TRECOURT Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Etienne PARIZET** INSA Lyon Laboratoire LVA Bâtiment St. Exupéry 25 bis av. Jean Capelle 69621 Villeurbanne CEDEX etienne.parizet@insa-lyon.fr
ED 483 ScSo	**ScSo**[1] https://edsciencessociales.universite-lyon.fr Sec. : Mélina FAVETON Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Bruno MILLY** (INSA : J.Y. TOUSSAINT) Univ. Lyon 2 Campus Berges du Rhône 18, quai Claude Bernard 69365 LYON CEDEX 07 Bureau BEL 319 bruno.milly@univ-lyon2.fr

[1] ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Après cette aventure de plus de trois ans, je tiens à exprimer ma gratitude aux différentes personnes qui m'ont accompagnée.

Mes premiers remerciements vont naturellement à Philippe Lamarre et Sylvie Cazalens parce que sans eux, rien de possible. J'ai de la chance de vous avoir eu comme directeur et encadrante de thèse. Je vous remercie pour la confiance que vous m'avez accordée, pour m'avoir si bien encadrée et accompagnée pendant ces trois années.

Je tiens également à remercier l'ensemble du jury, à commencer par les rapporteurs, Mathieu D'Aquin et Clément Jonquet, pour le temps consacré à la lecture de mon manuscrit et pour leur précieux retours. Merci à Hala Skaf-Molli, chacune de nos rencontres, en conférences, dans le cadre du projet ANR et pour cette soutenance, a été une source de motivation pour moi. Merci aussi à Frédérique Laforest et Fatiha Saïs d'avoir accepté de faire partie de ce jury et pour les échanges intéressants que nous avons eus.

J'ai eu la chance de participer au projet ANR DeKaloG, et je tiens à remercier l'ensemble de ses membres. En particulier, j'exprime ma plus sincère gratitude à Pierre Maillot pour avoir été un remarquable collaborateur, et à Fabien Gandon, Franck Michel et Catherine Faron pour m'avoir accueillie dans leur équipe à Sophia-Antipolis. Ce séjour a illuminé ma thèse.

Merci à toute l'équipe administrative du LIRIS, en particulier Caroline Ferri qui m'a permis de faire mes déplacements en toute sérénité malgré des outils parfois ingrats, et pour sa sympathie de tous les jours. Merci aussi à Jean-Marc Petit, directeur du LIRIS, pour sa bienveillance et pour m'avoir donné quelques privilèges dans le choix de mon bureau.

Enfin, je tiens à remercier tou.te.s mes collègues et ami.e.s découvert.e.s pendant cette thèse : Simon, mon premier ami de thèse, arrivé un peu trop tard et parti un peu vite, et Pierre, soutien indéfectible, puis Théophile, Marc, Cédric, Juliana, Taha, Aymar, Vincent, et Martial. Je leur souhaite tout le meilleur pour la suite et la fin de leur thèse. Je remercie aussi mes partenaires d'escalade, Anne et Coraline, qui m'ont permis de me vider l'esprit une fois par semaine. Mes derniers remerciements vont à ma famille, ainsi qu'à Rémi, pour m'avoir soutenue, encouragée, remotivée, et pour avoir été présent à chaque instant.

Abstract

Many Knowledge Graphs (KG) are available on the Web, and it may be difficult to decide which one to work with. Various criteria may influence this choice, beyond the relevance of the domain and the content, the use of standards, the identification of the creators... are also important. Indeed, the opening up of more and more data, encouraged by data openness policies of governments and the growing importance of data in today's society, comes with additional requirements in terms of quality and transparency.

To help users choose one KG over another, we aim to provide an estimate of the transparency of a given knowledge graph. When thinking about this notion, several questions naturally come to mind. Do we know who created the KG? From what source? How? For what purpose?... This kind of information is essential to build trust in the data and increase their reuse. In addition, provenance information enables their reproducibility and verification. However, the notion of transparency does not have well-defined boundaries. To try to understand it, we first explore the notion of transparency and its related concepts (openness, accessibility, verifiability...). Then, given the lack of precise requirements for transparency as a whole, we focus on one of its closely related concepts and propose a measure of the accountability of KGs. We use our measure to evaluate hundreds of KGs available through SPARQL endpoints. While most of them do not provide any accountability information within their data, our measure allows to discriminate among the others. Finally, we compare our measure with others studying data quality or FAIRness of KGs.

This comparison highlights that each measure has its own particularities, but also shares similarities with many other existing measures. As a result, choosing the proper measure to evaluate KGs for a given task is not easy, since they are described in many different ways and places. Given that many of them rely on a hierarchical structure, we propose to define a formal basis for describing the measures in a common framework. It aims to facilitate their understanding, reuse, comparison, and sharing by defining operators to manipulate them, either to build new ones, or to compare them. We also propose a web application for designing and comparing measures defined in this way.

Résumé

De nombreux graphes de connaissances (KG) sont disponibles sur le Web, et il peut être difficile de décider avec lequel travailler. Différents critères peuvent influencer ce choix, au-delà de la pertinence du domaine et du contenu, l'utilisation de standards, l'identification des créateurs... sont également importants. En effet, la mise à disposition de toujours plus de données, encouragée par les politiques gouvernementales d'ouverture des données et l'importance croissante des données dans notre société actuelle, s'accompagne d'attentes supplémentaires en termes de qualité et de transparence.

Afin d'aider les utilisateurs à choisir un KG plutôt qu'un autre, nous voulons fournir une estimation de la transparence de KG. Lorsque l'on pense à cette notion, plusieurs questions se posent naturellement. Savons-nous qui a créé le graphe de connaissances ? À partir de quelle source ? De quelle manière ? Dans quel but ? Ces types d'information sont essentiels pour renforcer la confiance dans les données et favoriser leur réutilisation. En outre, les informations de provenance permettent la reproductibilité des données et leur vérification. Cependant, les contours de la notion de transparence ne sont pas clairement définis. Pour tenter de mieux la comprendre, nous explorons tout d'abord cette notion et ses concepts associés (accessibilité, ouverture, vérifiabilité...). Étant donné l'absence d'exigences précises concernant la transparence dans sa globalité, nous nous concentrons ensuite sur un concept proche, et proposons une mesure de « l'accountability » des KG. Puis, nous utilisons notre mesure pour évaluer des centaines de KGs disponibles via des SPARQL endpoints. Bien que la plupart d'entre eux ne fournissent aucune information sur l'accountability dans leurs données, notre mesure permet de distinguer et départager les autres. Enfin, nous comparons notre mesure avec d'autres mesures pour les KG sur la qualité des données et les principes FAIR.

Cette comparaison montre que chaque mesure a ses propres spécificités, tout en partageant des points communs avec les nombreuses autres mesures existantes. Aussi, choisir la mesure appropriée pour évaluer les KG dans le cadre d'une tâche donnée n'est pas aisé, d'autant plus qu'elles sont décrites de manières variées et à différents endroits. Étant donné que beaucoup reposent sur une structure hiérarchique, nous proposons de définir une base formelle pour décrire les mesures dans un cadre commun. Ce cadre vise à faciliter leur compréhension, leur réutilisation, leur comparaison et leur partage en définissant des opérateurs permettant de les manipuler, soit pour en créer de nouvelles, soit pour les comparer. Nous proposons également une application web pour concevoir et comparer des mesures définies de cette manière.

Contents

1 Introduction

Contents

1.1 General context

Knowledge Graphs (KGs) are at the core of numerous tasks and applications such as search engines, personal voice assistants, recommendation engines, fraud detection, drug discovery, and many others. According to Paulheim [1], a "knowledge graph *(i)* mainly describes real world entities and their interrelations, organized in a graph, *(ii)* defines possible classes and relations of entities in a schema, *(iii)* allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains." In addition, the opening of more and more KGs following the Linked Open Data principles gives access to a huge amount of interrelated data for both humans and machines. Indeed, Linked Data principles states the following requirements [2, 3]: use URIs as names for things, use HTTP URIs so that people can look up those names, when someone looks up a URI, provide useful information, using the standards (RDF, SPARQL), and include links to other URIs, so that they can discover more things.

These open KGs can be generic and cross-domain, such as Wikidata [4] and DBPedia [5], which contain general knowledge extracted from Wikipedia or collaboratively edited by people. Cross-domain KGs enable accessing information about people, places, books, and so on. For instance, the BBC used DBPedia to enrich the content of its news stories with information about the people involved [6]. Many other KGs are specific to a particular domain, such as life sciences, linguistics, geography, government, etc., as illustrated by the LOD cloud diagram[1]. For instance, LinkedGeoData [7] contains spatial information and aims to simplify information integration and provide contextual information and background knowledge on places and locations. In the world of Semantic Web, ontologies play an important role. They are "a formal, explicit specification of a shared conceptualization that is characterized by high

[1] https://lod-cloud.net/

semantic expressiveness" [8], "a description of the concepts and relationships that really or fundamentally exist for a particular domain" [9]. While, knowledge graphs are sometimes considered synonymous with ontologies, or considered as "very large ontologies", it is a more general concept that may use ontologies [8]. For instance, BioPortal [10] gives access to numerous biomedical ontologies to increase interoperability between these types of data and to help improve clinical care.

With so many available KGs, it may be difficult to decide which one to choose. In addition, the opening up of more and more data, encouraged by data openness policies of governments and the growing importance of data in today's society, comes with additional requirements in terms of quality and transparency. Therefore, when choosing one or more KGs, various criteria may be taken into account. Beyond the relevance of the domain and content, the use of standards, the identification of the creators of the KG, and numerous other criteria can influence this choice. This thesis is part of the ANR project DeKaloG, which "encourages the growth of a public and decentralized web of knowledge graphs"[2]. The project notably aims to improve the findability of KGs by building a semantic index of KGs, itself a KG, with descriptions of the KGs and ranking statistics ; and to promote transparency by proposing models for capturing different levels of transparency and a method for efficiently querying them. Combining these objectives, a transparency value can be given to the indexed KGs to encourage greater transparency and to promote the most transparent knowledge graphs when choosing among several possible ones. To this end, we focus more specifically on transparency and aim to provide an evaluation of the transparency of open knowledge graphs, with clear and explicit expectations and measurements. As part of the project, both an index and transparency evaluation aim to be used for federated queries. So, as the project, we focus on RDF graphs in particular.

When thinking of this notion, some questions come naturally to mind. Do we know who created the KG? From what source? How? For what purpose?... This kind of information is essential to build trust in the data and increase their reuse. Knowing how the data were produced and from what source enables their reproducibility and verifiability. In addition, knowing who created these data may provide insights into the choices made and the orientation taken in the collection and management of the data, such as geographical restrictions or potential biases.

However, the notion of transparency does not have well delimited boundaries [11, 12]. It has been studied in many fields, including computer science. However, there is no real consensus on the definition of this concept. Basically, it requires access to more information for all users. But there are no clear and explicit requirements for the transparency of a data source, nor any measure or evaluation, be it for KGs or any other data source. More precise definitions or clearer requirements can be identified through the prism of associated concepts such as

[2] https://anr.fr/Project-ANR-19-CE23-0014

provenance, reproducibility, accountability, openness, accessibility. They also show that general expectations about the transparency of a KG rely on the metadata it provides, such as to answer the questions above. In accordance with this view, we only focus on metadata in this thesis. In our context, it corresponds to all data used to describe the KGs.

Other studies have also addressed the issue of the evaluation of KGs and proposed some measures. For instance, data quality studies, which consider both data and metadata, evaluate many different characteristics of KGs [13] such as accessibility, accuracy, conciseness, etc. More recently, the FAIR (**F**indability, **A**ccessibility, **I**nteroperability, **R**eusability) principles [14] have been stated in very general terms to promote good practices in the management of online resources. They have then been adapted into measures of RDF graphs [15, 16]. Like transparency, some aspects of these measures require access to metadata. Hence a key question for both these measures and transparency is to know exactly which metadata are required and how to find them.

Indeed, finding precise metadata in public KG is not easy and many obstacles stand in the way of transparency. First of all, few KGs provide metadata [17]. When they do, there are several places to look for this information. It can be found outside the KG itself, as textual information on its web page, or in the metadata of that page. It can also be accessed from its description in a public data repository (such as Zenodo) where the KG is registered. Alternatively, the metadata can sometimes be found by dereferencing a particular URI of the KG, or directly within its data.

Beyond the multiplicity of places where metadata can be found, there is also the problem of the heterogeneity of their expression. There exist numerous RDF vocabularies and several are dedicated to the description of datasets [18]. For instance, there are many ways to say that someone created the KG. This hinders access to the relevant metadata and thus reduces in some way the transparency of KGs.

1.2 Overview of the thesis

Facing the absence of consensus on the definitions of transparency, we make a review of several definitions of transparency and its related concepts, such as reproducibility or openness. While there is no existing measure of the transparency of a source, the study of these elements highlights the difficulty of considering or qualifying something as transparent in absolute terms without a more explicit context. Therefore, we propose a new definition of the transparency of a data source that emphasizes the importance of context and of defining an information need.

Given the lack of precise requirements for transparency as a whole, we focus on accountability. It is a concept close to transparency [19, 20], that has been studied in detail by Oppold and Herschel [21]. The authors propose a metadata model for expressing accountability information

about datasets and define clear requirements for the information, i.e. the metadata, to be provided in order to be considered accountable. Although it is general for any type of datasets, it can be adapted to knowledge graphs. From this starting point, we define a set of requirements to evaluating KGs.

We choose to search for the metadata within the KG itself, among the other data. Indeed, transparency requires that information be easily accessible [22], so metadata should be in the same location as the KG. In addition, no different technology is needed to access metadata compared to data, making them more accessible again. Furthermore, there is no major obstacle for KG providers to make metadata available within the KG: this is made possible by the particular structure of the KG, which can contain heterogeneous data and mix both data and metadata, possibly partitioned by different sub-graphs. Finally, as part of the DeKaloG project, we attach particular importance to the data present within the graph and accessible via a SPARQL endpoint.

In this way, we propose a measure of KG accountability and an evaluation of hundreds of KGs available through a SPARQL endpoint. To our knowledge, this is the first measure of KG accountability. To analyze the relevance and novelty of our measure, we then compare our measure with others measures from data quality studies or adapted from the FAIR principles.

Going further with KGs evaluation, there are numerous measures, some of which assess common or similar characteristics. This raises the question of how to choose one measure over another, how to reuse them and, more generally, how to compare them. Given the similarity of structure observed between the measures studied, we propose to define a formal basis for describing the measures in a common framework. We add elements for combining and modifying existing measures to facilitate the reuse and creation of new measures. We also propose some elements for comparison. Finally, to put this into practice, we propose a web application for designing and comparing measures.

1.3 Manuscript organization

This thesis is organized in four main chapters.

In Chapter 2, we briefly introduce the Semantic Web and knowledge graphs. Then, we explore the notion of transparency and its related concepts as described in the literature of Linked Data and of some other domains. Finally, we present existing measures and frameworks for evaluating knowledge graphs.

Chapter 3 is dedicated to accountability as a particular aspect of transparency. We define the KGAcc framework: it provides a structured measure of accountability expressed as both natural language requirements and SPARQL queries. Then, the measure is used to evaluate

many knowledge graphs accessible via SPARQL endpoints. Finally, the accountability measure is compared with measures of data quality and of the FAIR principles.

In Chapter 4, we propose a formalization to clearly and explicitly define and represent measures as tree-structures of requirements. We also detail some operators to manipulate them and to create new measures by reusing existing ones. We propose additional operators to facilitate their comparison. Finally, we define an RDF vocabulary to enable the representation of the measures thus defined in the Semantic Web.

Chapter 5 presents a web application whose formal basis relies on the previous chapter. It is dedicated to the definition of such tree-like measures and to their comparison. This application also allows to evaluate knowledge graphs with these measures and to analyze the results.

Finally, in the last chapter, we conclude this thesis and propose some future work.

2

Preliminaries

Contents

STUDYING the transparency of knowledge graphs raises several questions. What is the transparency of knowledge graphs? How to evaluate it? To provide some answers, we start by briefly describing knowledge graphs. Then, to address the lack of consensus on the precise meaning of the concept, we explore the notion of transparency. We seek to understand it and its different meanings and to identify what it requires, widening a bit this context of Linked Data. Finally, we focus on the evaluation aspect and discuss the existing measures of knowledge graphs and of assessment frameworks.

2.1 Knowledge Graphs and Semantic Web

The Semantic Web was introduced by Tim Berners-Lee in 2001 with an ambitious promise: "A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities" [23]. It aims at changing the paradigm from a web of documents to a web of (linked) data. It has been promoted and standardized by the World Wide Web Consortium (W3C). Since then, its use has grown in the academic world as well as in the industry.

A term that often comes up with the semantic web is the term Knowledge Graph (KG). It is commonly understood as a graph representing real-world entities and their relationships. This term was popularized by the Google's Knowledge Graph in 2012 [24]. But other knowledge graphs existed before, even if they were not qualified as such, like those based on Wikipedia: DBpedia [5], YAGO [25], and Wikidata [4]. And others have been proposed later, may they be openly accessible or not, such as the knowledge graph of products of Amazon [26]. Therefore, several definitions of *knowledge graph* exist and they are not always consistent with each other, as highlighted by Ehrlinger and Wöß [8]. The authors list some representative definitions. They range from "networks of all kind things which are relevant to a specific domain", to "large networks of entities, their semantic types, properties, and relationships between entities", to

any graph-based knowledge representation or limited to those using ontologies and reasoners. Paulheim [1] provides a more precise definition: a "knowledge graph *(i)* mainly describes real world entities and their interrelations, organized in a graph, *(ii)* defines possible classes and relations of entities in a schema, *(iii)* allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains." Since in this thesis, we focus on RDF graphs, we adopt the more specific definition of Färber et al. [27], who "define a knowledge graph as an RDF graph", whose formal definition is given below.

In this section, we describe the standards adopted by the W3C for semantic web, starting with RDF, before clarifying the terms used throughout this thesis. Then we describe some existing vocabulary which we use in this thesis.

2.1.1 RDF graphs and SPARQL queries

RDF (Resource Description Framework) [28] is a data model that enables to describe resources and so to represent information on the Web. Resources are real-world objects, people, places, abstract concepts, etc., that are identified by an IRI (Internationalized Resource Identifier). These resources are described by *RDF triples* of the form (subject, predicate, object), where the subject represents the resource, the predicate is a property that connects the subject to the object, and the latter can be seen as the value of the property for the given subject. For instance, the RDF triple (Alice, lives, Liechtenstein) states that Alice lives in Liechtenstein. Notice that Alice here is the identifier that represents a person, not the person itself. Hence it is interesting to add triples like (Alice, name, "Alice") to specify who the identifier Alice refers to. The same applies for Liechtenstein. The elements forming a triple can be of several types: an IRI, i.e. the identifier representing a resource ; a literal, i.e., a value such as a string, number, or date ; or a blank node, i.e., a local identifier to refer to an anonymous resource that is partially known. For instance, the two triples (Alice, isFriendOf, _1) and (_1, isFriendOf, Bob) use a blank node '_1' which is an unknown person, but they allow to say that Alice and Bob have a friend in common. In the triple (Alice, age, "84"), *Alice* and *age* are IRIs, and *84* is a literal. Figure 2.1 illustrates the RDF graph that results from these triples. Formally, RDF triples and RDF graphs are defined as follows.

Definition 1 (RDF triple). *Let $\mathcal{I}$ be the set of IRIs, $\mathcal{B}$ the set of blank nodes, and $\mathcal{L}$ the set of literals. An RDF triple is a tuple $(s, p, o) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, where s is the subject, p the predicate and o the object [29].*

Definition 2 (RDF graph). *An RDF graph, or RDF dataset, is a set of RDF triples [29].*

To retrieve information from RDF graphs, a query language has been developed: SPARQL (**S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage) is a graph-matching query language for

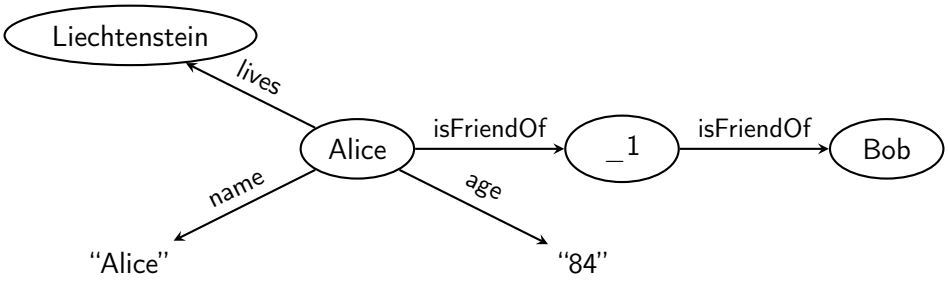


Figure 2.1: Example of a small RDF graph

querying RDF graphs. The core of SPARQL is based on graph patterns: the goal is to match them against the knowledge graph to solve the query. Hence, a query provides one or more triple patterns in which one or more elements are variables.

Definition 3 (Triple pattern). *Let $\mathcal{V}$ be the set of (query) variables. A triple pattern is a tuple* $(s, p, o) \in (\mathcal{V} \cup \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) \times (\mathcal{V} \cup \mathcal{I}) \times (\mathcal{V} \cup \mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$[1] *[30].*

A graph pattern is a set of triple patterns. The simplest graph pattern is a conjunction of triple patterns, in which case they must match the graph all together. There exist more complex graph patterns, formed by disjunctions of graph patterns, where at least one of them must match ; or by optional graph patterns, which can be used to extend the solution without being more restrictive, etc.

Therefore, a simple SPARQL query is of the form `SELECT` V `WHERE` P, with V being a set of variables to identify and P a graph pattern. The result of the query consists of a set of bindings between the variables and their matching values in the queried RDF graph. More precisely, it returns "all, or a subset of[2], the variables bound in a query pattern match" [30]. There exist other types of SPARQL queries, also described in the W3C specification [30]. ASK queries result in a boolean "indicating whether a query pattern matches or not". CONSTRUCT queries return "an RDF graph constructed by substituting variables in a set of triple templates". Finally, DESCRIBE queries provide "an RDF graph that describes the resources found". We will not go further in the presentation of SPARQL and refer to the W3C specification [30] for the complete formalization.

Example 1. Let us illustrate SPARQL queries with two small examples. Listing 2.1 is a simple SELECT query that looks for for all entities that are subject of the isFriendOf property. According to the small graph represented in Figure 2.1, two entities match for `p1`: Alice and the blank node '_1'. An ASK query is given in Listing 2.2. It requires an entity named "Alice" whose age is 44. Since Alice is 84 in our example, there is no matching entity and the result of the query is False.

[1] Notice that s could theoretically be a literal according to the W3C specification of SPARQL, even if the associated triple pattern will not match any RDF triple.

[2] It returns only a subset if there is an optional graph pattern.

```
SELECT ?p1
WHERE { ?p1 isFriendOf ?p2 .}
```

Listing 2.1: Simple SPARQL SELECT query

```
ASK { ?p name "Alice" .
      ?p age "44" . }
```

Listing 2.2: Simple SPARQL ASK query

In this thesis, we use the following terms, which are also recalled in the Glossary. A *knowledge graph* to refer to an RDF graph. In addition, when considering RDF graphs, we use the term *dataset* to denote either a whole RDF graph or a subgraph of it[3]. The usual method for defining and identifying such subgraphs is to use named graphs. The two most common ways to access an RDF graph are through a dump or a SPARQL endpoint. A *SPARQL endpoint* is a web service that provides access to the data via SPARQL queries. It allows both human and software agents to send complex SPARQL queries to the KG to retrieve information. For instance, the SPARQL endpoint of Wikidata [4] is currently `https://query.wikidata.org/`[4]. A *dump*, or data dump, or RDF dump, is an RDF export of the dataset. There are several advantages and drawbacks for both of them. A SPARQL endpoint provides access to the latest version of the knowledge graph and is easy to query since it is possible to send a SPARQL query directly to it. However, there might be restrictions that hinder the usability of the SPARQL endpoint [17], such as a maximum runtime for a query, a limited number of results, or of implemented SPARQL features, etc. A dump is the exact opposite, it is not always up-to-date compared to the data owned by the provider, it is not straightforward to query, and it requires enough local resources to store it. However, a local SPARQL endpoint can be used to give access to it without the limitations of a remote SPARQL endpoint.

2.1.2 Common RDF vocabularies

In addition to being an abstract syntax for the expression of linked data, RDF is also a vocabulary that provides a few basic concepts, such as the class `rdf:Property`, the property `rdf:type` indicating that an element is of a certain class, or other terms for the expression of collections of elements. RDF is extended by the RDF Schema (RDFS) model [31], which is designed for the definition of new vocabularies with terms such as `rdfs:subClassOf`, and by OWL [32], which allows to describe richer vocabularies and to reason on the data with properties like `owl:sameAs` for instance.

As a result, numerous vocabularies have been defined, shared, and adopted by the semantic web community. The Linked Open Vocabularies (LOV) catalog [33] lists many of them. There are some general vocabularies that are domain agnostic, such as vocabularies for describing concepts or people, and some specific vocabularies that concern only certain domains or use

[3] Note that the term is also used in its broader sense in the rest of this chapter.
[4] Accessed: 15 February 2024

Table 2.1: Main vocabularies and prefixes used (by alphabetic order of prefix)

Short Name	Prefix & Namespace	Description
ccRel	cc: http://creativecommons.org/ns#	Licence
DataID	dataid: http://dataid.dbpedia.org/ns/core#	Dataset
DCAT	dcat: http://www.w3.org/ns/dcat#	Dataset
DC Terms	dct: http://purl.org/dc/terms/	General
DQV	dqv: http://www.w3.org/ns/dqv#	Quality
FOAF	foaf: http://xmlns.com/foaf/0.1/	General
PAV	pav: http://purl.org/pav/	Provenance
PROV-O	prov: http://www.w3.org/ns/prov#	Provenance
schema.org	schema: http://schema.org	General
SPARQL-SD	sd: http://www.w3.org/ns/sparql-service-description#	Endpoint
SKOS	skos: http://www.w3.org/2004/02/skos/core#	General
VoID	void: http://rdfs.org/ns/void#	Dataset

cases, such as vocabularies for describing biological species. Without being exhaustive, we focus on the general vocabularies and present some important ones that we used in this thesis to describe datasets and related concepts. Most of them are listed in Table 2.1. An exhaustive list of the vocabularies used in this thesis can be found in Appendix A.1, with their full and short names, their prefix and namespace, and the context in which we used them.

Several vocabularies are dedicated to the description of datasets (or catalogs). The VoID vocabulary [34] is specifically designed for the expression of metadata of RDF datasets. It allows to express statistical properties about the number of entities, triples, classes... as well as to reference a SPARQL endpoint, a dump, some vocabularies used, and so on. VoID is used in many semantic web applications (SPARQLES [35], SPORTAL [36], LOD Laundromat [37]...). Provided by the W3C, but without being a recommendation[5], "it has become the *de facto* standard for describing RDF datasets" [35] with many good practices based on VoID. The DCAT vocabulary [38] is a W3C recommendation for describing datasets and data catalogs published on the Web. DataID [39] extends these vocabularies to provide richer descriptions of datasets and catalogs. It is supported by DBPedia. Finally, SPARQL-SD [40] is another W3C recommendation that enables the description of SPARQL endpoints. Gray et al. [41] have established the need to combine them in order to properly describe datasets, since none of them is sufficient on its own. They also listed the most important properties in them.

All these vocabularies rely on more general ones. The Dublin Core[6] is a very general metadata model, designed to describe documents. It provides properties such as creator, contributor, title, description, etc. It is divided into three vocabularies: the "Dublin Core elements" (DC Elements), which is deprecated but widely used, the "Dublin Core terms" (DC

[5] I.e. an official standard.

[6] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

Terms) which replaces and extends the latter, and the "DCMI Type Vocabulary" (DCMI Type), which provides additional classes, such as one for dataset.

Creative Commons allows to describe some rights, licenses and conditions of use and to associate them with documents using the ccRel vocabulary[7]. The Dublin Core and other vocabularies also enable the description of rights or licenses. FOAF[8] is a vocabulary mainly used to describe people, their identity and various contact information. It can also describe relationships between them, as well as groups of people. Another interesting vocabulary is SKOS, a W3C recommendation [42]. It is centered on the definition of "concepts" to form Simple Knowledge Organization Systems (SKOS), such as classifications, thesauri... It allows to precisely describe concepts with different types of labels, descriptions and examples, and links between concepts. Finally, schema.org is a very general and widely used vocabulary, founded by Google, Microsoft, Yahoo and Yandex. Originally dedicated to adding microdata to web pages, it enables the description of many types of entities, such as creative works (which include datasets), people, organizations, places, events, and so on.

Provenance and data quality vocabularies are two other types of vocabularies of particular interest. PROV-O [43] is a W3C recommendation for the expression of provenance issues. It mainly relies on three classes: an "agent" (human or software) is responsible for an "activity" that affects an "entity". Where an entity is "a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary" [43]. Therefore, it can describe complex processes that create, modify, use, or affect an entity. The PAV vocabulary [44] extends and specializes PROV with more specific properties. Finally, DQV [45], presented by a W3C note, is used to describe various kinds of quality information about datasets, including measurements, metrics, and classification of metrics. This vocabulary is particularly interesting for representing results of experiments.

To our knowledge, there is no vocabulary dedicated to transparency (of a dataset). However, some of the presented vocabularies allow to think about various elements to look for in the context of transparency. Before doing so, we need to focus on this notion of transparency to better understand it, and later try to identify exactly what is interesting in these vocabularies.

2.2 Transparency and related concepts

The demand for transparency is high and is growing in many areas, including everyday life (politics, government, administration, economy, environment, health, personal data management, etc.) and science. Although often cited and desired, the contours of this notion often

[7] http://creativecommons.org/ns
[8] http://xmlns.com/foaf/spec/

remain blurred and can even vary significantly from one context to another. Such lack of precision makes it possible for an element to be qualified as transparent by its promoters without actually meeting the expectations of external observers. For example, many governments have made significant efforts to provide open data [11]. However, from the public's point of view, there are limitations that can hinder transparency: file formats that limit queries (e.g. image scans instead of machine-readable data), data formats that are difficult to use (e.g. proprietary format), little information on how certain data are produced, difficulties integrating with other sources, etc. The use of norms and standards for data publication (domain modeling, data format, access and query protocols...) brings an obvious improvement. So does Tim Berners-Lee's five stars scheme [46] for linked data which addresses some of the issues raised above. Following this recommendation, Knowledge Graphs make it possible to represent, publish, give access to and search for information in any domain. Therefore, the use of a public knowledge graph, with an open access to its content, can only improve transparency. However, it is not sufficient in itself to guarantee it.

In this section, we study transparency with the overall goal of evaluating the transparency of a knowledge graph and being able to answer the question: is this knowledge graph transparent? More generally, we try to identify what are the requirements for the transparency of a source. Therefore, we first present a review of different definitions of transparency that can be found in the literature. To understand the concept in a broader scope, we study transparency not only as it is described in computer science but also in other domains. We also present many of its companion concepts that are very often used to complement and clarify it: accessibility, accountability, completeness, findability, interoperability, openness, privacy, provenance, quality, repeatability, replicability, reproducibility, trustworthiness, understandability, verifiability, etc. The synthesis makes us distinguish between notions that characterize data sources and those that characterize users' needs, and leads us to propose a new definition of transparency that aims to apply to any context and consider these related concepts. The study of transparency and its related concepts has been presented in a report for the ANR DeKaloG project [47].

2.2.1 Transparency: Its various meanings

Although transparency is cited in many papers, it remains a rather "ill-defined concept" when going beyond the general definition of a dictionary, with no uniform view of what constitutes it [11]. We do not intend to provide an exhaustive list of the different meanings of transparency in the literature. Rather, we analyze different definitions and usages through some representative articles. We selected a literature review on transparency to get a global view on its meanings and identify what it requires, articles that specifically focus on transparency in

the context of knowledge graphs, and some articles that focus on transparency in computer science. All these articles come from different domains, not only computer science, but also governance, law, biology, etc.

Starting from a general understanding, the Merriam-Webster dictionary defines *transparent* as "characterized by visibility or accessibility of information especially concerning business practices"[9]. Visibility and accessibility are also mentioned in the context of governance, policy and law in [12], where several definitions of transparency are given. They range from "a metaphorical window [...] allowing outsiders of [...] a decision-making body to gain [...] visual access to that body's insides" to others "based on the availability of information". Others definitions describe transparency slightly differently: "in a relational manner", between observer and observed or ruler and ruled. In a study on transparency of governments with linked data, several parties are also mentioned as well as the provision of information: Matheus et al. [11] emphasize that "transparency is aimed at overcoming the information asymmetry between the government and the public". In the same spirit, Schwabe et al. [48] propose the following definition of transparency: "The availability of information about an actor that allows other actors to monitor the workings or performance of the first actor."

This ability to monitor, check, and verify appears in many domains that require transparency. For example, data journalism and fact-checking particularly encourage journalists to be more transparent about their methodology and sources[10] in order to increase their trustworthiness: "transparency refers to a verification process presented in a way that allows the audience or readers to decide for themselves why they should trust or distrust it" [49]. As another example, biological researchers have developed their own platform, Galaxy[11], in order to promote transparency as well as accessibility and reproducibility [50], offering not only the possibility to verify results but also to reuse them.

In computer science, interest in transparency is recent and is growing with the use of big data along with personal data and machine learning [51]. Transparency of digital objects, from data to software, is becoming a key issue. For instance, Bertino [22] introduces several definitions of transparency: it is "the ability to access and work with data no matter where it is located" and "the guarantee that the data being provided is accurate and from some official source". She argues that these definitions are no longer adequate and proposes a new one that is privacy-oriented: "Data transparency is the ability of subjects to effectively gain access to all information related to data used in processes and decisions that affect the subjects." This is consistent

[9] https://www.merriam-webster.com/dictionary/transparent Accessed: 11 April 2023

[10] Note that *source protection* is an essential right of journalists in many countries, which prohibits authorities from forcing them to reveal the identity of an anonymous source. Therefore, transparency of sources should not be imposed, nor is it desirable in all circumstances.

[11] www.galaxyproject.org

with the definition "Transparency is the ability to interpret the information extraction process in order to verify which aspects of the data determine its results." [52], where transparency is considered a dimension of data quality. And it may also be required for algorithms: "an algorithm is transparent (for a specific purpose) if it discloses its motivation and actions" [53].

These selected examples already show that definitions of transparency are highly contextual. Obviously, they depend on the object to be transparent: transparency of a government is not the same as transparency of data. But even for the same object, there can be significant variations, as shown by Wyatt [12] in the context of government and by Bertino [22] for data. But all the definitions aim at the provision of more information. They require the access to information, uses and motivations related to the data and the process. Moreover, transparency can only be truly attested by an external observer (subjects, the public...). Therefore, transparency is needed "for a specific purpose", which is user-defined, and requires an observed element, an observer [48] and a means of observation [12]. In addition, transparency often involves other concepts that support this contextual aspect but also circumscribe its boundaries.

2.2.2 Related concepts

Many concepts are associated with transparency, either to describe situations where the need for transparency is important, or to specify how to obtain transparency. We study them, mostly in the context of computer science, to improve our understanding of the notion and to identify more precise requirements for it. Without claiming to be exhaustive, we present the concepts most frequently associated with transparency. For each of them, we detail their relationship with transparency and propose to retain an intuitive definition in order to obtain a set of coherently defined concepts.

Openness

Openness refers to the idea of making all data or information that one has open and available. This concept is particularly present in the field of public administration. Indeed, asking for all the information used by an administration to be made public is quite an intuitive solution to bridge the information gap between the administration and the public. Such a consideration has paved the way for the emergence of open data [54], which also offers the opportunity to reuse data in order to expand knowledge and create new products and services. In some cases, transparency and openness are so close that they are considered synonymous [11, 12]. When they are note synonymous, the limit between these two is unclear when they are defined together by "There should be no secret record keeping. This includes both the publication of the existence of such collections, as well as their contents." [48]. Indeed, if the actors who

demand transparency share the same goals as those who open the data they collect, then hiding nothing should allow both to work properly with these data. However, if the actors involved do not share the same goal, then a difference between transparency and openness emerges. For instance, [11] shows that while openness is a necessary condition for transparency, only accountability (see below) can testify whether the data are sufficient to be transparent, whether enough information has been recorded and disclosed.

To take into account the possible difference between the data available according to one actor and the requirements of another actor, the intuitive definition we propose explicitly limits the scope of openness to available information.

Intuitive definition 1. *Openness is the full disclosure of all available data and information.*

Accessibility

While openness is necessary for transparency, it is not enough. Indeed, many of the previous definitions of transparency also explicitly require that the information be accessible, in the sense of being reachable, usable, or visible. For instance, accessibility appears in Bertino's definitions such as "effectively gain access to all information" and "the ability to access data" [22].

In the domain of data management, accessibility is one of the FAIR principles [14], which aim to improve machine actionability. In particular, they aim at acting "as a guideline for those wishing to enhance the reusability of their data holdings", from data to algorithms and workflows. According to the FAIR principles, accessibility requires that: "A1. (meta)data are retrievable by their identifier using a standardized communications protocol ; A1.1 the protocol is open, free, and universally implementable ; A1.2 the protocol allows for an authentication and authorization procedure, where necessary ; A2. metadata are accessible, even when the data are no longer available".

Finally, accessibility also appears as a dimension of the quality of knowledge graphs (KG). In [27], accessibility is "the extent to which data are available or easily and quickly retrievable", and it can be measured according to seven criteria: "dereferencing possibility of resources, availability of the KG, provisioning of public SPARQL endpoint, provisioning of an RDF export, support of content negotiation, linking HTML sites to RDF serializations, provisioning of KG metadata". It is also usually associated with a need for licensing [13, 27], as in this definition: "Accessibility implies that data or part of it must be available, retrievable, and contain a license" [55]. Interlinking [13, 27], security, and performance [13] are also sometimes mentioned as part of accessibility. We adopt this general definition, which takes into account the criteria that appear in all definitions.

Intuitive definition 2. *Accessibility is "the extent to which data are available or easily and*

quickly retrievable" [27].

Accountability

Accountability is a major topic in political and governmental research about transparency. For instance, transparency of governments in the big and Open Linked Data is described in [11] as being synonymous to both openness and accountability: "accountability implies answerability for one's actions or inactions and the responsibility for their consequences ; accountability means also taken responsibility for decisions."

In computer science, accountability also appears closely related to transparency. For example, Weitzner et al. write: "Information accountability means the use of information should be transparent so it is possible to determine whether a particular use is appropriate under a given set of rules and that the system enables individuals and institutions to be held accountable for misuse." [19]. Focusing more specifically on datasets, the LiQuID metadata model [21] is defined to make datasets accountable throughout their lifecycle. The following definition is given [21]: "Accountable datasets are datasets about which there is sufficient information to justify and explain the actions on these datasets to a forum of persons, in addition to descriptive information and information on the people responsible for it." In other words, transparency of relevant data is required at all stages of the lifecycle.

Note that in much articles, accountability requires transparency and goes beyond the general need for more information. Relevant and complete information is needed about people's responsibilities, justifications for data use and misuse, as well as sufficient information to verify all of this. Hence, we adopt the following definition.

Intuitive definition 3. *Accountability of a dataset means that "there is sufficient information to justify and explain the actions on these datasets to a forum of persons, in addition to descriptive information and information on the people responsible for it" [21] and "that the system enables individuals and institutions to be held accountable for misuse" [19] of the data.*

Provenance

In [52], transparency is a specific dimension of data quality that is measured according to two elements: data provenance and explanation in order to "interpret the information extraction process" and "verify which aspects of the data determine its results". In their work, data provenance is based on "metadata describing where the original data come from".

In the context of relational databases, provenance can be defined as additional information about where a query result came from or how it was computed [56]. Over the past twenty years, computing data provenance in database systems has been well studied, with some results

that can be adapted to the context of knowledge graphs. However, its computation is not always easy. Research on provenance in scientific workflow systems [57, 58], and more generally in scientific databases [50], enables reproducibility, a concept we discussed below.

The LiQuID model [21], designed to express dataset accountability, considers provenance as a central aspect. In fact, for each step of the data lifecycle, provenance information partly answers the questions who, when, where, how, and what, with at least a description and possibly an explanation. This link is illustrated by comparing the LiQuID metadata model and the PROV metadata model [59]. The PROV model is a recommendation from the W3C that aims to represent the provenance of digital objects, i.e. their origins, their modifications over time, the people who act on them, and so on.

In addition to some dedicated RDF vocabularies, many efforts have been made to propose ways to express provenance information (among other meta-information) within the graph at different scales, either at the statement level, with RDF reification [60], or at the scale of a set of triples with named graphs [61]. Provenance is also depicted as a central element of knowledge graphs in [48], where statements (i.e., RDF triples) are associated with provenance information that specifies who, where, when and how they were made. Among the example constraints applied to a knowledge graph, one concerns transparency. It explicitly requires the provenance of the document that helped create the focused statement.

We adopt the following definition from the PROV recommendation, since it appears to be the more complete definition and applies to any kind of data, dataset and more.

Intuitive definition 4. *"Provenance is a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing" [62].*

Verifiability and verification

Usually, when a source publishes information, it hopes and expects that the reader will accept the information and believe it to be true. In the process of accepting or rejecting such information, its verifiability is an important element. Verifiability enhances trustworthiness [13, 27] because it refers to all available elements that enable the verification of the information. It also helps readers form their own opinion about the information. Verification can be carried out directly either by the reader or by a trusted, unbiased third party, using any means at their disposal. If the conclusion is positive, the information is said to be verified. These notions of verification, verified, and verifiability are therefore very close, but different and complementary. In the following, we adopt the definition below.

Intuitive definition 5. *"Verifiability is the degree and ease with which the data can be checked for correctness" [27].*

The information needed to verify information may vary from person to person, as we all have different abilities, skills and resources. Therefore, it may be more important [13], or at least sufficient, to know that third parties have conducted audits, and to have access to their comments and conclusions. Journalists, fact-checkers, and many others are aware of this. According to [63], third-party verifications increase inferability. It is "the ability to draw accurate conclusions from" information and a necessary condition for transparency [63]. Hence, we retain the following definition.

Intuitive definition 6. *"A verified data is information that has been vetted by a third party" [63].*

Accordingly, the transparency of a source consists in providing enough elements to enable a person to perform an audit. This often corresponds to the way in which the source itself obtained the information, i.e. its provenance [13]. In experimental science, verifiability can be achieved through the reproducibility [50].

Reproducibility & co.

When people talk about transparency in science, and especially in experimental science, they often think about how the experiments were conducted, in what environment, and so on. Indeed, "science thrives on reproducibility" [64]. And to be reproducible, a high level of transparency is required.

In fact, reproducibility has different definitions and is usually understood with (at least) two other 'R' notions: *replicability* and *repeatability*. According to the ACM [65], all three notions require obtaining similar results on multiple trials of an experiment. They differ in who performs the experiment and under what conditions. Repeatability concerns experiments conducted by the same team in the same experimental setup ; reproducibility requires a different team in the same experimental setup ; and replicability involves a different team in a different experimental setup. In the following, the term reproducibility implicitly refers to these three notions.

Hence, transparency is essential for reproducibility in many different fields, from psychology to computer science, including economics and many others. And the reproducibility crisis has highlighted its importance. Thus, there is a growing need for transparency of data, methods, and research materials [66]. In artificial intelligence [67], this takes the form of disclosure of the computer code, parameters and dataset used, but also the whole data processing and training pipeline. In this case, reproducibility partly relies on provenance information. We consider here the ACM definition.

Intuitive definition 7. *Reproducibility means that the experimental result "can be obtained with stated precision by a different team using the same measurement procedure, the same*

measuring system, under the same operating conditions, in the same or a different location on multiple trials." [65].

Privacy

Bertino [22] closely relates transparency to the notion of privacy. According to her definition, transparency is giving access to information, to data, but not just any data: the data disclosed should not expose the personal data itself, but the use of that data. Data transparency is then about providing access to more information in order to support data privacy. That is captured in this definition: "Data transparency is the ability of subjects to effectively gain access to all information related to data used in processes and decisions that affect the subjects."

This link with transparency which supports data privacy may be surprising since privacy is more often presented as the opposite, where it stands against transparency. Indeed, privacy is also the "controlled access to information related to an agent" [48]. In this case, privacy aims at restricting access to data and thus at being less transparent. The opposition between these two concepts can be expressed as follows: "privacy tends to limit or restrict actions over information items, whereas transparency tends to allow (in some cases, mandate) actions over them, which explains the natural tension that exists between the two" [48]. Weitzner et al. [19] makes the following distinctions between these two sides of privacy: privacy rights, as understood by Bertino, and privacy protections, that oppose transparency. They are defined as follows, privacy rights "relate to the collection and use of personal information" and are opposed to "privacy protections that seek to preserve control over, say, one's physical integrity." Still, these two visions may not be contradictory, since Bertino does not require that all personal data be revealed to everyone, and since the definition of privacy only imposes a controlled access, not a closed access.

Secrecy is a concept close to privacy, as it usually tends to limit the disclosure of information. It is also more general, as it is not limited to personal data, but can apply to companies and governments: "government secrecy [is] the logical antonym of transparency" [12].

For the following, we only consider the "privacy rights" aspect. "Privacy protections" are outside the scope of this thesis for two reasons: first, because it goes against transparency, and second, because we will only consider open knowledge graphs, which are not expected to contain private information. For privacy rights, we adopt the following definition, extracted from the definition of data transparency from Bertino.

Intuitive definition 8. *Privacy rights means that subjects are able to acquire "all information related to data used in processes and decisions that affect the subjects" [22].*

Explanation & Understandability

According to Firmani et al. [52], transparency can be divided into *data provenance* and *explanation*, where explanation "describes how a result has been obtained". It seems obvious that a good explanation increases transparency. But it is also very dependent on the one receiving it. Indeed, to be useful, an explanation must be adapted to the audience. This leads us to consider the concept of understandability. It "emphasizes on explaining results or processes to make these transparent to some audience" [68]. Understandability also increases transparency, but it is still very dependent on the audience.

In this thesis, we assume that we do not know the audience, hence explanation and understandability are not directly within our scope. Moreover, this definition of explaining is partly included in our definition of provenance, the difference lies in how the description is made: for explanation and understandability, it explicitly requires that it be human readable and human understandable.

2.2.3 Proposal of a global picture

We have presented various definitions of transparency and its related concepts. Transparency definitions present some similarities: they require access to more information (openness and accessibility) and underline that different types of actors are concerned, such as information providers and consumers. Providers give access to the data sources which contain data and metadata. Consumers, or simply users, have their own interpretation of transparency and some more precise expectations which depend on the domain and the context. We can see all other related concepts, such as accountability or reproducibility, as specific contexts that require transparency.

Figure 2.2 summarizes the relationship between transparency, these concepts, data sources, and users. Each of these contexts specifies needs of data and metadata, some of which are specific, while others are more generic such as provenance or certification information. Central to the picture is transparency, which relies not only on openness (increasing the data exposed) and accessibility (which guarantees that these data can be read or worked with), but also on the completeness of the data actually available in the source with respect to the data and metadata that constitute the user's need. Therefore, assessing the transparency of a source amounts to asking the question: "Is the set of accessible information complete enough to satisfy the user's need ?" In this line, we propose a new definition of the transparency of a data source that explicitly takes into account its contextual aspect.

Intuitive definition 9. *A source's transparency w.r.t. a given need of observation consists in giving any observer access to all the suitable metadata regarding this need.*

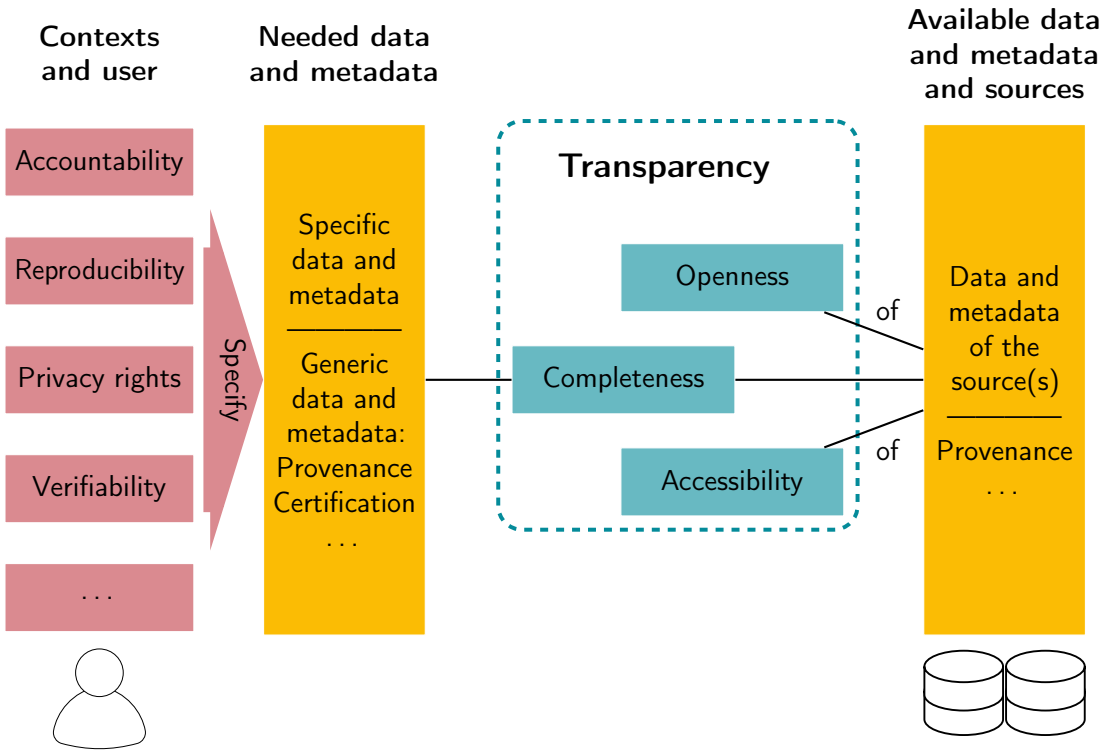


Figure 2.2: Transparency and companion concepts

This formulation combines the elements introduced by the different definitions in Section 2.2.1. First, transparency can only be considered in practice with respect to a *need of observation* that should explicitly specify what information to disclose, and which depends on the context, the domain. Therefore, to obtain transparency, information provided by the source must meet the need in the most appropriate and complete way. For example, if the need expressed is a need of reproducibility of a numerical process, then transparency consists in giving, among other things, full access to the source code, to the data used and to the environment of execution. But there is no interest in observing information that is not related to the need, such as the employer of the people responsible for the numerical process or their age: it is not suitable information. Then, transparency can only be truly attested by an external observer, who may choose or define the need.

2.2.4 Specifying needs

Our definition of transparency relies on the definition of observation needs to explicitly describe what information a source should provide. Designing an observation need is both crucial and delicate, as it serves as the basis for evaluating the transparency of a source. There is a risk

of either being too lax and declaring a source transparent without extensive checks, or, on the contrary, going too far and declaring any source non-transparent by requiring it to satisfy every conceivable need. Finding the right balance can be difficult.

There are two types of observation needs. They are either specific to one person and for a particular purpose, or prototypical and more consensual to a group of people. We focus only on the second type. According to the vision presented in Figure 2.2, it is possible to define a need with respect to one or more given contexts, such as accountability and/or verifiability for instance.

In the light of the previous sections, we do not believe in the existence of a general observation need that would cover all transparency in absolute terms. Therefore, several needs must be defined, either specific to one context or more transversal and covering multiple contexts, possibly less in depth than the more focused needs. The latter would provide a more generic view, but less precise than the former. To define a prototypical need, one must rely on existing studies focused on the given context(s) to identify requirements and state them in clear and explicit terms. In this line, accountability is an interesting context to study for several reasons. First, it is an important aspect of transparency, sometimes considered synonymous with it. Then, a systematic study, conducted to define the LiQuID metadata model [21], provides clear requirements of what is needed to be accountable.

Once an observation need is explicitly defined with a list of precise requirements, it becomes possible to evaluate the transparency of a data source. In our case, we focus on knowledge graphs. Therefore, taking advantage of the interoperability offered by the semantic web, the aspect of accessibility and openness comes down to the queryability of the information sought. And the transparency of a knowledge graph regarding a given observation need is the completeness of the information that can be retrieved with a query with respect to the requirements specified by the need. Other studies use queries to evaluate knowledge graphs, such as data quality measures for instance. Indeed, the latter can be seen as other kinds of needs that also express requirements but not specifically dedicated to finding information.

2.3 Data quality measures of knowledge graphs, assessment frameworks and FAIR principles

In this section, we analyze existing measures for linked data. The objectives are (1) to study the measures or metrics that are close to transparency or that can help measure it, and (2) to identify the existing frameworks for evaluating a knowledge graph. To this end, we explore the studies that have highlighted the many facets of data quality for knowledge graphs. Then, general monitoring tools allow to assess and draw profiles of SPARQL endpoints. Finally,

some measures have recently emerged to ease the adoption of good practices, such as the FAIR principles [14]. All these studies can be seen as part of data quality, each with its own focus and its own requirements. In this way, they can be seen as some kind of prototypical observation needs that goes beyond the scope of transparency. We present here the different measures for knowledge graphs and semantic resources. Finally, we present some frameworks that enable the assessment of knowledge graphs.

2.3.1 Terminology used

Before introducing the existing measures, we clarify the terms that will be used throughout this thesis. We use *metric* to refer to a procedure for measuring a given criterion, a single quality characteristic, or a requirement, it is a concrete and usually implemented quality indicator. For instance, a metric concerning the availability of a SPARQL endpoint checks "if a `void:sparqlEndpoint` is specified for a dataset and if the server responds to a SPARQL query" [45]. A *measure*, is a more abstract concept that consists of an aggregated set of metrics, for instance a measure of data quality relies on many metrics. A *measurement* is the evaluation of a dataset (or a knowledge graph) with respect to a given metric, it is associated with the *value* obtained by the dataset on the metric. An assessment *score* is the overall score obtained on the measure and computed based on the measurement of each metric using a scoring function. These choices rely on the Data Quality Vocabulary [45] and are consistent with data quality studies [13, 27, 69]. Finally, an evaluation framework, or *assessment framework*, is an implemented tool that allows to evaluate datasets based on a single measure, or that can be used or extended with several measures.

2.3.2 Evaluating knowledge graphs

There are several studies that audit, analyze and measure KGs, including data quality studies, FAIR principles and various monitoring tools. They all give an insight into the quality of KGs, with their own particularities. While these three different types of studies are presented here, the metrics and criteria they use are discussed in more detail, compared, and categorized in the following subsection.

Data quality studies

The evaluation and monitoring of knowledge graph quality is an active field of research [70, 71]. Data quality is "commonly conceived as fitness for use for a certain application or use case" [13]. Two points seem important in this formulation: the notion of "fitness for use" and

the fact that this notion is to be considered from the point of view of a particular application or use case. Much of the quality literature has been devoted to specifying different facets of "fitness for use": availability, accuracy, conciseness, completeness, understandability, timeliness, and so on. As a result, existing measures provide extensive lists of metrics covering these concerns. They mainly focus on examining general elements that can be beneficial to any application, but their importance varies from one to another.

To organize all these characteristics, Wang and Strong [72] introduce a framework for assessing data quality. It was then adapted to knowledge graphs [13, 27, 69]. In this framework, data quality is divided into several "categories". Each category is subdivided into "dimensions", which contain one or more "criteria". Finally, each data quality criterion is associated with a metric in order to measure it on a given knowledge graph. The most common data quality categories are the following [27, 69, 72].

- The "*intrinsic category* denotes that data have quality in their own right" [72] and is therefore "independent of the user's context" [13].
- The "*contextual category* highlights the requirement that data quality must be considered within the context of the task at hand" [72].
- Then, the "*representational category*" evaluates "how well the data is represented in terms of common best practices and guidelines" [69].
- Finally, the "*accessibility category*" includes "aspects related to the access and retrieval of data to obtain either the entire or some portion of the data for a particular use case" [13].

Zaveri et al. [13] add two new categories, the "*trust category*", which focuses "on the perceived trustworthiness of the dataset", and the "*dataset dynamicity category*", which covers dataset's "freshness over time, the frequency of change over time and its freshness over time for a specific task".

Among the previously mentioned studies for evaluating the data quality of knowledge graphs, Zaveri et al. [13] conduct a systematic review and thus propose a detailed theoretical measure consisting of 6 categories, 23 dimensions, and 96 metrics. Note that these metrics are not implemented, but the procedures are precisely described. Debattista et al. [69] complement this important work by partially implementing its metrics, excluding those that are subjective or too difficult to implement. They aim at a general framework, so they evaluate 130 datasets from the Linked Open Data Cloud[12] with 27 metrics of 13 dimensions and 4 categories. In parallel, Färber et al. [27] evaluate five popular knowledge graphs, namely DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, with 34 metrics in 11 dimensions and 4 categories, partly based on [13]. Other studies reuse these works to measure or improve

[12] https://lod-cloud.net/

the quality of knowledge graphs. For instance, some of these metrics are reimplemented as SPARQL queries in the IndeGx framework [17], which aims at indexing knowledge graphs and providing various descriptive elements. In addition, other studies use them not only to evaluate but also to monitor and improve the data quality of knowledge graphs during their construction [70] or throughout their life cycle [71].

Monitoring tools for SPARQL endpoints

Finally, there exist some catalogs of SPARQL endpoints that provide more information about them through metrics but with a more limited scope than quality studies. They aim to provide information about the usability of the SPARQL endpoints with a main focus on their availability and performance over time. More precisely, SPARQLES [35] measures discoverability, interoperability, performance, and availability. SpEnd [73] is very similar to SPARQLES with many common metrics. YummyData [74], specialized in biomedical KGs, proposes to monitor the quality of SPARQL endpoints using what is called the Umaka score. It is based on six dimensions: availability, freshness, operation, usefulness, validity, and performance. They both use metrics closely related with existing dimensions of data quality or FAIRness. There are some other approaches not in our scope. For instance, SPORTAL [36] mainly focuses on computing statistical information to provide a description of endpoints but is not interested in assessment metrics.

FAIR principles

Finally, the FAIR Guiding Principles [14] have been introduced in 2016. They state, in very general terms, guidelines for the production and sharing of digital objects regarding *Findability*, *Accessibility*, *Interoperability*, and *Reusability*. The FAIR principles are not specific to knowledge graphs, they are intended to apply to any data, or more generally, to any digital object, including software [75] and workflows [76]. Among other things, they aim to help producers improve their datasets in order to "enhance the reusability of their data holdings" [14]. These generally stated principles have then been implemented in different contexts for various types of digital objects to help providers increase the FAIRness[13] of their data [77, 78] and to evaluate the FAIRness of a dataset [79–82]. While FAIR principles obviously contribute to the quality of a dataset through accessibility, interoperability, and some aspects of reusability, they also bring a new insight with the findability.

For the evaluation of knowledge graphs in particular, several approaches exist. O'FAIRe [16] and FOOPS! [83] evaluate ontologies according to their definitions and the metadata they are

[13] FAIRness: compliance with the FAIR principles.

annotated with. FAIR-Checker [15] and F-UJI [84] are online tools for evaluating RDF resources according to the metadata provided by their dereferencing and the RDF data contained in the HTML pages documenting them. O'FAIRe and FAIR-Checker have initially been designed to be used by the life sciences community, so they notably include the referencing of the evaluated object in specialized databases for the life sciences community, such as BioPortal [10]. However, they are generic enough to be used in other contexts. F-UJI is oriented towards scientific research data objects and relies on databases such as DataCite [85]. Some of these tools, or the measures on which they are based, have been compared in various studies, such as [86, 87].

In the wake of the FAIR principles, other guidelines have been proposed to complement them. The CARE principles [88] promote Collective benefit, Authority to control, Responsibility, and Ethics, for indigenous data governance. The TRUST principles [89] have a specific focus on Transparency as well as on Responsibility, User focus, Sustainability and Technology. In the context of this thesis, we only focus on the FAIR principles. Indeed, they have been more extensively studied by the community, with works to measure and assess the various the principles.

2.3.3 Classification of the existing metrics

All these auditing tools provide metrics or guidelines. The FAIR principles and monitoring tools add some precision to data quality and even new dimensions. Among all these tools, three different types of metrics seem to appear. The first one is independent of the data and concerns the system hosting the RDF dataset. Then, metrics about the form of the data are not interested in the content itself nor in the meaning of the data, but in how it is written. Finally, some metrics focus on the content, the information conveyed by the data. It is important to notice that these three types are not independent. For instance, vocabularies concern both the content, because they provide information about the domain covered by the KG, and the form of the data, because they give insight into the schema and structure of the KG. Unless otherwise specified, data is understood as being both data and metadata.

Metrics concerning the hosting system

The first type of metric focuses on the system that hosts the dataset and makes it available. Hence, it mainly concerns accessibility, both in data quality studies and in FAIRness based approaches. The latter is mainly concerned with the protocol and conditions of access to the dataset, while the former brings some precision specific to knowledge graphs. For instance, a SPARQL endpoint should be available as well as a dump. Data quality studies also provides some metrics of performance of the dataset service (latency, throughput, scalability) [13].

In the FAIR principles, findability includes other principles relying on this system, including

that data must be "registered or indexed in a searchable resource" [14]. Indeed, many metrics of FAIRness rely only on sending HTTP requests that are not interested in the data [81].

Most of the metrics performed by SPARQLES and YummyData also focus on the hosting system by sending HTTP requests and SPARQL queries. Indeed, SPARQLES sends generic queries, from the simplest to the longest to compute, to measure availability and performance, and queries that check compliance with SPARQL 1.0 and SPARQL 1.1 features to measure interoperability. Similarly, YummyData uses HTTP requests to test content negotiation and measure operation, and queries to estimate availability and performance. The fact that monitoring tools are mostly concerned with the hosting system and not the data can be explained by the fact that they aim to inform about the availability and usability of a list of KGs, which starts by these technical considerations.

Metrics concerning the form of the data

Metrics about the form of the data are not interested in the information conveyed nor in the meaning of the data, but in how it is written: the syntax, the schema used, the compliance with of rules, but also the number of triples, properties, and so on.

Consistency, conciseness, amount of data, and many other dimensions are associated with metrics concerning the form of the data. For instance, they rely on computing the number of triples that do not respect some RDF inference rules or OWL rules, checking the use of some vocabularies and some properties, counting the number of unique objects, etc. YummyData does the same for the operation dimension, counting the number of properties, labels, classes, and datatypes, and for a metric of validity. This results in a majority of statistical metrics about the dataset. Indeed, in the survey by Zaveri et al. [13], "most of the metrics take the form of a ratio, which measures the occurrence of observed instances out of the occurrence of the desired instances". In the FAIR principles, these considerations are supported by interoperability, which first requires a "formal, accessible, shared, and broadly applicable language for knowledge representation" and the use of "vocabularies that follow FAIR principles" [14].

Metrics concerning the information conveyed by the data

The last type of metric requires specific information within the dataset, so these metrics concern the information conveyed by the data or metadata. For instance, simple elements such as the "presence of the title, content and URI of the dataset" are mentioned in data quality studies [13]. The presence of a license is also a common requirement. The reusability component of the FAIR principles asks that "(meta)data are released with a clear and accessible data usage license" [14]. It is also a dimension of the accessibility category of data quality [13].

Similarly, FAIR principles and data quality studies both require provenance information. It can be seen as part of reusability (FAIR principles) or more specifically as the provenance dimension introduced within the contextual category in [69].

All these metrics are very specific to certain information that is considered very essential. But in some cases, the information required is not as consensual as the need for a license, or depends more on the context or domain. In the contextual category of data quality studies, the completeness dimension fills this gap. It "refers to the degree to which all required information is present in a particular dataset" for a given task [13]. As defined in [27], it is divided into three metrics: schema completeness, property completeness (or column completeness), and population completeness. Population completeness (respectively schema completeness) uses a gold standard that defines the entities (respectively, the classes and properties) that should be represented in the KG. Column completeness evaluates whether all entities of a given class have a value for a given property.

A survey focusing on Knowledge Graphs Completeness [90] identifies four other types of completeness: interlinking completeness, currency completeness, labeling completeness, and metadata completeness. The first three are mainly assessed by statistical metrics, that respectively measure how much instances are interlinked with other KGs, the availability of valid elements over different time periods, and the presence of labels. Metadata completeness is defined as "the degree to which metadata properties and values are not missing in a dataset for a given task" [90].

Coming back to transparency, it can be seen as part of the contextual category of data quality, since it must always be considered within a given context. In addition, the evaluation of transparency consists in checking the completeness with respect to a need. In addition, since transparency mostly concerns metadata, it is a kind of metadata completeness. To this end, we believe that the existing metrics are not sufficient to evaluate KG completeness with respect to the information defined as necessary. Indeed, these metrics are partial and do not allow to verify the presence of a multitude of diverse information. We do not limit ourselves to schemas or values of a specific property, we would like to use any query, without limitations. Moreover, some of them require a gold standard to be provided, whereas we suppose that the information is not known in advance, only the kind of information desired (e.g. the authors).

2.3.4 Available assessment frameworks

Several frameworks are available for evaluating knowledge graphs. To identify which ones can be reused for new evaluations, such as transparency or accountability, we focus only on those with an open implementation.

Sieve[14] [91] is a framework for assessing the quality of Linked Data in a customizable way, but it requires "a high degree of user involvement" and "much time for understanding the required XML file structure and specification" [13] in order to configure the tasks to be performed. It provides several data quality metrics, including one for property completeness evaluation. The idea of a generic framework for quality evaluation has been proposed by Debattista et al. [92] with their framework named Luzzu[15]. It was used to evaluate the quality of datasets from the LOD cloud [69]. It allows to evaluate SPARQL endpoints or RDF dumps, but the use of the former is not recommended with this tool. Luzzu and Sieve enable users to select metrics among those defined and to declare new ones. However, neither relies on SPARQL queries, metrics of Luzzu are implemented as Java classes, and Sieve uses XML configuration files that refer to an implementation in Scala.

Other frameworks are specifically designed for SPARQL endpoints, such as the monitoring tools YummyData[16] [74] and SPARQLES[17] [35], which allow the evaluation of some given quality aspects. Some of their metrics rely on SPARQL queries. Both tools do not aim to be extended with other metrics.

The IndeGx framework[18] [17] proposes several data quality evaluation metrics for knowledge graphs and endpoints. Its primary use case is to build an index of KGs by extracting and computing various information about them, such as metadata, statistical information, and quality scores. All these data are stored in RDF. It relies on a powerful SPARQL-based test suite that is easily extensible with new queries through a set of declarative rules. These are expressed in RDF, with a *test* on one side, which is a query to send to the SPARQL endpoint, and an *action* on the other side, wich expresses what to write in the resulting RDF graph based on the result of the test. Therefore, it enables querying many KGs endpoints with multiple queries. As for SPARQLES and YummyData, IndeGx only works with SPARQL endpoints.

2.4 Conclusion

Our study of transparency shows that it appears to be very contextual, prone to interpretation, and without any real consensus. Therefore, we have suggested a new definition of transparency: A source's transparency w.r.t. a given need of observation consists in giving any observer access to all the suitable metadata regarding this need. In this way, transparency is part of

[14] https://github.com/wbsg/ldif/tree/master/ldif/ldif-modules/ldif-sieve
[15] https://github.com/Luzzu/Framework
[16] https://github.com/dbcls/umakadata
[17] https://github.com/pyvandenbussche/sparqles
[18] First version: https://github.com/Wimmics/dekalog ; Second version: https://github.com/Wimmics/IndeGx

data quality, as it also aims to measure the "fitness for use" of the dataset for a particular purpose. However, it focuses on a certain type of metrics only, those that request and search for given information. In addition, we did not identify in the literature any specifications or recommendations on which to rely to measure transparency. Similarly, we did not find any measures focusing on transparency. However, in light of the definitions of transparency and related concepts, we have come to the conclusion that defining a universal measure of transparency is neither possible nor desirable. Indeed, our definition of transparency requires the definition of clear and explicit needs to be able to measure it.

To propose a prototypical need, one of the concepts related to transparency has caught our attention. Accountability is a particular context of transparency, perhaps the closest since it is sometimes considered synonymous with transparency. There is no measure for accountability of knowledge graphs. However, it is the subject of a detailed and systematic study [21] that defines precise requirements in this respect. For all these reasons, we propose to define a (partial) measure of transparency by defining a particular need of accountability and then to evaluate KGs against this need, i.e. to assess to what extent they provide the required information. Therefore, we will propose such a measure of accountability in Chapter 3.

On the other hand, the study of the measures shows that there is no universal data quality measure for Knowledge Graphs. Each measure is divided into many metrics that vary from one to another. As a result, users can often find specific metrics that fit their context and objectives. In addition, many of these measures follow a hierarchical organization. Some of these are very close, such as those on data quality, which often reuse the hierarchy defined by Wang and Strong [72] or the metrics of Zaveri et al. [13]. Other measures are built on the same foundation, such as all the FAIRness measures. Since they do not always consider the whole set of principles, it would be interesting to compare them and identify the shared and missing parts. These considerations lead us to define theoretical basis for expressing, manipulating, and comparing measures and to propose a related tool enabling anyone to build their own tree-like measure, by eventually reusing existing metrics.

3

Assessing Knowledge Graphs Accountability

Contents

THIS chapter explores a particular aspect of transparency: accountability. We define a measure of the accountability of knowledge graphs through a set of requirements stated in natural language and associated with metrics expressed in SPARQL. We then evaluate many knowledge graphs that are publicly available on the Web via SPARQL endpoints. Finally, we compare the accountability measure with existing measures of data quality and FAIRness.

3.1 Introduction

Knowledge Graphs (KGs), and Linked Open Data in particular, enable the generation and exchange of more and more information on the Web. This abundance of easily accessible data on the Web offers many opportunities for researchers, companies, and ordinary citizens. However, in order to properly and legally use and share these data, it is important to know some information about a knowledge graph, such as for what purpose it was created, by whom, etc.

Such meta-information contributes to the correct use of KGs but not only. Designing systems enabling individuals and institutions to be held accountable also enhances trust in the data and it is therefore increasingly important [19]. Hence, we focus on this concept very close to transparency: accountability, and more specifically dataset accountability. It requires the provision of information about actions on the dataset in addition to "descriptive information and information on the people responsible for it" [21]. As with transparency, this information should be easily accessible [12] and queryable.

For instance, the GDPR (General Data Protection Regulation) aims to protect personal data. To this end, Article 17 about the "right to be forgotten" states that the "subject shall have the right to obtain from the controller the erasure of personal data concerning him or her"[1] unless it falls under the right of freedom of expression and information. Therefore,

[1] https://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04

any KG holding personal data, such as Wikidata, should provide contact information of that controller, i.e. a person responsible for the data, and ideally allow users to access it directly via its SPARQL endpoint. As another example, to avoid misinterpretation and to improve the (re)use of the data, it is often necessary to know for what purpose they were created, and for whom the data are intended. For instance, in some database mainly dedicated to teaching purposes, such as the MONDIAL Database[2], some inaccuracies can be tolerated (or even desired). However, it cannot be reused without precaution for other purposes. Therefore, it should indicate its intended audience or its expected use. However, when querying its SPARQL endpoint, this information is not available.

Accountability ensures that this kind of information of major interest is effectively available. Several studies are already looking for metadata, either as some particular aspects of data quality [13, 27, 69], or as some requirements of the FAIR principles (Findability, Accessibility, Interoperability, Reproducibility) [14, 79, 84, 93]. Yet, they neither take into account the information used in the two previous examples nor several other pieces of accountability information. This highlights the importance of accountability as a specific and distinct characteristic and the importance of evaluating this aspect.

Hence, in this chapter, we propose a new framework, KGAcc, dedicated to the assessment of RDF graphs accountability. In light of the previously mentioned elements, we believe that accountability metadata should be easily findable. Since software agents are particularly present in the semantic web, metadata about KGs "need to be available in a machine-readable format" [27]. In addition, KGs have the ability to store and represent metadata in the same way as data. Putting these things together, we consider that this information should be present and searchable within the data of the KG itself. Although we recognize that this may hinder access to some metadata, or may not conform to all practices, such as the FAIR Digital Object approach[3], which recommends a clear and explicit separation between data and metadata. Indeed, we will not look for metadata embedded in web pages such as FAIR-Checker [15], nor will we look for an external VoID file using the `/.well-known/void` mechanism[4], nor a YAML file. This choice is also guided by the context of the ANR project DeKaloG and the construction of an index by querying SPARQL endpoints. Therefore, we will only look for accountability metadata with SPARQL queries. Furthermore, we are interested in whether the information is present or not, not how it is expressed, i.e. regardless of the vocabulary used. Given these constraints, the framework consists of organized accountability requirements, expressed as questions in natural language and as SPARQL queries, and a measure of accountability. We experiment it on many KGs offering a publicly available SPARQL endpoint. Our accountability measure gives

[2] https://www.semwebtech.org/mondial/10/
[3] https://fairdo.org
[4] https://www.w3.org/TR/void/#well-known

an indication of the accountability of KGs to dataset users and providers. It aims to guide users in choosing one KG over another, and to help providers identify ways to improve their datasets.

To define such a measure, several questions arise, such as, what metadata are required? How to evaluate heterogeneous KGs? First, to define the requirements, we rely on the LiQuID metadata model which focuses on dataset accountability in general [21]. It provides an explicit list of accountability requirements expressed in natural language. The problem, then, is to adapt this model to the particularities of knowledge graphs and to define the requirements in terms of SPARQL queries. They are designed taking into account the expressiveness of the most common vocabularies. To evaluate the KGs, we use the SPARQL-based test suite of the IndeGx framework [17]. In addition, to illustrate the specificities of this measure, we compare it with several measures of data quality and FAIRness [13, 27, 69, 79, 93].

The main results of this chapter have been published in [94] and in [95].

The chapter is organized as follows. Section 3.2 describes accountability in more detail and existing studies on it. We define the KGAcc framework in Section 3.3, from the definition of the requirements in natural language, to their expression as SPARQL queries, and finally to the computation of the overall accountability score. Then, Section 3.4 is devoted to the description of the methodology and tools for evaluating RDF graphs. Section 3.5 presents the evaluation campaigns and the results thus obtained. We then compare our accountability measure with the existing assessments of knowledge graphs in Section 3.6. Finally, we conclude in the last section.

3.2 Accountability: overview and existing model

Accountability requires that there is sufficient information to describe the data [21], the actions on the data, from its creation [96] to its use [19], and the people responsible for these data as well as these actions [19, 21]. It may concern different levels of the information system, such as information accountability [19, 96], system accountability [97], and dataset accountability [21]. Since knowledge graphs are a specific type of dataset, we focus on the latter to evaluate the accountability of knowledge graphs. As defined in Chapter 2, accountability of a dataset means that "there is sufficient information to justify and explain the actions on these datasets to a forum of persons, in addition to descriptive information and information on the people responsible for it" [21] and "that the system enables individuals and institutions to be held accountable for misuse" [19] of the data.

As discussed in Chapter 2, the accountability of knowledge graphs can be considered as part of their data quality, and more specifically, measuring the accountability of KGs is a special case of assessing metadata completeness. Many studies consider the presence of metadata to evaluate knowledge graphs. A few metrics of the data quality studies of KGs

focus on metadata [13, 27, 69]. For instance, provenance information is required by a metric on trustworthiness. The FAIR principles [14] are also interested in metadata through the F2 principle of findability and the principles of reusability, which explicitly mention a license and provenance information. Therefore, the required metadata may overlap between accountability, data quality, and FAIRness, while having its own specificities. Because of the high variability of the actual implementations of these metrics and principles, we confront them with our own requirements at the scale of the RDF properties in Section 3.6.

To our knowledge, there is no measure of the accountability of knowledge graphs. There exists an ontology for capturing accountability information of artificial intelligence systems [97], but not dedicated to the description of the knowledge graph itself. Therefore, the closest model designed for this purpose is the LiQuID metadata model [21], which considers datasets in general, but not knowledge graphs specifically. It offers a way for datasets to represent accountability metadata throughout their life cycle. The model has been validated using a real-world workload based on an expert survey and a list of guidelines from existing regulations: the Federal Trade Commission and the General Data Protection Regulation (GDPR). To our knowledge, it is the only one to provide such a precise and explicit list of accountability requirements, presented in the form of questions describing the model. For this reason, we use LiQuID as the basis for defining our accountability measure for KGs.

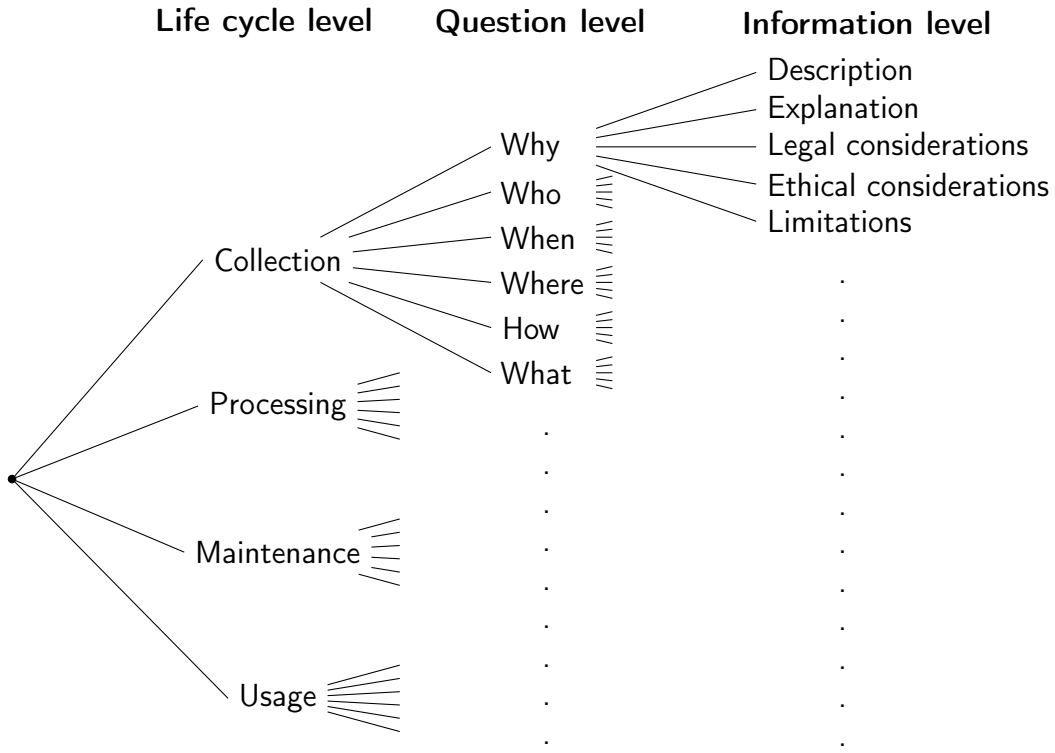


Figure 3.1: The LiQuID metadata model [21]

The LiQuID metadata model relies on a hierarchical structure. It follows a systematic approach, aiming to cover all aspects of accountability. First, it considers each step of a dataset's life cycle: data collection, data processing, data maintenance and data usage. Then, each step is structured according to different question types: why, who, when, where, how and what. Finally, each question type is divided into different fields of information level: description, explanation, legal and ethical considerations and limitations. The model is illustrated by Figure 3.1. Each branch of this hierarchy is precisely described by one or more questions provided by the authors [98]. For instance, for "data processing", question type "when", the question associated with the field "description" is "On what date(s) or time frame(s) has the data been processed?". Other questions are provided in Appendix A.2. The LiQuID approach therefore requires a large amount of very detailed information, with a total of 207 questions representing what data sources should expose to be as accountable as possible.

3.3 KGAcc framework: Requirements and measure

In this section, we define the KGAcc framework. It consists of all elements used in the definition of our accountability measure, from the hierarchy and questions, to the SPARQL queries and finally, the measure itself. First, we define the requirements of knowledge graphs accountability, i.e. the precise information that KGs must contain. Requirements are expressed as questions in natural language that KGs must answer. To be as unambiguous as possible and to enable automatic evaluation of KGs, we go one step further and express these requirements using SPARQL queries. Finally, we formally define a measure of accountability.

Our proposal is based on the LiQuID metadata model [21]. To illustrate the use of this model, precise questions are provided [98]. They specify what a dataset must answer to be considered accountable. LiQuID is not specific to any type of dataset, so it requires information that is very general: potentially difficult to query, or irrelevant for open KG, such as some legal aspects or related to personal data. Therefore, it is necessary to adapt it to the context of KGs. The different steps of designing the KGAcc framework and evaluating and calculating the accountability score of KGs are illustrated in Figure 3.2.

3.3.1 Adaptation of LiQuID for knowledge graphs

Ideally, to assess the accountability of a KG, we should consider all LiQuID questions. However, it is not possible for all questions to be adapted for KGs and translated into SPARQL queries. Therefore, we face two problems: selecting those that can be adapted to KGs regarding the expressiveness of the existing vocabularies, and then making them more precise

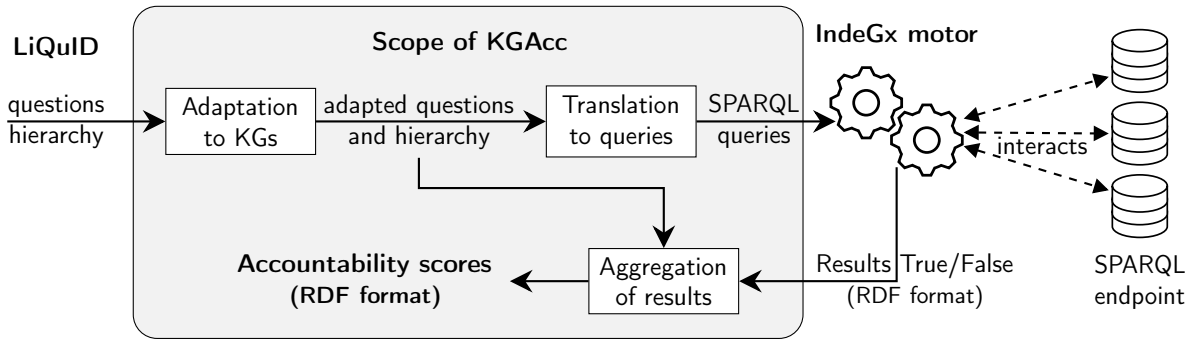


Figure 3.2: Process to define and measure knowledge graph accountability

and specific to KGs if needed.

Indeed, as shown by Oppold and Herschel [21], the two general metadata models Dublin Core[5] and PROV [43], cannot cover all the fields proposed by LiQuID. According to them, both models "contain few fields, some of them too general to be mapped to specific LiQuID fields". We make the same observation with other general metadata models used in KGs, especially if the task is not to express the information, but to query it. As a consequence of this lack of expressiveness, some questions cannot be translated into queries. As an example, some fields of the information level require too specific information, such as "Why is it lawful to collect this kind of data?" or "Why is it ethical to create a dataset for this cause?". To our knowledge, there is no vocabulary for expressing this specific information in a KG. As another example, some questions issued at different steps of the life cycle are not distinguishable in RDF. This is in particular the case for the collection and processing steps.

Faced with these difficulties, one possible strategy would be to consider all questions as necessary, knowing that some of them will never be answered. However, we believe that the current limitations of vocabularies should not have a negative impact on the evaluation of knowledge graphs. First, because they are not responsible for this, and second, because it would significantly lower their score, making the scoring scale inappropriate for comparison and making it more difficult to identify what needs to be improved with a score of zero on both the questions that cannot be answered and the ones that KG could answer but did not. Therefore, we opt for a softer strategy in which the maximum score of accountability is attainable: we evaluate KGs only on the basis of the information they are able to provide. It consists in keeping only questions compatible with the most common vocabularies of the semantic web. Therefore, we make the following adaptations: *(i)* only the field "description" of the information level is considered, *(ii)* the data processing step of the life cycle level is merged into the data collection step, *(iii)* the question types "why", "what" in "data collection" and "what" in "data

[5] https://dublincore.org/specifications/dublin-core/dcmi-terms/

maintenance" are not considered, and *(iv)* two questions concerning the exact methods and tools used for creation and maintenance are not considered in favor of more flexible questions concerning the methodology or procedure only. The resulting hierarchy is shown in Figure 3.3. As for the rest of the chapter, we omit the last level, as it only contains the "description" element. All LiQuID questions of this field "description" of the three considered life cycle steps are listed in the Appendix (A.2, A.3, A.4). Among these, the LiQuID questions maintained for accountability evaluation are those with associated questions in the context of KGs.

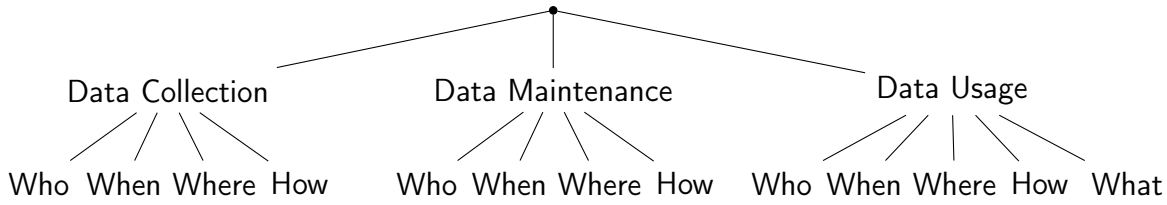


Figure 3.3: The KGAcc hierarchy, adapted from LiQuID [21] to fit the context of knowledge graphs

This definition of the accountability requirements, driven by the desire to ask reasonable questions, leads to a core set of 23 LiQuID questions, out of the 207 LiQuID questions. While this seems to be a significant gap, most of the removed questions come from fields other than the main field "description" in the information level, and so are less important. Then, the remaining questions are adapted to the context of KGs to define the KGAcc framework. We make them more precise, and divide them into smaller parts, so they focus on only one element each. For instance, LiQuID question "Where is the data set published/available?" splits into "What is the webpage presenting the KG and/or allowing to gain access to it?" and "Where to access the KG (either through a dump or a SPARQL endpoint)?". This precision is made as faithfully as possible, with the aforementioned limitations. Table 3.1 illustrates this adaptation. Therefore, the KGAcc framework results in 30 questions: 5 for "data collection", 5 for "data maintenance", and 20 for "data usage". The KGAcc hierarchy and questions are represented in Figure 3.4, page 41. All correspondences between the original LiQuID questions and their adaptation are available in the Appendix (A.2, A.3, A.4).

3.3.2 SPARQL implementation of the questions

Once the questions have been defined, each of them is translated into a SPARQL query or a succession of SPARQL queries. Since queries are associated with questions for which we require an answer but do not care about what it is, they are "ASK" queries. The answer TRUE is considered a success, as it means that the desired information is present and accessible within the KG. On the opposite, the answer FALSE or an error (e.g., timeout exception) is a failure, as it means the KG is unable to provide the wanted information.

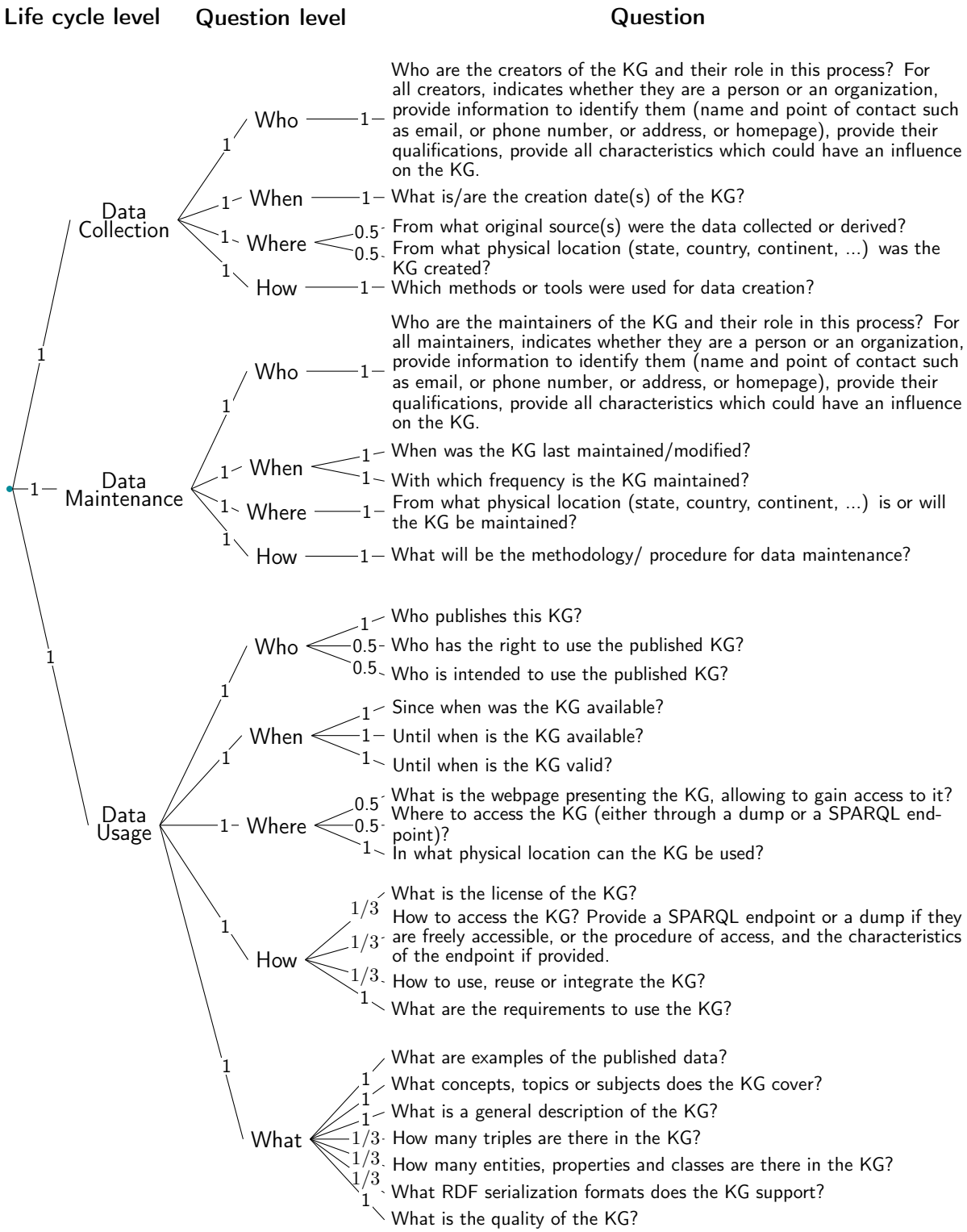


Figure 3.4: The KGAcc hierarchy of requirements of accountability

Table 3.1: Excerpt of accountability requirements concerning Data Usage: Original questions from LiQuID and the adapted ones in the KGAcc framework

	Questions from LiQuID	KGAcc Questions	Weight
Who	Who publishes this data set?	Who publishes this KG?	1
	Who has used/ can use the published data set?	Who has the right to use the published KG?	1/2
		Who is intended to use the published KG?	1/2
When	When can/ was the published data set be used?	Since when was the KG available?	1
	When is it available?	Until when is the KG available?	1
	Until what point in time is it valid?	Until when is the KG valid?	1
Where	Where is the data set published/ available?	What is the webpage presenting the KG and/or allowing to gain access to it?	1/2
		Where to access the KG (either through a dump or a SPARQL endpoint)?	1/2
	Where (place, geographically) can the published data set be used?	In what physical location can the KG be used?	1

In addition, all the questions concern the KG to evaluate, so the associated queries must also characterize the KG itself. In other terms, it has to be the subject of all our queries, or more precisely, its IRI. We will discuss how to identify it later, here it is left generic so that it can be reused in any context.

So, to translate questions into queries, it is necessary to identify the associated terms in the existing vocabularies. As mentioned before, there is no vocabulary dedicated to accountability, nor any guidelines on how to express such information. However, there is a wide variety of languages and several ways to describe a dataset in RDF. Therefore, we do not favor the use of one vocabulary over another: we are interested in whether the information is there or not, not how it is expressed. Hence, we use more than ten vocabularies of reference, chosen regarding their relevance to describe datasets and concepts around: VoID [34] is used to express metadata about RDF datasets. DCAT[6] and DataID[7] allow the description of datasets and catalogs of datasets. SPARQL-SD [40] enables to describe SPARQL endpoints. These vocabularies rely on other general vocabularies, the Dublin Core, FOAF[8] and SKOS[9]. We also

[6] https://www.w3.org/ns/dcat
[7] http://dataid.dbpedia.org/ns/core
[8] http://xmlns.com/foaf/spec/
[9] https://www.w3.org/TR/skos-reference/

use PROV-O and PAV [44] for provenance issues. DQV[10] is used to describe the quality of datasets. Finally, we use schema.org a very general and widely used vocabulary, and some specific vocabularies used only for licenses[11], such as Creative Commons[12].

In the rest of this chapter, the following prefix are used:

```
PREFIX dataid: <http://dataid.dbpedia.org/ns/core#>
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dce: <http://purl.org/dc/elements/1.1/>
PREFIX dcmitype: <http://purl.org/dc/dcmitype/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX schema: <http://schema.org/>
PREFIX sd: <http://www.w3.org/ns/sparql-service-description#>
PREFIX void: <http://rdfs.org/ns/void#>
```

Listing 3.1: Extended query associated with "List of prefixes by alphabetic order"

To be as complete as possible, each query takes into account the heterogeneity of the selected vocabularies and uses all coherent properties and classes of these vocabularies, covering all possible ways to express the questions using them. For instance, a query asking for a publisher accepts all publisher-like properties (using the Dublin Core, schema and PROV) as illustrated by Listing 3.2. It shows the translation of the question "Who publishes this dataset?" into a query, where ?kg must be replaced by the IRI of the knowledge graph at hand. All our queries are available on a GitHub repository[13], a summary of the preferred properties for each question is also available[14], and all the considered properties, listed by question, are given in Appendix A.3.

Some questions are translated into a succession of queries, with one main query and one or more others that are executed only if the first one succeeds. For instance, the question about the creators of the KG is translated by one main query that looks for creators, and four other queries that search for the identity, type, qualifications, and influences of each of the identified creator.

```
ASK {
  {?kg dct:publisher ?publisher .}
  UNION {?kg dce:publisher ?publisher .}
```

[10] https://www.w3.org/TR/vocab-dqv/

[11] Namely: Creative Commons, DOAP vocabulary: http://usefulinc.com/ns/doap, NEPOMUK Information Element Ontology (NIE): http://www.semanticdesktop.org/ontologies/2007/01/19/nie, Standards Ontology (STO): https://w3id.org/i40/sto, XHTML Vocabulary: http://www.w3.org/1999/xhtml/vocab

[12] http://creativecommons.org/ns

[13] https://github.com/Jendersen/KG_accountability/tree/v1.0/rules

[14] https://github.com/Jendersen/KG_accountability/blob/main/docs/questions_and_properties.md

```
  UNION {?kg schema:publisher ?publisher .}
  UNION {?kg schema:sdPublisher ?publisher .}
  UNION {?kg prov:wasGeneratedBy ?act .
    ?act a prov:Publish .
    ?act prov:wasAssociatedWith ?publisher .}
}
```

Listing 3.2: Extended query associated with "Who publishes this dataset?"

Finally, notice that to takes into account the heterogeneity of the vocabularies, we can either use the queries in their extended version, including all possible ways of expressing the required information, as in Listing 3.2. Alternatively, we can express the requirements in the form of a compact query, as in Listing 3.3, completed with a set of equivalences between properties (and between more complex graph patterns if necessary). The equivalences are listed in Appendix A.3. These two alternatives lead to the definition of different querying strategies, which are discussed later in the section on assessment.

```
ASK { ?kg dct:publisher ?publisher . }
```

Listing 3.3: Compact query associated with "Who publishes this dataset?"

3.3.3 Definition of an accountability measure

First, we define the score obtained for each question. Then it is possible to determine the score of each node of the KGAcc hierarchy defined in Figure 3.3 (see page 40), from the bottom to the top. The score at the top of the hierarchy is the overall accountability score.

First, let us introduce the weights used to calculate this score. They express the relative importance of one question compared to another. Since LiQuID does not weight its questions, nor does it mention any relative importance of some elements over others, we assume that they are of equal importance. To stay close to this, we use the following rule. When m (with $m \geq 1$) KGAcc questions come from a same LiQuID question, each of them is assigned a weight of $1/m$. Table 3.1 (page 42) illustrates these weights. For instance, "data usage - who" has three questions, coming from two LiQuID questions. The first leads to one question, so its weight is 1, and the second leads to two questions, therefore their weight is $1/2$ each.

So, to calculate the score, we start at the bottom of the hierarchy: the questions. A successful query gives a score of 1 to its associated question, while a failure gives a score of 0. The scores of the three questions associated with a succession of queries are the average of the scores given by each query. Then, the accountability score of a leaf of the KGAcc hierarchy (e.g. "data usage - who") is the weighted average of the scores obtained for its associated questions,

with the weights introduced earlier. In the previous example, the accountability score of "data usage - who" is the weighted average of the scores obtained for its three questions, with the weights of 1, $1/2$, and $1/2$ respectively. For the other elements of the hierarchy, we determine their score by computing the (non-weighted) average of the scores of the elements underneath.

Formally, let g be a knowledge graph, ℓ a leaf node of the KGAcc hierarchy (e.g. "data usage - who"), and let $Q(\ell)$ denote all questions associated with ℓ. With $score$ a function giving the score of g for a given question q, w_q the weight of question q, the accountability score of g w.r.t. ℓ is:

$$accountability(g, \ell) = \frac{\sum_{q \in Q(\ell)} w_q \cdot score(g, q)}{\sum_{q \in Q(\ell)} w_q}$$

and the score of a given node n of the KGAcc hierarchy which is not a leaf is:

$$accountability(g, n) = \frac{\sum_{n' \, child \, of \, n} accountability(g, n')}{number \;\; of \;\; children \;\; of \;\; n}$$

The global accountability score is the score for the upper node in the hierarchy.

3.4 Assessing methods and tools

We aim to evaluate knowledge graphs that are accessible through public SPARQL endpoints. To do so, we first select a tool to query them. Next, an important prerequisite for our queries is to identify their IRI within their own data, notably to determine exactly what is being measured. Then we present the different possible strategies to evaluate how the KG answers the queries defined in the previous section.

3.4.1 Assessment tool

In order to conduct our experiments and to query KGs with our own set of queries, several frameworks can be used, they are described in Section 2.3.4. Luzzu [69] and Sieve [91] enable users to choose metrics among those defined and to declare new ones. Monitoring tools, such as SPARQLES [35], also allow the assessment of some quality aspects. Instead of these, we choose the IndeGx framework [17]. It enables to profile KGs accessible through SPARQL endpoints using rules based on SPARQL queries and declared in RDF. Its primary use case is to use this SPARQL-based test suite to build a queryable index of SPARQL endpoints of the Linked Open Data by extracting and computing their description. Therefore, the IndeGx

framework enables querying many KGs, with multiple queries, and storing the results in RDF. We also choose IndeGx because it relies entirely on SPARQL queries, unlike Sieve and Luzzu, and it is easily extensible. Hence, IndeGx cover all our needs.

To evaluate KG accountability, we use it as an engine to submit our own queries to KGs. To do so, we provide SPARQL queries and configure the actions to be taken based on their results, i.e. which triples to write or update in the resulting RDF graph. So, we embed a set of queries into the framework, and declare how to store the result (True or False) for each evaluation query and for each KG using the DQV Vocabulary.

3.4.2 Prerequisite: Identify the IRI of the KG

An important aspect of our approach is the strict interpretation of questions and their faithful translation into queries. This leads us to be demanding as for the way KGs express metadata: we look for metadata explicitly linked to the IRI of the KG to ensure that the information is actually about it. Other methods are satisfied with just looking for metadata, even if they are not linked to an entity of type dataset (e.g. [69] when looking for a license of the dataset) or to the endpoint under study (e.g. [17] when looking for provenance). In the first case, the metadata could be about any entity, such as an image. The second case is a problem when querying a SPARQL endpoint, if its KG contains information about other KGs (or datasets), then the metadata could concern any of the datasets without being sure that it concerns the KG under study.

Therefore, a prerequisite of all our queries is to identify the IRI that the studied KG uses to refer to itself. Indeed, this IRI is the subject of at least one triple in all our queries, as illustrated in Listing 3.2, page 43. In our context, we evaluate remote SPARQL endpoints for which we do not have precise knowledge. Therefore, we use a query to identify their IRIs. And since we query KGs through SPARQL endpoints, the query used is the one defined and presented in Listing 3.4: it looks for an entity of class Dataset (from different vocabularies) which is linked to the URL (noted `$rawEndpointUrl`) of the endpoint interrogated. If the KG does not provide an answer to this query, it will not answer any of our queries. This query can be extended in Listing 3.5 to consider more ways of expressing a dataset and more ways of linking it to the endpoint URL. Notice that there may be multiple IRIs identified for the same KG: for example, some KGs define one dataset per named graph and thus may contain several datasets. In some other contexts, when evaluating dumps for instance, other strategies may be used, either replacing the variable representing the KG with its actual IRI if known or softening these queries.

```
1 SELECT ?kg WHERE {
2   # The KG must be a Dataset...
```

```
  { ?kg a dcat:Dataset }
  UNION { ?kg a void:Dataset }
  # and it must be linked to the endpoint URL
  ?kg ?endpointLink $rawEndpointUrl .
}
```

Listing 3.4: Simple query to identify the IRI of the studied KG

```
SELECT ?kg WHERE {
  # The KG must be a Dataset...
  { ?kg a dcat:Dataset }
  UNION { ?kg a void:Dataset }
  UNION { ?kg a dcmitype:Dataset }
  UNION { ?kg a schema:Dataset }
  UNION { ?kg a sd:Dataset }
  UNION { ?kg a dataid:Dataset }
  # and it must be linked to the endpoint URL, directly or not
  { ?kg ?endpointLink $rawEndpointUrl .}
  UNION {?kg dcat:accessService ?service .
         ?service dcat:endpointURL $rawEndpointUrl .}
  UNION {?kg dcat:accessService ?service .
         ?service sd:endpoint $rawEndpointUrl .}
  UNION {?service dcat:servesDataset ?kg .
         ?service dcat:endpointURL $rawEndpointUrl .}
  UNION {?service dcat:servesDataset ?kg .
         ?service sd:endpoint $rawEndpointUrl .}
}
```

Listing 3.5: Complete query to identify the IRI of the studied KG

3.4.3 Assessment strategies

Several strategies exist for querying and evaluating KGs. First, we detail two strategies for interrogating SPARQL endpoints, which differ in the queries sent to the remote endpoints and in what is performed locally. The first one works completely remotely and considers the queries in their extended form, as the one presented in Listing 3.2, while the second one reduces the load on remote SPARQL endpoints and uses their compact form (cf. Listing 3.3). Then, it is necessary to define exactly what is being measured, and again there are two possibilities. Either the KG is evaluated as a whole, or we consider the dataset it contains separately.

Querying strategies

The first strategy, which we call the select/evaluate strategy, consists in selecting the endpoints providing an IRI of a dataset linked to the SPARQL endpoint URL. This is done using a preliminary query: Listing 3.4 or 3.5. If an endpoint does not provide an answer to this query, it will not answer any of the accountability queries. All endpoints satisfying this query are selected for the next step, the others are assigned an accountability score of 0 as they cannot succeed any of our queries. The next step considers only the selected endpoints and evaluates them with all accountability queries in their extended version, such as Listing 3.2, page 43. The results of each query for each SPARQL endpoint are stored in a resulting RDF graph. In this strategy, both steps are performed on the remote SPARQL endpoints.

The second strategy, which we call the extract/augment/evaluate strategy, takes a different approach. To reduce the complexity and number of queries sent to remote SPARQL endpoints, it extracts the data of interest and then works locally. Therefore, it proceeds in the three following steps, with only the first one interacting with remote SPARQL endpoints. First, some queries extract the description of any dataset identified by Listing 3.4 or 3.5. Indeed, they extract all triples that have the IRI of the dataset as a subject, and some more triples concerning the contributors of this dataset for instance. These extracted descriptions are stored in a local RDF graph. If no dataset could be identified for an endpoint, then no description could be extracted and the endpoint is declared as having no accountability. Then, we saturate them locally by adding equivalent properties, as identified when designing the accountability queries: some queries complete the extracted triples with new triples having all equivalent properties. Finally, the accountability queries in their compact form (see Listing 3.3, page 44) are executed on the local augmented RDF graph, to measure the accountability of each dataset. The results of the evaluations are stored in the same RDF graph. This strategy has several other advantages: in the case of an index, it enriches the metadata stored about a KG and makes the information easier to find through its expression with multiple vocabularies. In addition, it facilitates the addition of new metrics, since they can rely on the existing equivalences and thus reduce the difficulty of writing the corresponding queries.

Evaluation target

In addition to these two assessment strategies, it is important to identify what is being evaluated: either the KG at a whole, or the KG as multiple datasets. In the previous subsection, we detailed how to identify the IRI of the dataset(s) it contains. However, there may be several datasets identified by this method, for example if each named graph defines a dataset. In addition, there is no certain and automatic way to identify the main

dataset that represents the KG as a whole. Indeed, there is no class or property in any existing vocabulary to characterize it, some of the KG providers name the dataset with the string "void" in it (e.g., http://ldf.fi/ww1lod/void/Dataset), or even better with ".well-known/void" (e.g., http://caligraph.org/.well-known/void), referring to the best practice of the "well-known uris"[15] mechanism, but this is neither systematic nor a guarantee. So we identified two possibilities for defining what to evaluate, given a knowledge graph in which multiple datasets are identified.

The first one considers all datasets at the same time, as if they were a single dataset, so that finding information for any one of them is satisfying for the whole KG. For instance, if there is a creation date in a dataset A of the KG and an author in a dataset B of the KG, then we consider that both pieces of information are present concerning the KG itself. In the second one, we evaluate each dataset separately. This evaluation is more precise and accurate (indeed, the creation date of a dataset is not necessarily that of the KG). The accountability of the KG itself can be analyzed by studying the accountability of each of its datasets, with the option of taking the best, worst or average score, or by trying to find by hand the one that directly represents the KG. If only one dataset is identified in the KG, both strategies will produce the same result.

The queries do not change to make this distinction. What changes is who the information is associated with. In the first case, the result of each query is associated with the SPARQL endpoint URL, while in the second case it is separately associated with each IRI. The select/evaluate strategy, combined with the use of IndeGx, does not enable to distinguish between the different datasets of a KG and obliges to consider it as a whole. The extract/augment/evaluate strategy, made possible by an improvement of IndeGx, allows to consider both the KG as a whole and the KG as multiple datasets, since each extracted data can be associated with the endpoint URL or with the IRI itself.

3.5 Evaluation campaigns

We conducted two evaluation campaigns, using different assessment strategies. After describing these campaigns, we present the results obtained. hen, we discuss two aspects of the results: we examine the capabilities of knowledge graphs with regard to accountability, and, we discuss the measure itself and the relevance of the KGAcc questions.

[15] https://www.w3.org/TR/void/#well-known

3.5.1 First campaign

Description

The first campaign uses the queries of the KGAcc framework in their extended form, using the select/evaluate strategy. It uses the query of Listing 3.4 (see page 46) as a preliminary query to identify the IRI of the datasets contained in a KG. It evaluates the KG by considering its whole content as a single dataset, without distinguishing its datasets. All the queries and results of this campaign are publicly available on our GitHub repository[16].

We used IndeGx in its first version[17] for the campaign. It was conducted in September and October 2022. It queries 670 SPARQL endpoints already identified by IndeGx and extracted from the LOD Cloud, Wikidata, SPARQLES, Yummy Data and Linked Wiki. The preliminary query to select the SPARQL endpoint is executed on all endpoints at three different dates and times, separated by several days. The objective is to detect all candidates succeeding this query at least once and not to penalize them if they are not available during the evaluation period. Then, the selected endpoints are evaluated with all accountability queries, running them three times, still at different time points. For each endpoint, only the results of the last experiment for which it was available are kept. This should be the most up to date.

Finally, given the results obtained for each query and thus each question, the accountability score can be computed. As defined in subsection 3.3.3 (page 44), an average is used to calculate the score for each aspect of the KGAcc hierarchy, until the overall accountability score is obtained.

Results

Among the 670 SPARQL endpoints tested, only 29 successfully pass the preliminary query. Therefore, for the KGs associated with the endpoints that fail this query, we could not find any accountability metadata. The measure of accountability allows to discriminate between the 29 KGs left, with scores distributed between 2.2% and 44% of the maximum achievable score. The mean and median of these values are of 22%. On average, endpoints are twice more complete on "data usage" than on "data collection", and more than twice better on "data collection" than on "data maintenance".

In details, Figure 3.5 shows their overall accountability scores. They are divided into the three life cycle steps "data collection", "data maintenance" and "data usage". For instance, the accountability of http://taxref.mnhn.fr/sparql and http://id.nlm.nih.gov/mesh/sparql on these dimensions is shown on Figure 3.6a. It is also possible to compare them in more detail, considering lower aspects of the KGAcc hierarchy. Hence, Figure 3.6b compares

[16] https://github.com/Jendersen/KG_accountability/tree/v1.0
[17] https://github.com/Wimmics/dekalog (Release: v2.6.2)

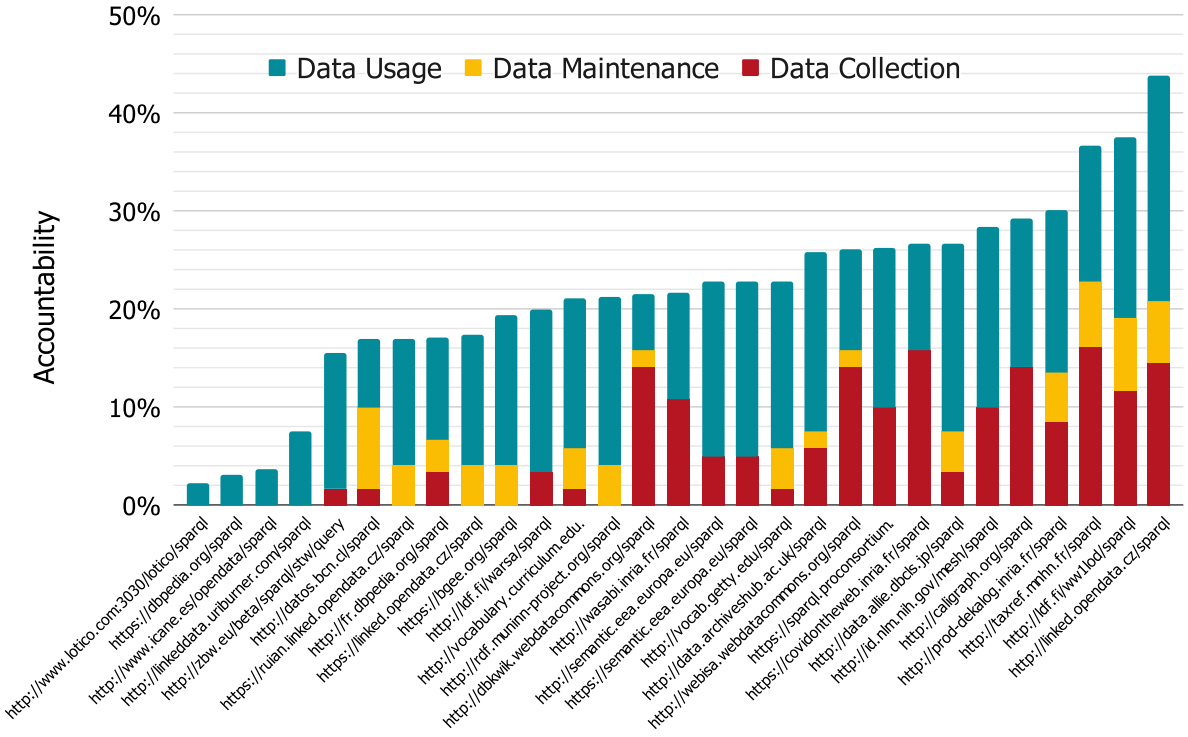
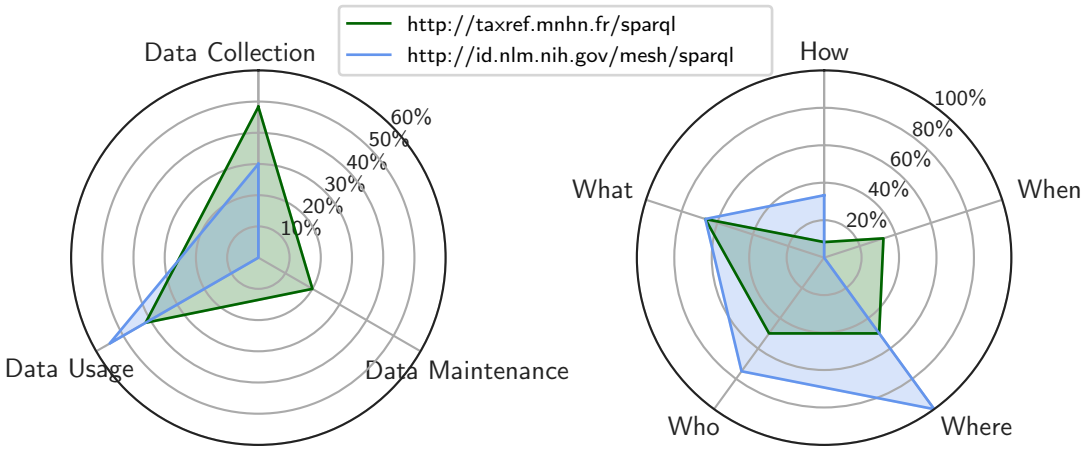


Figure 3.5: First campaign: Accountability score obtained by each KG, identified by their endpoint URL

the two KGs on the question types of "data usage". An even finer analysis is possible, since all scores are available, including those obtained for each question.



(a) Accountability of two KGs w.r.t. the Life Cycle Steps

(b) Accountability of two KGs w.r.t. the Question Types of Data Usage

Figure 3.6: Accountability of two KGs w.r.t. different aspects of the KGAcc hierarchy

3.5.2 Second campaign

Description

The second campaign followed a two-week stay with the Wimmics team at the INRIA Sophia-Antipolis. This enabled us to benefit from the expertise of other members of the DeKaloG project and led us to slightly improve some queries compared to the first campaign, mainly by relaxing overly constrained queries. The changes are as follows.

- Most of the changes concern the PROV-O alternatives of the queries. Indeed, the constraints on the type of `prov:Activity` (such as `prov:Create`, `prov:Contribute`, `prov:Modify`...) have been removed because these classes are not part of the core PROV-O vocabulary [99] and are not used in practice. (7 queries modified)
- Textual descriptions of the methods are no longer required since they are not relevant. (2 queries modified)
- Some alternatives have been added to queries: 3 queries have one additional alternative, 3 queries with 2 additional alternatives, and 3 queries with more alternatives for licences.
- Two queries were corrected: one property seems to be inappropriate in a query and was removed, and one property was misspelled (`prov:wasGeneratedAtTime` instead of `prov:generatedAtTime`).

Several queries appeared twice in the aforementioned changes, so in total 13 of the 30 queries were modified. However, these changes are small compared to what the queries were before, since the most important properties were already there. In addition, changes on PROV have a limited impact because the other alternatives are much more likely to match the KG than the PROV part due to the limited use of this vocabulary to describe the KG.

The second campaign uses these new queries of KGAcc in their compact form, using the extract/augment/evaluate strategy. It evaluates separately the different datasets the KG contains. It uses the query of Listing 3.5 (see page 47) to identify their IRI, which is the extended version of the identification of a dataset. All the queries and results of this second campaign are publicly available on our GitHub repository[18].

We used IndeGx in its second version[19] for this campaign, which was conducted in June 2023. It queries 336 of the previous SPARQL endpoints, which have been cleaned of all endpoints that did not answered any of IndeGx's experiments over a period of eight months [17]. These endpoints were queried at three different time points. For each endpoint, only the results of an experiment for which it was available are kept. All endpoints that failed the query in

[18] https://github.com/Jendersen/KG_accountability/tree/v2.0
[19] https://github.com/Wimmics/IndeGx (Release: v2.0)

Listing 3.5 are assigned an accountability score of 0, as no triple concerning its KG could be extracted. Finally, as for the first campaign, the accountability score can be computed as defined by the KGAcc framework in subsection 3.3.3, page 44.

Results

Among the 336 endpoints tested, only 26 successfully provide accountability information. The others were unavailable or did not provide easily accessible metadata about themselves within their data. Among the 26 endpoints, 166 different datasets were identified (in the sense of `dcat:Dataset` or `void:Dataset`...), with accountability scores varying between 3.1% and 59%, with an average score of 26%. On average, datasets are still more accountable concerning "data usage" (41%) than "data collection" (25%) and than "data maintenance" (12%). Figure 3.7 shows the accountability score of the best dataset of each SPARQL endpoint, divided according to the three life cycle steps like in Figure 3.5. However, there is no guarantee that this best dataset is the one of most interest, i.e. the one that describes the KG. For instance, Table 3.2 details the results obtained by each dataset of one KG on each KGAcc question, the best being the last two on the Table, but they do not represent the KG itself. As for the previous campaign, it is still possible to compare datasets in more detail to highlights their strengths and weaknesses.

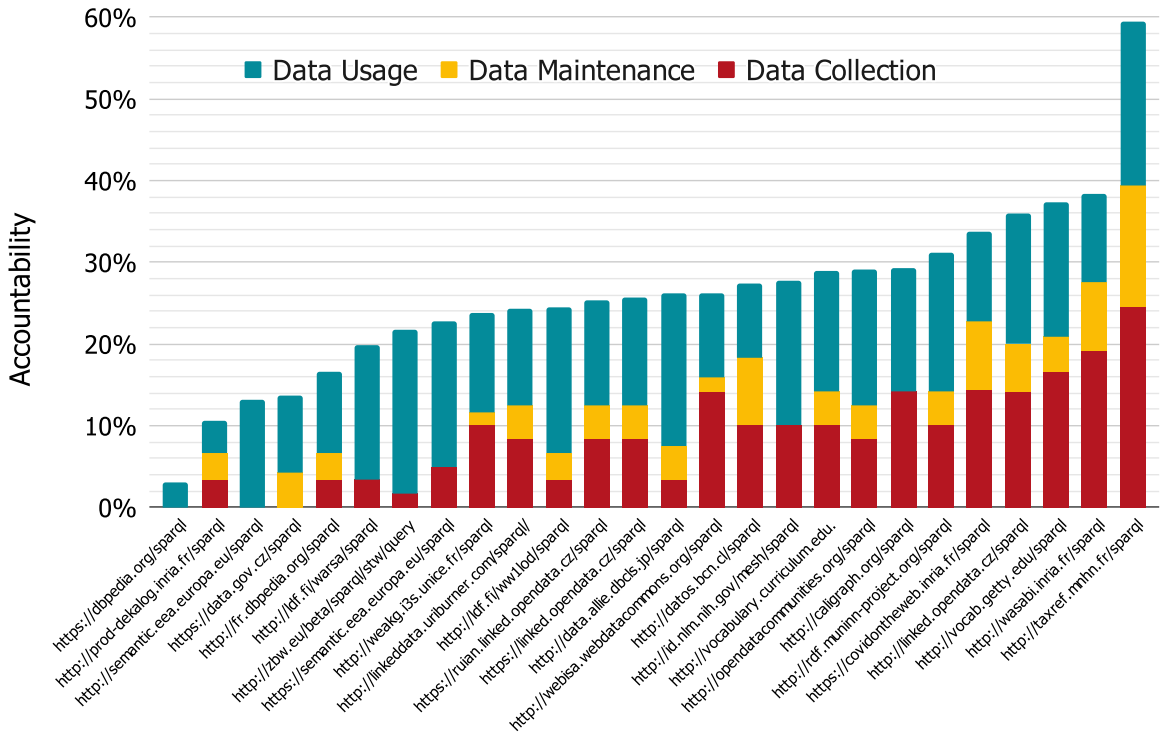


Figure 3.7: Second campaign: Accountability score obtained by the best dataset of each endpoint

Table 3.2: Details of the score obtain on each KGAcc question by the four dataset of SPARQL endpoint: http://linkeddata.uriburner.com/sparql/

	Dataset:	.../flickr-wrappr	.../viaf/data	.../L687573-F1	.../L687765-F1
Collection	contributor	0.2	0	0	0
	date	0	0	1	1
	source	0	0	0	0
	location	0	0	0	0
	methodology	0	0	0	0
Maintenance	contributor	0	0	0	0
	date	0	0	1	1
	frequency	0	0	0	0
	location	0	0	0	0
	methodology	0	0	0	0
Usage	dataset publisher	0	1	0	0
	rights	0	1	1	1
	audience	0	0	0	0
	start availability	0	0	0	0
	end availability	0	0	0	0
	end validity	0	0	0	0
	dataset webpage	1	1	1	1
	access address	1	1	1	1
	location	0	1	1	1
	license	0	1	1	1
	access	0.5	0.5	0.5	0.5
	reuse	0	0	0	0
	requirements	0	0	0	0
	examples	0	0	0	0
	concepts covered	1	1	1	1
	dataset description	1	1	0	0
	dataset triples	0	0	0	0
	dataset entities	0	0	0	0
	RDF serialization	0	0	1	1
	dataset quality	0	0	0	0

3.5.3 Comparisons

The two evaluation campaigns did not identify the same SPARQL endpoints with accountability information. Indeed, 23 SPARQL endpoints were identified by both campaigns. Six endpoints appear in the first campaign but not in the second one: five of them were not available at the time of the evaluation and the data of the last one have changed between the two campaigns so that no dataset can be found anymore. Three endpoints appear in the second campaign but not in the first one: one because it was added to the catalog, one thanks to the new ways of identifying the IRI of the dataset and thus a better handling of heterogeneity (see Listing 3.5, compared to Listing 3.4), and the last one thanks to an improvement in

the IndeGx engine between its two versions.

The second campaign is more precise as it enables to consider separately the different datasets available on the same endpoint. There is only one dataset for most of the endpoints: 19 out of 26. In this case, there is no difference with the first strategy because the only identified dataset represents the whole KG. But it might be important for the rest of them. For instance, http://linked.opendata.cz/sparql contains 96 datasets linked to it. For the first campaign, it obtained a score of 44%, which combines any information about one of its datasets, no matter which one. The second campaign evaluated the accountability of each dataset separately, with scores ranging from 17% to 36%. This explains why the score of the best dataset of http://linked.opendata.cz/sparql in Figure 3.7 is lower than the score of the endpoint in Figure 3.5.

When considering SPARQL endpoints that appear in both campaigns, we see that some of their accountability scores have changed. There are two explanations for this. First, because the queries were slightly improved between the first and second campaigns, with some over-constrained queries being relaxed as described earlier. Second, some knowledge graphs have changed and been improved. This is the case of TaxRef[20] (which contains only one dataset), whose accountability score increased from 37% to 59%. This significant improvement is due to specific work on improving metadata, especially with respect to the FAIR principles.

3.5.4 Conclusions on the results and discussion on the framework

The small number of KGs for which accountability metadata could be found is disappointing, but not surprising. This observation is in line with other results obtained by IndeGx [17]: only 33 of the 339 KGs tested contain some provenance metadata linked to an instance of `void:Dataset` or `dcat:Dataset`, i.e. less than 10% of them provide a self-description within their data. Considering that five endpoints were unavailable at the time of the second experiment but did provide findable accountability information in the first experiment, we can assume that 31 of the same 339 KGs provide accountability metadata. Therefore, requiring the IRI of the KG to be linked to the queried SPARQL endpoint is not so constraining, while this provides more guarantees of evaluating the proper IRI.

However, for some KGs, some metadata may be stored outside the KG itself, or inside but not findable in the way shown in Listing 3.5. Concerning the former case, several approaches store metadata in external files, or in the web page of the KG. The best practice proposed by the W3C consists of providing all metadata in a `/.well-known/void` file. However, according to Maillot et al. [17], only 10 of 339 KGs provide metadata in this way. Looking for metadata

[20] http://taxref.mnhn.fr/sparql

stored elsewhere would be interesting, but again the question is how to identify and locate them. There is no unified way to store them or even to provide the address where they can be accessed, even though some communities have adopted good practices. The latter case may happen if *(i)* metadata is too difficult to find, then, it points out the fact that they are less accountable because the information is less accessible ; or if *(ii)* the KG is not available through a SPARQL endpoint. This case is beyond the scope of our study. However, our queries and the list of properties we provide for each question can also be used to evaluate RDF dumps or any metadata expressed in RDF. For instance, our queries can be used by replacing the variable `?kg` with the actual IRI of the KG, if known, or by constraining it differently, such as by removing the link to the endpoint URL.

Given the fact that most of the KGs have no findable accountability metadata, we consider that an accountability above 26% (the mean of the last campaign) is a good score of accountability. Even though the accountability scores are quite low, all KGs have a margin to improve considering the whole set of queries that are answered by at least one KG. Indeed, a KG that passed all these queries would have an accountability score of 79%. In particular, there is a large margin for improvement in collection and maintenance.

Several reasons may explain the scores obtained on the different life cycle steps. The "data usage" step covers general description elements that are widely used such as a description, a publisher, or a license, that more than half of the 26 endpoints provide. Furthermore, it encompasses all questions involving VoID vocabulary, which are each answered at least once by more than 50% of the endpoints on average. "Data usage" also requires a link to the endpoint or a dump, so having an answer to this question is expected considering how we identify the IRI of the KG. "Data maintenance" usually has bad scores. This may be due to the fact that for half of the questions, there is only one possible property among the chosen vocabularies, with no alternative. For instance, the modification frequency can only be obtained with the property `accrualPeriodicity` from the Dublin Core vocabulary. The lack of alternative solutions to express this concept makes it more difficult to answer the query and highlights the fact that the question is less common. "Data creation" has various results with very common requirements, such as the creator and the creation date, and more difficult questions to answer such as the creation method.

As far as the framework is concerned, at least two questions can be raised: are the requirements relevant? Are they too demanding? Figure 3.8 provides some answers from the perspective of data providers. It highlights the extent to which KGs provide answers by aspect of the KGAcc hierarchy, which can suggest what types of information are relevant to ask. The figure represents the distribution of the scores of accountability of the best dataset of each endpoint w.r.t. each aspect of the KGAcc hierarchy (second campaign). Each box represents

the first quartile (Q1), the median, and the third quartile (Q3) of the scores obtained on the different aspects and the whiskers indicate the minimum and the maximum scores obtained. It shows that the question types of "data usage" usually have good scores in the different KGs. It also shows that half of the aspects can be fully covered, including "data collection - who", "data collection - when" and "data collection - how" for instance.

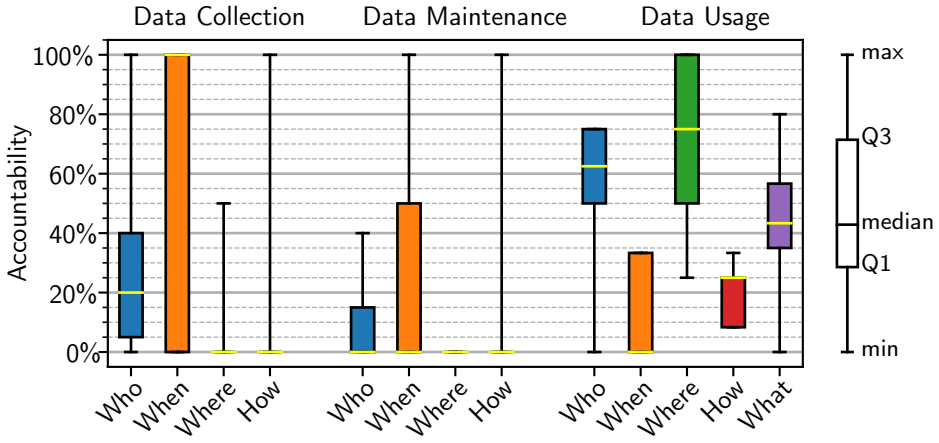


Figure 3.8: Box plot showing the distribution of the scores of accountability w.r.t. each aspect of the hierarchy.

On the one hand, as many of the aspects sometimes get the maximum score, it shows that these queries are relevant and that KGs have a good margin to improve. On the other hand, some of the low scores observed on that figure may be explained because some queries are too demanding. Indeed, it is important to note that, in this second campaign, 6 out of 30 queries never succeed. A part of them is due to some strict choices we made that resulted in over-constrained queries. For instance, in "data usage - when" the end date of availability of the dataset may be difficult for providers to specify, as they may hope that their KGs will be available indefinitely. Other questions concerning locations may not be in line with the current practices. Indeed, they are especially important for KGs that hold private information, which is generally not the case for public SPARQL endpoints. As other frameworks, depending on the context, KGAcc may be discussed and improved with experts and KG providers.

3.6 Comparison with other assessment measures for knowledge graphs

To take the analysis of our framework a step further, we compare it in detail with several data quality and FAIRness measures. Our goal is to make a theoretical comparison between what is required by these measures and what is required by ours. We are not interested

in how the associated evaluation frameworks work, nor in how they search the information within the RDF resources, this has been discussed in Section 2.3.2. Comparing measures by looking at requirements expressed in natural language is difficult and often inaccurate, due to the possible ambiguity of requirements and the differences between specification and implementation. Indeed, depending on the context, the latter may not fully match the specification, or may not be as complete as one would expect. For example, FAIR requires provenance, but without the implementation it is impossible to know what is being looked for (authors, methodology, affiliations, etc.).

Therefore, the comparison is made at the level of the required properties: we aim to verify to what extent other studies require the properties demanded by the KGAcc framework. To do so, we focus on studies that consider RDF properties and RDF datasets, and that provide an easy access to the exhaustive list of RDF properties or patterns used in their metrics to allow comparison. This is why works such as F-UJI [84] or Sieve [91] are not considered.

Concerning data quality, Zaveri et al. [13] provide an organized list of metrics obtained by a systematic literature review. This work is theoretical and does not implement these metrics, therefore, we do not take it into account. However, we focus on two major studies of data quality inspired by Zaveri et al. First, Färber et al. [27] evaluate the data quality of five cross-domain KGs, namely DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. While the implementation of their metrics is not available, each metric is richly described. Secondly, Debattista et al. [69] provide a more generic set of data quality metrics enabling the evaluation of any KG. Their implementation is available online but the article [69] describing them is more detailed and understandable. Therefore, for all these studies, we base our comparison solely on the referenced article.

For FAIRness, FAIR-checker [79] is interested in RDF triples as embedded metadata in web pages. For comparison, we rely on the specifications provided by the online tool[21] when evaluating a resource. We also consider O'FAIRe [93] which focuses on RDF ontologies. It provides an online tool[22] to see the results obtained by a list of ontologies. To compare with them, we consider the complete list of questions and their required properties[23]. In total, this leads us to consider two data quality studies and two FAIRness ones.

Table 3.3 summarizes our comparative study. For each KGAcc query, if one of its required properties is also required in a data quality or FAIR metric, a mark is indicated in the table. If this property must not necessarily concern the KG (e.g. the creator of some resource instead of the creator of the dataset), then the mark is $\approx$, showing that the FAIR or data quality metric is

[21] https://fair-checker.france-bioinformatique.fr/check Accessed: 10 April 2023

[22] https://agroportal.lirmm.fr/landscape#fairness_assessment

[23] https://github.com/agroportal/fairness/blob/master/doc/results/FAIR-questions.md Accessed: 10 April 2023

Table 3.3: Comparison between the KGAcc queries and the metrics proposed by the works on data quality and the FAIR principles

Lifecycle step	Question type	Accountability query	Data Quality		FAIR	
			D [69]	F [27]	C [79]	O [93]
Data Collection	Who	Creator	✓		⊂	✓
		- Creator's info.				
	When	Creation date			⊂	⊂
	Where	Source		≈	⊂	✓
		Creation location			⊂	
	How	Creation method			⊂	✓
Data Maintenance	Who	Contributor			⊂	✓
		- Contributor's info.				
	When	Modification date		≈	⊂	⊂
		Frequency				✓
	Where	Modification location				
	How	Modification method			⊂	
Data Usage	Who	Publishers	✓			⊂
		Usage rights	✓	✓	✓	✓
		Audience				⊂
	When	Start of availability				
		End of availability				
		End of validity				⊂
	Where	Webpage				⊂
		Access URL		✓		⊂
		Usage location	✓	✓	✓	✓
	How	License	✓	✓	✓	✓
		Access URL		✓		⊂
		- Endpoint's info.				
		Usage information				
		Usage requirements				
	What	Examples				
		Concepts				⊂
		Description				⊂
		Triples				
		Entities prop. classes				
		Serialization	✓			
		Quality				⊂

✓ Property required and must concern the dataset (or the ontology).
≈ Property required but not linked to any particular resource.
⊂ One of the required properties is listed in the metric among other semantically different properties.

not really related to dataset accountability. Otherwise, if this property is mandatory to obtain a maximum score on the data quality or FAIR metric, the mark is ✓. If the property is listed among other properties and only one or two or n of these properties are necessary for success, then the mark is $\subset$. For instance, in O'FAIRe the third question for principle F2 states that to obtain the maximum score, six properties should be used from a list of 37 properties, whatever those six properties may be. Therefore, unlike the ✓ mark, a $\subset$ mark does not guarantee that passing the FAIR metric ensures passing the accountability query.

This table highlights several elements concerning data quality metrics. First, as part of the measure of accessibility, all these evaluations require a license to be present using properties such as `dct:licence` [27, 69]. They also demand some particular provenance information: the creators or the publishers of the KG [69], or other information not specifically related to the KG such as the source of some data to enhance trustworthiness [27], their modification dates [27], or traceability of the data [69]. Some other metadata is expected to be provided, such as the serialization formats [69]. Finally, Färber et al. [27] request the provision of KG metadata citing as an example the URI of the SPARQL endpoint or the RDF export URL to indicate where to access the data.

Concerning FAIR metrics, only two metrics are related to accountability in FAIR-checker. The first one measures the 'R1.1' principle that requires a license. The second one measures the provisioning of provenance information (R1.2) by checking that at least one of the 23 listed properties is found (such as `prov:wasDerivedFrom`, `pav:createdBy`, etc.). O'FAIRe offers more similarities with our work. There are mainly two different kinds of metrics of interest compared to our queries. First, the metrics concerning the reusability principles focus on one information each and are mostly also required by accountability (creator, contributor, source, method, periodicity, license and rights). Secondly, some metrics of findability require that the ontology uses some well-known properties. Indeed, two questions of the principle 'F2' cover properties required by at least 11 accountability queries (such as `dct:created`, `void:dataDump`...). O'FAIRe also considers other vocabularies compared to ours, such as the MOD ontology [100], which focuses on ontology metadata. However, including it in KGAcc would not have changed the results of the comparison, since only one property is related to accountability in a question that is already marked as $\subset$ thanks to other properties. This ontology is also interesting as it establishes some equivalences between existing terms.

As a result of this comparison, both data quality and FAIRness share common interests with accountability. Therefore, improving them may have a positive impact on the assessment of accountability and vice versa. However, neither data quality nor FAIRness focuses specifically on accountability as a whole and does not take into account all the elements it requires. The general studies on data quality only slightly overlap with accountability. FAIRness has more

similarities, particularly with regard to the steps of data collection and maintenance as it is mainly interested in questions of provenance. In particular, O'FAIRe seems to have many similarities. However, most of them result solely from the two findability metrics, which are not very informative about the type of metadata that are present, since they cover no more than 11 of our queries. Therefore, the measure of accountability is much more detailed, precise, and focused than O'FAIRe on our point of interest. And as the result of each query is available, the former provides a much more relevant view of accountability.

On the other hand, we can also observe that quality and FAIRness measures are more global and study much more diverse aspects than accountability. For example, the metrics shared with accountability cover only 20% of the FAIR principles implemented by FAIR-checker and one third of the principles for O'FAIRe.

3.7 Conclusion

In order to evaluate the accountability of RDF graphs, we proposed : *(i)* the KGAcc framework which defines achievable requirements regarding the metadata that the KGs should expose and an associated measure, *(ii)* the evaluation of a large set of endpoints, *(iii)* a comparison of our approach with other frameworks that assess data quality or compliance with FAIR principles.

The KGAcc requirements are expressed as SPARQL queries. They are obtained through a meticulous adaptation of an existing hierarchy of natural language questions, proposed for datasets in general [21], to the specific context of KGs. The dedicated vocabularies make easy stating some requirements, but their lack of expressiveness makes some questions collapse into the same query or prevents from considering demanding and precise questions about ethical and legal questions. This is why we end up with a relatively small set of queries.

The evaluation of many RDF graphs reveals that most of them do not provide any of the required information within their data, even though some of the required information is very commonly requested. In general, there is a lack of consensus on how and where to express the metadata of a KG, which may explain the small number of RDF graphs for which we found metadata. From our point of view, efforts should be made to push for the adoption of good practices, such as providing metadata through a `/.well-known/void` file, or to propose a standard way of identifying metadata within the graphs, such as a new property or class meaning "I am the metadata describing the current RDF graph". However, there are RDF graphs that provide some of the expected information, showing that our demands are reasonable from the point of view of the KG providers. Therefore, there is room for improvement for many RDF graphs. Beyond the obtained scores, to our knowledge, this is the first time that such an evaluation is performed for this type of datasets. And our measurement is detailed

enough to help any KG producer to precisely identify missing information and how to express it to provide answers to queries, and thus to improve on these aspects.

Our comparative study shows that data quality and FAIRness assessment frameworks share properties with our measure of accountability, but they do not cover all its aspects. Similarly, FAIR principles and data quality explore broader aspects of datasets that are beyond the scope of our measure. Therefore, a good score of FAIRness or of quality does not guarantee a good accountability score and vice versa. In particular, O'FAIRe is the framework considering the most properties common to accountability. However, it is not so demanding in terms of accountability and cannot be considered as a framework dedicated to this aspect.

Finally, our measure of accountability has some limitations and thus can certainly be improved. First, our queries do not take into account the heterogeneity of all existing semantic web vocabularies. Therefore, some KGs may provide the required information, but not in a way that we can find it. To face this problem, we have two options. Either we enrich the queries to include all possible representations of the required information. It might be interesting to automate this task of defining equivalences between properties, for example by using ontology alignment. But when do we stop looking for additional vocabularies? Is it still accountable if no one but the provider can find the information? This would also be a never-ending quest. Or, we agree on a consensual representation of the accountability information. In this case, KGs have to represent the required information according to this new reference. Whatever the viewpoint, it is clear that recommendations, guidelines or standards for accountability would be useful. In this respect, our framework is a very first contribution, with all the required properties clearly listed, which could be a starting point for a working group.

Second, as we have said, many LiQuID questions are not covered. This is due to the lack of current vocabularies to express certain information. Thus, an interesting future work would be to study the creation of a new vocabulary dedicated to the expression of these elements. The easiest way would be to implement the LiQuID metadata model in RDF, but we could probably be more precise to allow not only the expression but also the querying of such information.

Finally, the formulation of some of our queries may not be in line with the current practices of the KGs. For instance, some of the queries using PROV-O may be over-constrained. In addition, some of the requirements may be more important than others in the global accountability score. Therefore, in another future work, we could improve our measure in several ways: by relaxing some queries, or by introducing gradations in the definition of accountability requirements. We could also introduce some weights to aggregate the results differently. To do this, we can take inspiration from the study that determined the weights for each O'FAIRe question [101]. In addition, we could study current practices, such as in [102], by analyzing which properties or patterns are most frequently answered in each query. In this way, we could determine a smaller

set of recommended properties to add to a knowledge graph and encourage KG providers to add them through initiatives such as Metadatamatic [103].

The KGAcc framework is based on a hierarchical structure, as are data quality studies and FAIRness measures. In addition, KGAcc is made of distinct elements that are used by the measure: this hierarchical organization, natural language requirements, here expressed as questions, and finally SPARQL queries that implement the requirements. Finally, the hierarchy and weights are used to calculate the overall score by successive weighted averages. Most of these different components can also be identified in the other measures we have presented so far. Therefore, we propose a formalization to describe such measures in a shared way, facilitating their comparison and their reuse.

4

Measures Based on a Tree-Structure of Requirements

Contents

MANY measures of knowledge graphs exist, but they are expressed in very different ways, making them more difficult to understand, select, compare, and reuse. In this chapter, we propose a formalization to represent these measures in a common framework. We also define some operators to manipulate them. Finally, we propose some tools to compare measures, helping users to better highlight their differences, to choose one over another, or to combine them.

4.1 Introduction

Data quality measures [27, 69], FAIRness evaluations [15, 93], the Umaka score [74], and the emergence of many other assessment tools illustrate the importance of measuring digital objects. All of these measures cover different uses and expectations, with some overlap between them. Rather than searching for a measure that would cover all different needs, we believe it is important to capture and have access to all of them. However, there is a wide variety of practices, with or without open and accessible implementation, with or without precise description of the measure in natural language. As a result, it is sometimes difficult to unambiguously and precisely understand what is being measured and how. It is even more difficult to compare some measures, even if they share the same foundation, such as the FAIR principles [14].

By studying the different measures presented in the previous chapters, on accountability, data quality, and the FAIR principles, we observed that most of them share a common structure. They are composed of several metrics, i.e. atomic items to be measured, organized by a tree structure with different categories (or principles, or question types...). Therefore, we aim to propose a framework for users to express and organize their measure according to such a structure.

The main objective is to enable a clear description and understanding of the measure: with a complete developed structure, the specification of what is measured in natural language, and the importance of all these aspects when computing the overall score of the measure.

By proposing a common formalism for expressing measures, we also intend to facilitate the comparison of measures, through the identification of what is shared or not, and the analysis of the different importance they give to these shared concepts. Finally, we aim to encourage the reuse of existing measures to build new ones, allowing users to consider only part of a measure or to combine two measures.

To achieve this, the formalism to be defined should enable the representation of measures with the tree-like structure of categories (or dimensions, principles...) on which many encountered measures are based. This formalism should also *(i)* allow the representation of hierarchies with no depth constraint, such as the three-level data quality measures, the four-level LiQuID hierarchy and the unbalanced FAIR principles ; *(ii)* allow a clear separation of the implementation and the specification of the measure, to provide a clear description of the measure in natural language and to enable different people to specify and to implement the measure ; *(iii)* enable the precision of weights to indicate the importance of each element.

As a result, a measure defined in such a formalism would be explicit both in terms of its structure and its requirements. It can be used to document and understand some quality indication of KGs and is useful for all users of KGs: for data producers, data consumers, producers of KG indexes, and people providing recommendations (FAIR...) on the semantic web. In addition, a wide range of needs can benefit from this formalism, allowing the definition of prototypical measures shared by a large number of people, or for a very specific measure of more individual interest.

Therefore, we propose to formalize a measure as a set of requirements expressed in natural language and organized by a weighted tree structure of analysis elements. Requirements are the precise and unambiguous specification of what to measure, regardless of any implementation. They are close to the concept of metrics but we prefer requirements to insist on the natural language specification. The tree-like nature echoes the progressive and systematic way of listing questions when analyzing a problem. General definitions are independent of any technology, so query languages are introduced only with the implementation of the measure. It is then possible to evaluate a knowledge graph against this measure. We provide the formal definitions corresponding to these ideas, with two central elements we call a *Tree-Structure of Requirements* (TSoR) and an *Implemented TSoR-based Measure* (ITM).

Then, we define operators to help users create measures based on existing ones, by focusing on some aspects of a given measure, or by combining one measure with another to enrich it with new analysis elements or requirements. Second, we provide keys to compare several measures in terms of their structure and, to some extent, the elements they evaluate. The goal is not only to make it easier to share measures, but also to compare them to help a user choose one over another or to combine them.

A primary version of the formalization has first been presented in a report for the ANR DeKaloG project [104].

In this chapter, the next section is dedicated to related work. Then we develop the formalization: the first definitions in Section 4.3, the operators to modify and manipulate measures Section 4.4, and the comparison in Section 4.5. These last three sections are illustrated with examples on data quality, accountability, and FAIR.

4.2 Related work

There are several measures for auditing and evaluating KGs, including data quality, FAIR principles, and several monitoring tools. Although these compound measures all rely on a hierarchical structure, they are not described with the same terms. We described these measures of knowledge graphs in detail in the Section 2.3 of the preliminaries. Here, we analyze the organization and representation of measures that follow a hierarchical structure.

First, data quality studies provide an extensive list of metrics covering many concerns. Wang and Strong [72] introduce a data quality assessment framework that organizes the metrics into a hierarchical structure of three levels. It was then adapted for knowledge graphs [13, 27, 69]. In this framework, data quality is divided into several "categories", such as contextual category, accessibility, etc. Each category is subdivided into "dimensions" containing one or more "criteria". Finally, each data quality criterion is associated with a metric in order to measure it on a given knowledge graph. Then, FAIR [14] is also organized according to the four concepts it covers: *Findability*, *Accessibility*, *Interoperability*, and *Reusability*, and then in principles and sub-principles.

These measures are not expressed in the same framework. An effort has been made to standardize the results of an experiment with the Data Quality Vocabulary [45], even though it is not widely adopted[1]. This vocabulary is based on data quality studies and uses the same hierarchy: metric, dimension, category. However, to the best of our knowledge, there is not yet a solution to precisely describe a measure in a common and standardized way. In addition, this three-level structure may be too restrictive. For example, the LiQuID hierarchy (see Chapter 3) has four levels (considering the questions as metrics).

Top-bottom approaches first define a main goal and then refine it according to domains, criteria, etc. In this way, they often rely on a tree structure. One of these is the Goal-Question-Metric approach [106]. It is a hierarchical model that starts with a goal that states what is being

[1]DQV is ranked 1188 out of 3029 of popularity in prefix.cc. It is used in zero dataset according to the Linked Open Vocabularies. Plus, it does not appear in any of the vocabularies identified by Shi et al. [105] in their results on vocabulary usage.

measured, for what reasons, etc. This goal is then characterized and refined into questions. And the questions are refined into metrics to answer the questions in a quantitative way. One question can lead to several metrics, and one metric can be used by several questions. Behkamal et al. [107] use this approach to define new metrics for evaluating datasets of the LOD cloud. In this way, "the process for metric definition starts by defining a set of goals, developing certain questions to characterize each of the goals, and is finalized by proposing metrics, which answer the posed questions". As with DQV, the number of levels is too restrictive, and it would be interesting to add more layers with goals, sub-goals... before branching to questions.

The Analytic Hierarchy Process [108] further develops the structure and refines the goal on several levels. In addition, it allows users to assign weights to its elements, expressing their relative importance. This process was notably used to assess metadata quality in open data portals [109].

Our approach allows to formally define a measure composed of several items. We model it as a list of requirements structured by a hierarchy of analysis elements. It is quite generic: the analysis elements may represent categories, dimensions, principles, criteria, goals, or domains, etc. Thus, any of the previous approaches may be expressed with our model. The requirements here are similar to the questions in the Goal-Question-Metric approach, but unlike it, we consider that requirements should be precise enough so that they do not need to be refined. As for the Analytic Hierarchy Process, we also introduce weights to assign relative importance between elements making up the measure.

4.3 Formal representations of measures

Measures are often defined according to a set of criteria, items or questions that we call requirements. These requirements are specifications expressed in natural language. They are structured according to a hierarchy that makes it possible to classify them or, on the contrary, that corresponds to the analysis that led to the emergence of the different questions, for example following a problem analysis approach. To help people to precisely describe and share their measures, both requirements and structure, we introduce the notion of *Tree-Structure of Requirements* (TSoR). The TSoR is not sufficient to completely define the measure. Therefore, in addition to this specification aspect, the definitions we propose consider two other complementary points. First, implementations of the requirements lead to an *Implemented TSoR* and elements for computing a score lead to a *TSoR-based Measure*. Finally, a measure that is fully specified with all these elements is called an Implemented TSoR-based Measure (ITM).

4.3.1 Tree-Structure of Requirements (TSoR)

A Tree-Structure of Requirements (TSoR) consists of structuring requirements, the atomic items to be evaluated, with analysis elements, i.e., concepts that classify requirements. From a technical point of view, a **Tree-Structure of Requirements** (or TSoR) is composed of two kinds of elements and relations. First, the **requirements** should be thought of as a specification for an implementation. They must be expressed in natural language, be as unambiguous as possible, and focus on evaluating a single thing. They are structured according to **analysis elements**, which are a kind of classifier of the requirements (bottom-up vision), or conversely, a refinement of an analysis (top-down vision). These analysis elements are also expressed in natural language and are structured with a relation "**is specified by**". It indicates that an analysis element splits into subelements. Requirements are associated with analysis elements by the relation "**expects**". All together they form a rooted tree.

Definition 4 (Tree-Structure of Requirements). *A **Tree-Structure of Requirements** (TSoR), is a quintuplet $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ such that:*

- *$\mathcal{A}$ is a non-empty finite set of **analysis elements**, with a specific element $\top \in \mathcal{A}$;*
- *$Sp \subseteq \mathcal{A}^2$ is a relation such that $(\alpha, \alpha') \in Sp$ reads "α **is specified by** α' " ;*
- *$\mathcal{R}$ is a set of **requirements** disjoint from $\mathcal{A}$;*
- *$Ex \subseteq (\mathcal{A} \times \mathcal{R})$ is a relation such that $(\alpha, r) \in Ex$ means that the analysis element α **expects** requirement r to be fulfilled ;*
- *$\langle \mathcal{A} \cup \mathcal{R}, Sp \cup Ex, \top \rangle$ is a directed **tree rooted by** $\top$.*

A TSoR reduced to its structure: $\langle \mathcal{A}, Sp, \top \rangle$, is called an **analysis structure**. In the following, the term node refers to both analysis elements and requirements.

Example 2. We define TSoR $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ as:

- $\mathcal{A} = \{\top, \mathrm{A}, \mathrm{B}, \mathrm{A1}, \mathrm{A2}, \mathrm{C}, \mathrm{D}\}$ is the set of analysis elements ;
- $\mathcal{R} = \{i, j, v, w, x\}$ are the requirements ;
- $Sp = \{(\top, \mathrm{A}), (\top, \mathrm{B}), (\mathrm{A}, \mathrm{A1}), (\mathrm{A}, \mathrm{A2}), (\mathrm{B}, \mathrm{C}), (\mathrm{B}, \mathrm{D})\}$ are the refining links between analysis elements ;
- $Ex = \{(\mathrm{A1}, v), (\mathrm{A1}, w), (\mathrm{A2}, x), (\mathrm{A}, y), (\mathrm{D}, i), (\mathrm{D}, j)\}$ are the "expects" links between analysis elements and requirements.

This TSoR is represented graphically in Figure 4.1, where the rectangles are analysis elements and the circles are requirements. Element A expects requirement y and is specified by $\mathrm{A1}$, which expects requirements v and w.

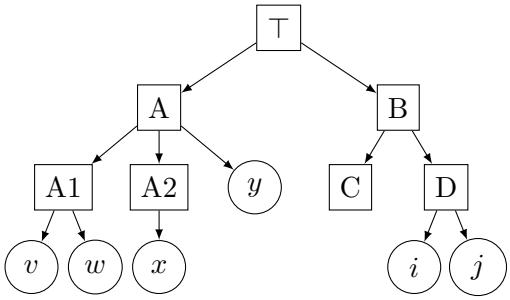


Figure 4.1: Synthetic example of a TSoR

We distinguish the nodes of the TSoR from their contents. The content of a node is its name and, when available, its description, expressed in natural language. It is referred to by the function $\mathrm{content}$. This makes it possible to have several nodes with the same content. For example, following the KGAcc hierarchy, we can have one node named "Who" under a node "Data Collection", another node named "Who" under "Data Maintenance" and another one under "Data Usage". When there is no ambiguity, we will sometimes confuse the nodes with the content. To be strict, we should define the notion of identifier of a node, which we do not for the sake of simplicity.

Notation 1 (Content of a node of a TSoR). *Let b be an analysis element or a requirement. Then $\mathrm{content}(b)$ denotes the content of the node.*

Example 3 (TSoR: Quality of Linked Open Data [69]). The quality evaluation of the LOD cloud as defined by Debattista et al. [69] can be described by a TSoR as illustrated in Figure 4.2. Categories are the first level of the tree, while dimensions are the second level. The last level represents the requirements, called *metrics* by Debattista et al.

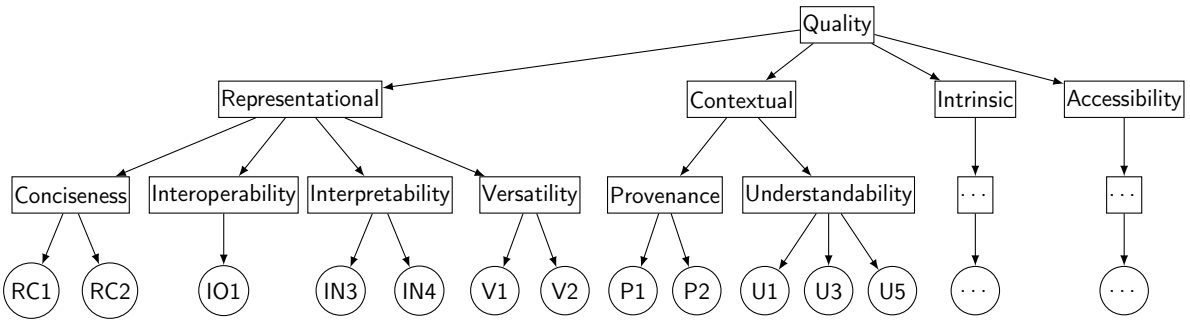


Figure 4.2: TSoR partially representing the data quality metrics and organisation from [69]

This TSoR $\mathcal{T}_Q = \langle \mathcal{A}_Q, \mathcal{R}_Q, Sp_Q, Ex_Q, \top_Q \rangle$ is defined by:

- $\top_Q =$ Quality ;

- $\mathcal{A}_Q = \{\top_Q, \text{Representational}, \text{Contextual}, \text{Intrinsic}, \text{Accessibility}, \text{Conciseness}, \text{Interoperability}, \text{Interpretability}, \text{Versatility}, \text{Provenance}, \text{Understandability}, ...\}$;
- $\mathcal{R}_Q = \{\text{RC1}, \text{RC2}, \text{IO1}, \text{IN3}, \text{IN4}, \text{V1}, \text{V2}, \text{P1}, \text{P2}, \text{U1}, \text{U3}, \text{U5}, ...\}$;
- $Sp_Q = \{(\top_Q, \text{Representational}), (\top_Q, \text{Contextual}), (\text{Representational}, \text{Conciseness}), ...\}$;
- $Ex_Q = \{(\text{Conciseness}, \text{RC1},)(\text{Conciseness}, \text{RC12}), (\text{Interoperability}, \text{IO1}), ...\}$.

The requirements are represented by identifiers, i.e., short names, such as "RC1", but they are also expressed in natural language: "Keeping URIs Short" and with a longer description: "The metric computation is based on the W3C best practices for URIs, where the editor suggests that a URI should not be longer than 80 characters..." [69]. Therefore, $\text{content}(\text{RC1}) =$ "Name: Keeping URIs Short, Description: The metric computation...".

To manipulate a TSoR, we introduce the following notations.

Notation 2. *Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR.*

- $children^{\mathcal{A}}(\alpha, \mathcal{T})$ *is the set of children of α that are analysis elements:*

$$children^{\mathcal{A}}(\alpha, \mathcal{T}) = \{\alpha' \in \mathcal{A} \mid (\alpha, \alpha') \in Sp\}.$$

- $desc^{\mathcal{A}}(\alpha, \mathcal{T})$ *is the set of descendants of α that are analysis elements. Note that an analysis element is not a descendant of itself. It is defined recursively as follows:*

$$desc^{\mathcal{A}}(\alpha, \mathcal{T}) = children^{\mathcal{A}}(\alpha, \mathcal{T}) \cup \{desc^{\mathcal{A}}(\alpha', \mathcal{T}) \mid \alpha' \in children^{\mathcal{A}}(\alpha, \mathcal{T})\}.$$

- $children^{\mathcal{R}}(\alpha, \mathcal{T})$ *is the set of requirements expected by α:*

$$children^{\mathcal{R}}(\alpha, \mathcal{T}) = \{r \in \mathcal{R} \mid (\alpha, r) \in Ex\}.$$

- $desc^{\mathcal{R}}(\alpha, \mathcal{T})$ *is the set of requirements expected by α or its descendants:*

$$desc^{\mathcal{R}}(\alpha, \mathcal{T}) = children^{\mathcal{R}}(\alpha, \mathcal{T}) \cup \{children^{\mathcal{R}}(\alpha', \mathcal{T}) \mid \alpha' \in desc^{\mathcal{A}}(\alpha, \mathcal{T})\}.$$

Definition 4 only imposes constraints that are inherent to the directed rooted tree structure. Indeed, apart from the root, each analysis element specifies exactly one other analysis element, and a requirement is expected by exactly one analysis element (a tree is connected and acyclic). It is possible to define a TSoR that better corresponds to a coherent and fully specified analysis by using recommendations. Some of them use the notion of equivalence of analysis elements or of requirements. Two nodes are equivalent if they are semantically the same.

Due to the difficulty of analyzing and comparing natural language statements, we restrict equivalence to a simple equality of their contents.

Definition 5 (Equivalence between nodes of a TSoR). *Two nodes of a TSoR are equivalent, noted $\equiv$, if they are of the same type (both analysis elements, or both requirements) and if their contents are equal.*

We propose the following recommendations for a TSoR $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$. For a TSoR to be coherent, it is important to specify analysis elements with different and distinct aspects, and the same goes for requirements. Therefore, two sibling nodes should be independent, i.e. cover disjoint aspects. In particular, two equivalent nodes should not be siblings.

Recommendation 1 (Independence of siblings). *Two sibling nodes should be independent, i.e. studying one does not give any information about the other, and vice versa.*

We call a *dead-end* analysis element an analysis element that is a leaf in the tree, i.e., an element that does not lead to anything, neither to another analysis element nor to a requirement. Such an analysis element should be avoided because it cannot lead to an evaluation, nor be satisfied.

Recommendation 2 (No dead-end analysis element). *If $\mathcal{R} \neq \emptyset$, then each analysis element should either be specified or expect a requirement:*

$$\forall \alpha \in \mathcal{A}, (children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset) \vee (children^{\mathcal{R}}(\alpha, \mathcal{T}) \neq \emptyset).$$

Another point to consider concerns the analysis element to which a given requirement is linked. Intuitively, a requirement that is expected by an analysis element could also be expected by all of its ascendants. However, it is useless to make it appear at all levels of the hierarchy. More precisely, a requirement should be associated with the most precise, i.e., the deepest, relevant analysis elements.

Recommendation 3 (Frugality). *A requirement should not be linked to an analysis element if an equivalent one is already expected by one of the descendants of the analysis element.*

$$\forall r \in \mathcal{R}, \forall \alpha \in \mathcal{A}, \Big((\alpha, r) \in Ex\Big) \Rightarrow \Big(\nexists (\alpha', r') \in desc^{\mathcal{A}}(\alpha, \mathcal{T}) \times \mathcal{R}, (\alpha', r) \in Ex \wedge r \equiv r'\Big)$$

When an analysis element is precise enough, it no longer needs to be specified and can expect requirements. Therefore, we consider that an analysis element should not simultaneously be specified and expect a requirement.

Recommendation 4 (Uniformity of children types). *The children of a node should all be of the same type: either analysis element or requirement. Formally:*

$$\forall \alpha \in \mathcal{A}, \left(children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset\right) \Rightarrow \left(children^{\mathcal{R}}(\alpha, \mathcal{T}) = \emptyset\right).$$

The TSoR representing a data quality measure in Example 3 naturally follows all of these recommendations, unlike Example 2. But this is not always the case with existing measures, for example, the O'FAIRe measure of the FAIRness of ontologies [16] does not satisfy Recommendation 4.

4.3.2 TSoR implementation

As defined earlier, a TSoR is just the structuring part of a measure, expressed in natural language. To concretely evaluate a resource based on a TSoR, one needs *(i)* a way to check whether each requirement is fulfilled by the resource, and *(ii)* a function to compute the score of the resource. In this subsection, we propose a formal definition of the first aspect, the second will be defined shortly after.

Given a requirement, an "implementation" is any way to check whether it is fulfilled. From a semantic web user's point of view, several options may be considered: it can denote a SPARQL query, a program, or a combination of the two. The notion of *TSoR implementation* introduced is independent of the chosen option. It uses `IMP` to denote the set of requirement implementations. When a requirement is not implemented, an empty implementation ($\emptyset$) is associated with it.

Definition 6 (TSoR Implementation). *A **TSoR implementation**, noted $\mathcal{T}^I$, is a pair $\mathcal{T}^I = \langle \mathcal{T}, i \rangle$ such that:*

- *$\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ is a TSoR.*
- *$i : \mathcal{R} \to$ `IMP`: is a function identifying the implementation $i(r)$ to use for requirement r w.r.t. to $\mathcal{T}$.*

Example 4 (Implemented TSoR: Quality of Linked Open Data). Let us continue with Example 3 about the TSoR $\mathcal{T}_Q$. The first requirement, called "RC1" corresponding to "Keeping URIs Short", can be computed according to the following metric $RC1$[2] [69]. Let g be an RDF graph, $\mathcal{U}$ be the set of URIs in g, and $dlc(g)$ be the set of subjects or objects of all triples or quads of g if the predicate is not `rdf:type`.

$$RC1(g) = \frac{size(\{u \in \mathcal{U} \cap dlc(g) \mid len(u) \leq 80 \wedge ('?' \notin u)\})}{size(\mathcal{U} \cap dlc(g))}$$

[2]The original metric is $RC1(g) = \frac{size(\{u \in \mathcal{U} | len(u) \leq 80 \wedge ('?' \notin u)\})}{size(\mathcal{U} \cap dlc(g))}$. It has been modified to correspond to the implementation of the measure provided by the authors and to keep its values between 0 and 1.

Its implementation is available online in a specific file "ShortURIs.java"[3] in Java, as for the rest of the metrics.

Thus, the TSoR implementation of $\mathcal{T}_Q$ is: ${\mathcal{T}^I}_Q = \langle \mathcal{T}_Q, i_Q \rangle$, where

- $\mathcal{T}_Q$ is defined in Example 3 ;
- i_Q is a function such that:
  - $i_{RC1} = i_Q(RC1) =$ "ShortURIs.java"
  - $i_{RC2} = i_Q(RC2) =$ "NoProlixRDF.java"
  - $i_{IO1} = i_Q(IO1) =$ "ReuseExistingTerms.java"
  - $i_{IN3} = i(IN3) =$ "UndefinedClassesAndProperties.java"
  - $i_{IN4} = i_Q(IN4) =$ "BlankNodeUsage.java"
  - ... ;

Figure 4.3 shows ${\mathcal{T}^I}_Q$, where the implementations are represented by diamonds.

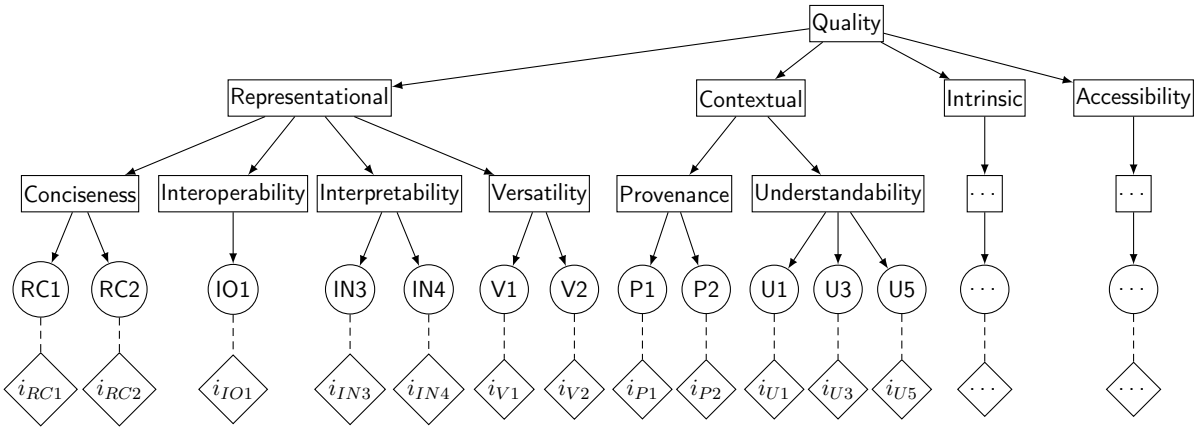


Figure 4.3: Implemented TSoR partially representing the data quality metrics from [69]

However, there is not just a single way to implement them. For instance, requirement "RC1" can also be expressed with the SPARQL query q_{RC1}, as shown in Listing 4.1. This would correspond to a different implemented TSoR based on the same TSoR $\mathcal{T}_Q$. Indeed, eight of the metrics described in [69] are implemented with SPARQL queries[4] in IndeGx [17]. The corresponding implemented TSoR is represented in Figure 4.4. If a requirement r has no diamond associated with it, it means that it is not implemented (so $i(r) = \emptyset$).

```
1 SELECT ((?nb_uri_short/?nb_uri) AS ?RC1) WHERE {
2   { SELECT (COUNT(?uri) AS ?nb_uri_short) WHERE {
```

[3] https://github.com/Luzzu/Metrics/blob/master/linked-data-quality-metrics/representation/src/main/java/io/github/luzzu/linkeddata/qualitymetrics/representational/conciseness/ShortURIs.java

[4] https://github.com/Wimmics/IndeGx/tree/main/rules/IndeGx_original_rules/

```
    { ?uri ?p ?o } UNION { ?s ?p ?uri }
    FILTER(isIRI(?uri) && ?p != rdf:type )
    FILTER(! CONTAINS(STR(?uri), "?")
           && STRLEN(STR(?uri)) < 80 )
  } }
  { SELECT (COUNT(?uri) AS ?nb_uri) WHERE {
    { ?uri ?p ?o } UNION { ?s ?p ?uri }
    FILTER(isIRI(?uri) && ?p != rdf:type )
  } }
}
```

Listing 4.1: SPARQL implementation from [17] of metric RC1 of data quality: "Keeping URIs short"

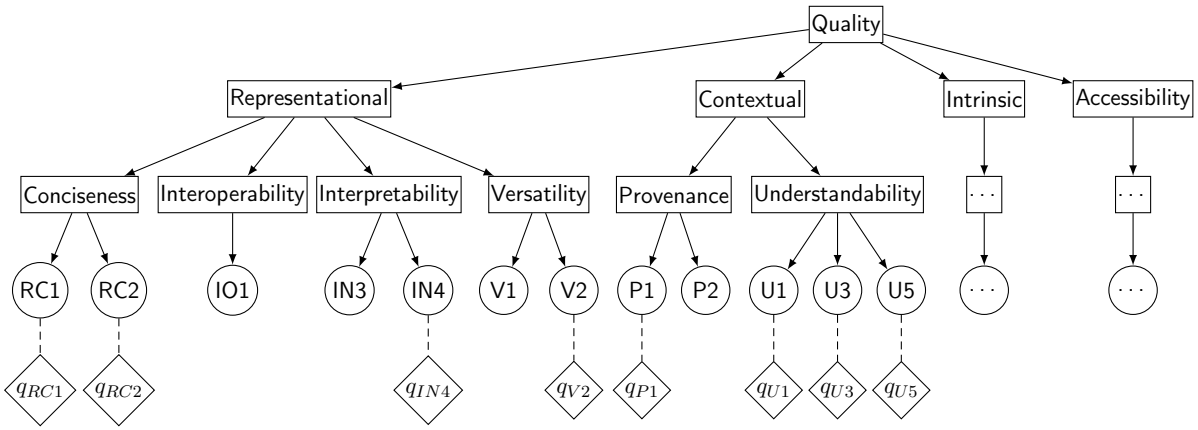


Figure 4.4: Implemented TSoR partially representing the data quality metrics from [69] and implemented by [17]

By hypothesis, we consider that all implementations are perfect, i.e., they fully comply with the requirements. However, this may not always be the case. If one wants to take this into account, he or she must introduce another indicator of quality or coverage. Such an indicator may be very difficult to estimate and may be highly subjective and contextual. In addition, to avoid misleading users about what is actually being measured, all requirements should be implemented. In general, we consider the following recommendation.

Recommendation 5 (Complete implementation). *Each requirement should be fully implemented.*

$$\forall r \in \mathcal{R}, i(r) \neq \emptyset$$

Example 5. The original implemented TSoR concerning the data quality measure ${\mathcal{T}^I}_Q$, defined in Example 4, satisfies Recommendation 5 but this is not the case for the adaptation proposed by IndeGx.

4.3.3 TSoR-based measure

A TSoR details and organizes requirements. To be able to measure and give a score to a knowledge graph or any digital resource (a dataset, a software...) based on this TSoR, we introduce other elements. We decorate the TSoR with weights that express the relative importance of each node with respect to its siblings. We also introduce an aggregation function that expresses how to compute a unique global score based on the result obtained for each requirement.

Definition 7 (TSoR-based Measure). *A **TSoR-based measure**, noted $\mathcal{T}_M$, is a triple $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ defined as follows.*

- *$\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ is a TSoR.*
- *$\delta : Sp \cup Ex \to \mathbb{R}^+$ is a function specifying the weight of an edge (α, b), where α is an analysis element and b is either an analysis element or a requirement. It indicates the relative importance of b with respect to its siblings, i.e., the other children of α.*
- *$\diamondsuit : \mathcal{P}(\mathbb{R}^2) \to \mathbb{R}$, is an aggregation function, where $\mathcal{P}(\mathbb{R}^2)$ is the power set of $\mathbb{R}^2$. An element of $\mathcal{P}(\mathbb{R}^2)$ represents a set of couples (v, w), where v is a value to be aggregated and w its weight.*

Let α be an analysis element and r, r' be requirements, $\delta(\alpha, r) = x \times \delta(\alpha, r')$ means that r is x times more important than r' relatively to α. The same applies for analysis elements.

An extension of this definition could be to assign an aggregation function to each analysis element instead of this global aggregation function. However, we believe that this brings more difficulties both in understanding and in defining a TSoR. Let us present a non-exhaustive list of aggregation functions that can be used.

Definition 8 (Aggregation functions). *Let $V \in \mathcal{P}(\mathbb{R}^2)$ be a non-empty finite set of couples (v, w), where v represents a value and w its associated weights.*

- *The **Weighted Average**, noted $\underset{\text{WA}}{\diamondsuit}$:*

$$\underset{\text{WA}}{\diamondsuit}(V) = \frac{\sum\limits_{(v,w)\in V} v \times w}{\sum\limits_{(v,w)\in V} w}$$

- *The non-weighted **Average**, noted $\underset{\text{A}}{\diamondsuit}$:*

$$\underset{\text{A}}{\diamondsuit}(V) = \frac{\sum\limits_{(v,w)\in V} v}{size(V)}$$

- *The **Weighted Sum**, noted $\underset{\text{WS}}{\diamondsuit}$:*

$$\underset{\text{WS}}{\diamondsuit}(V) = \sum_{(v,w)\in V} v \times w$$

- *The non-weighted **Sum**, noted $\underset{\text{S}}{\diamondsuit}$:*

$$\underset{\text{S}}{\diamondsuit}(V) = \sum_{(v,w)\in V} v$$

In addition, these aggregation functions give 0 for an empty set: $\underset{\text{WA}}{\diamondsuit}(\emptyset) = \underset{\text{A}}{\diamondsuit}(\emptyset) = \underset{\text{WS}}{\diamondsuit}(\emptyset) = \underset{\text{S}}{\diamondsuit}(\emptyset) = 0$.

In the definition of a TSoR-based measure, the weights are held by the edges and not by the node itself. Indeed, it depends on both the node and its parent: it is a local weight to be compared with that of its siblings.

Let us first illustrates the TSoR-based measure with a synthetic example.

Example 6 (TSoR-based measure). Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR, and $\mathcal{T}_M = \langle \mathcal{T}, \underset{\text{WA}}{\diamondsuit}, \delta \rangle$ a TSoR-based measure such that:

- $\mathcal{A} = \{\top, \mathrm{A}, \mathrm{B}\}$ with $\top$ the root of the TSoR
- $\mathcal{R} = \{v, w, x, y, z\}$
- $Sp = \{(\top, \mathrm{A}), (\top, \mathrm{B})\}$
- $Ex = \{(\mathrm{A}, v), (\mathrm{A}, w), (\mathrm{B}, x), (\mathrm{B}, y), (\mathrm{B}, z)\}$
- δ is defined as follows: $\delta(\top, \mathrm{A}) = 2$, $\delta(\top, \mathrm{B}) = 1$, $\delta(\mathrm{A}, v) = 1$, $\delta(\mathrm{A}, w) = 1$, $\delta(\mathrm{B}, x) = 2$, $\delta(\mathrm{B}, y) = 4$, $\delta(\mathrm{B}, z) = 2$

The tree representation is shown in Figure 4.5. Let us comment on this TSoR-based measure. v and w are equally important relatively to A, indeed their edges connecting them to A both have a weight of 1. Similarly, y is twice as important as x and z relatively to B. But x having a weight of 2 and v of 1 does not mean that x is more important than v. Indeed, weights are to be understood only compared to its siblings.

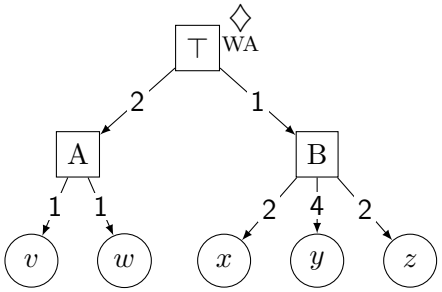


Figure 4.5: Simple example of a TSoR-based measure

Some measures have a hierarchy to organize their metrics, but do not use it to compute a global score. They only assign weights to requirements. For them to be represented in our formalism, it is necessary to assign weights to the analysis elements. To stay close to the original definition of the measure, we can either use a weighted sum as the aggregation function and a weight of 1 for "is specified by" relations, or use a weighted average and derive the weights as follows. Here is how one can proceed.

Method to calculate δ. This method holds to represent a measure in our formalism with a weighted average when the measure has only weights for its requirements. Let $\mathcal{T}$

be a TSoR and the aggregation function $\diamondsuit$ be the weighted average. Keeping the usual notation, let $\omega : \mathcal{R} \to \mathbb{R}^+$ be a function that assigns a weight to each requirement. Then the weight function δ is defined recursively as follows.

$$\begin{cases} \forall (\alpha, r) \in Ex, \quad \delta(\alpha, r) = \omega(r) \\ \forall (\alpha, \alpha') \in Sp, \quad \delta(\alpha, \alpha') = \displaystyle\sum_{b \in children^{\mathcal{A}}(\alpha', \mathcal{T}) \cup children^{\mathcal{R}}(\alpha', \mathcal{T})} \delta(\alpha', b) \end{cases}$$

Thus, if an analysis element is a dead-end (has no requirement and no analysis element), its weight is 0. Let us illustrate this by continuing the example on data quality.

Example 7 (TSoR-based measure: Quality of the LOD cloud). According to Debattista et al. [69], let $scr_m(g)$ be the score obtained by a KG g on a given metric m, then the aggregated data quality score is computed as follows:

$$quality(g) = \frac{\displaystyle\sum_{m \in \{RC1, ..., PE3\}} scr_m(g)}{size(\{RC1, ..., PE3\})}$$

Rewriting this in our own notations, the set of all metrics RC1 to PE3 is the set of requirements:

$$quality(g) = \frac{\displaystyle\sum_{m \in \mathcal{R}} scr_m(g)}{size(\mathcal{R})}$$

To model this with a TSoR-based measure, the aggregation function is, naturally, the weighted average. Then, the above function indicates that all requirements have the same importance when calculating the global quality score. To preserve both the hierarchy and this importance, we apply the previous method, assigning a weight of 1 to the requirements and to each analysis element the sum of the weights of its children.

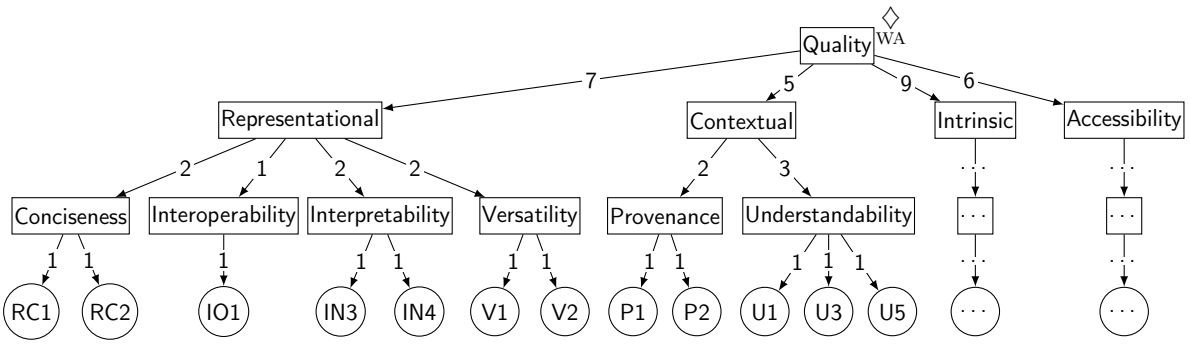


Figure 4.6: TSoR partially representing the data quality metrics and organisation from [69]

4.3.4 Implemented TSoR-based Measure (ITM)

Having both implementations and weights allows to automatically evaluate a digital resource. A TSoR that is extended as both a TSoR-based measure and a TSoR implementation is called an Implemented TSoR-based Measure.

Definition 9 (Implemented TSoR-based Measure (ITM)). *An **Implemented TSoR-based Measure**, or simply an ITM, is a tuple $\mathcal{T}_M^I = \langle \mathcal{T}, \diamondsuit, \delta, i \rangle$ where $\langle \mathcal{T}, i \rangle$ is a TSoR implemention and $\langle \mathcal{T}, \diamondsuit, \delta \rangle$ is a TSoR-based measure.*

In order to measure to what extent a digital resource satisfies an ITM, we introduce the ITM-based score. To do so, we need to be able to refer to the result obtained by the digital resource with respect to an implementation of a requirement. This is why we define first an evaluation function.

Definition 10 (Evaluation of a digital resource over an implementation). *The evaluation of a digital resource d over the implementation i of a requirement r is noted $\mathrm{eval}(d, i, r)$. It is the result of the execution of $i(r)$ on d, normalized between 0 and 1.*

Using semantic web technologies, an implementation may be SPARQL queries. Then, the evaluation can simply be the result of the execution: the value is 1 if True is obtained with a SPARQL ASK query, and 0 otherwise. In other cases, the query may compute the evaluation directly, as for the query q_{RC1} about keeping URIs short (see Listing 4.1, page 75), or it can be necessary to introduce post-processing to calculate the evaluation based on the result of the query. We can now define the ITM-based score. It is then illustrated for the weighted average for a better understanding.

Definition 11 (ITM-based score of a digital resource). *Given a function eval and an ITM $\mathcal{T}_M^I = \langle \mathcal{T}, \diamondsuit, \delta, i \rangle$, the score of a digital resource d is defined as follows:*

$$\mathrm{score}(d, \mathcal{T}_M^I, \mathrm{eval}) = \mathrm{score}(d, \mathcal{T}_M^I, \mathrm{eval}, \top)$$

where, the score for a given node $b \in \mathcal{A} \cup \mathcal{R}$ is defined recursively:

$$\mathrm{score}(d, \mathcal{T}_M^I, \mathrm{eval}, b) = \begin{cases} \mathrm{eval}(d, i, b) & \text{if } b \in \mathcal{R} \\ \diamondsuit \left(\left\{ \left(\mathrm{score}(d, \mathcal{T}_M^I, \mathrm{eval}, b'), \delta(b, b') \right) \mid b' \in children^{\mathcal{A} \cup \mathcal{R}}(b, \mathcal{T}) \right\} \right) & \text{otherwise} \end{cases}$$

with, if $b \in \mathcal{A}$, $children^{\mathcal{A} \cup \mathcal{R}}(b, \mathcal{T})$ is the set of children of b that are either in $\mathcal{A}$ or in $\mathcal{R}$.

Example 8 (ITM-based score of a digital resource using a weighted average). Let $\mathcal{T}_M^I$ be an ITM, where the aggregation function is the weighted average $\underset{\mathrm{WA}}{\diamondsuit}$. Let d be a digital resource,

then developing the weighted average, the score of d at a node $b \in \mathcal{A} \cup \mathcal{R}$ is:

$$\text{score}(d, \mathcal{T}_M^I, \text{eval}, b) = \begin{cases} \text{eval}(d, i, b) & \text{if } b \in \mathcal{R} \\ 0 & \text{if } b \in \mathcal{A} \wedge children^{\mathcal{A}\cup\mathcal{R}}(b, \mathcal{T}) = \emptyset \\ \dfrac{\sum\limits_{b' \in children^{\mathcal{A}\cup\mathcal{R}}(b,\mathcal{T})} \delta(b,b') \cdot \text{score}(d, \mathcal{T}_M^I, \text{eval}, b')}{\sum\limits_{b' \in children^{\mathcal{A}\cup\mathcal{R}}(b,\mathcal{T})} \delta(b,b')} & \text{otherwise} \end{cases}$$

Let us discuss the recommendation about no dead-end analysis element (Recommendation 2). If it is not followed, and if the weights of the problematic elements are not 0, then the maximum score (1 in the case of an average) is not attainable. The same behavior occurs for unimplemented requirements, if Recommendation 5 is not followed. There are several ways to treat unimplemented requirements: either the evaluation of a resource against them returns 0, because the resource cannot prove that it satisfies the requirement. Or it can be treated as 1 by default, because there is no evidence that it does not satisfy it. Or it can be considered separately as a value not applicable. The first and last cases do not allow the maximum value to be obtained.

4.3.5 Analysis of an ITM

To better understand the elements involved in an ITM, or simply in a TSoR-based measure, and their importance in the global score, we define the impact. The impact of a node, either a requirement or an analysis element, refers to its importance in the TSoR. It quantifies how much a node contributes to the calculation of the global score. More precisely, the positive impact of an analysis element is the score obtained if all requirements descending from it are evaluated as 1 and all others are evaluated as 0. The negative impact of an analysis element is the score obtained when all requirements descending from it are evaluated as 0 and all others are evaluated as 1. This latter is not formally defined here. In the following, we will refer to the impact as the positive impact. To define it, we first define the following evaluation function.

Definition 12 (Positive evaluation of the requirements under a given node). *Let $\mathcal{T}$ be a TSoR and $b \in \mathcal{A} \cup \mathcal{R}$ be a node of the TSoR. eval_b is the positive evaluation of b, that gives 1 for b or all the requirements under b, and 0 for all other requirements, independently of the resource and the implementation considered. To comply with the definition of a score, three parameters*

are required, the first two of which, d and i are ignored.

$$\mathrm{eval}_b(d, i, r) = \begin{cases} 1 & \textit{if } (b \in \mathcal{R}) \wedge (b = r) \\ 1 & \textit{if } (b \in \mathcal{A}) \wedge (r \in desc^{\mathcal{R}}(b, \mathcal{T})) \\ 0 & \textit{otherwise} \end{cases}$$

Definition 13 (Impact of a node). *Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ be a TSoR-based measure. The (positive) impact of a node $b \in \mathcal{A} \cup \mathcal{R}$ over $\mathcal{T}_M$ is noted* impact *and defined by:*

$$\mathrm{impact}(\mathcal{T}_M, b) = \mathrm{score}(\emptyset, \mathcal{T}_M, \mathrm{eval}_b)$$

Example 9 (Impact of some nodes over the TSoR-based measure representing data quality). To illustrate the impact, let us return to Example 7, page 79. Applying the previous definition, we obtain that each requirement has the same impact: $\frac{1}{27}$, which is consistent with the original definition of the measure, which computes the global score as a non-weighted average on all requirements.

We also compute the impact of the analysis elements. For instance, we obtain an impact of $\frac{2}{27}$ for the "Conciseness" node. We observe that the sum of the weights of the analysis elements at the same depth is 1, because the aggregation function is a weighted average and all analysis elements lead to a requirement.

The impact of the root $\top$ over a TSoR-based measure is also the achievable score over the TSoR. If the aggregation function is an average, then an achievable score of less than 1 means that some analysis elements do not lead to any requirement, in contradiction to Recommendation 2.

We also define the impact of a set of equivalent nodes. This is the score obtained when all requirements under the targeted node and under nodes equivalent to the targeted node are evaluated as 1, and all others are evaluated as 0.

Definition 14 (Impact of a set of equivalent nodes). *Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ be a TSoR-based measure of a TSoR $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$, and $b \in \mathcal{A} \cup \mathcal{R}$ be a node of the TSoR.* evalEq_b *is the positive evaluation of the set of nodes equivalent to b. Three parameters are required, the first two of which, d and i are ignored.*

$$\mathrm{evalEq}_b(d, i, r) = \begin{cases} 1 & \textit{if } b \in \mathcal{R} \wedge b \equiv r \\ 1 & \textit{if } b \in \mathcal{A} \wedge r \in desc^{\mathcal{R}}(b, \mathcal{T}) \\ 1 & \textit{if } b \in \mathcal{A} \wedge (\exists b' \in \mathcal{A}, b \equiv b' \wedge r \in desc^{\mathcal{R}}(b', \mathcal{T})) \\ 0 & \textit{otherwise} \end{cases}$$

The impact of a set of equivalent nodes of $b \in \mathcal{A} \cup \mathcal{R}$ over $\mathcal{T}_M$ is noted $\mathrm{impactEq}$ *and defined by:*

$$\mathrm{impactEq}(\mathcal{T}_M, b) = \mathrm{score}(\emptyset, \mathcal{T}_M, \mathrm{evalEq}_b)$$

From the definition of the TSoR to the definition of the ITM, we propose a framework for describing a measure with several levels of detail. This is intended to be general enough to express most existing measures. To help users to reuse and modify existing measures or to adapt their measure to recommendations, we define some operators on the TSoR in the following section.

4.4 Manipulation of TSoRs and ITMs

Before using an ITM to evaluate a digital resource, it must obviously be built first. It can either be built from scratch, starting with a root and then adding analysis elements and requirements, weighting elements, and implementations. Or it can be built by relying on one or more existing TSoRs or ITMs.

Therefore, numerous construction operators can be defined, from the construction of a TSoR reduced to its root, to the addition of a new analysis element or requirement as a child of an existing node, to the modification and deletion of nodes, weights, and implementations. We leave these operators aside, as they are not much more than classical tree operators. Instead, we focus on the reuse of existing ITMs to propose more sophisticated operators that aim to simplify some elaborate tasks of modifying them and creating new ones based on existing ones.

Therefore, we first focus on simplifying ITMs to make them satisfy some of the recommendations introduced earlier by proposing some "cleaning operators". Then, we propose some "building operators" to create new ITMs based on existing ones, for instance by focusing on some specific aspect of an ITM.

4.4.1 Adaptation to recommendations

Some TSoRs and ITMs may not follow all the recommendations we defined in sections 4.3.1 and 4.3.2 (pages 73 and 76), whether intentionally or not. For example, the analysis to create a TSoR may result in a very demanding analysis structure (the subtree of analysis elements) whereas only a few analysis elements lead to requirements. Nevertheless, it is possible to simplify TSoRs and ITMs to make them satisfy recommendations. First, Recommendation 2 (see page 73) states that each analysis element should lead to a requirement. When this recommendation is not followed, some analysis elements may appear to be useless. For instance, in Figure 4.7, the TSoR on the left has several analysis elements that do not expect any requirement, such as $\mathrm{A2}$, $\mathrm{C1}$ and $\mathrm{C2}$, or that do not lead to a requirement, such as

C. We introduce a simplification operator which removes all these analysis elements. As all the following operators, it is defined for a TSoR, an implemented TSoR and a TSoR-based measure, therefore, it is also defined for an ITM.

Definition 15 (TSoR simplification). *The **simplification** of a TSoR $\mathcal{T}$, noted* $\mathrm{Simplify}(\mathcal{T})$*, removes all its dead-end analysis elements, i.e., all analysis elements without any requirement in their descendants. Formally, let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR.* $\mathrm{Simplify}(\mathcal{T}) = \langle \mathcal{A}', \mathcal{R}, Sp', Ex, \top \rangle$ *is defined by:*

- $\mathcal{A}' = \{\alpha \in \mathcal{A} \mid desc^{\mathcal{R}}(\alpha, \mathcal{T}) \neq \emptyset\}$;
- $Sp' = Sp \cap (\mathcal{A}' \times \mathcal{A}')$.

*Let $\mathcal{T}^I = \langle \mathcal{T}, i \rangle$ be an implemented TSoR of $\mathcal{T}$. The **simplification** of $\mathcal{T}^I$ to remove dead-end analysis elements, noted* $\mathrm{Simplify}(\mathcal{T}^I)$*, is the pair* $\langle \mathrm{Simplify}(\mathcal{T}), i \rangle$ *with no change in i.*

*Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ be a TSoR-based measure of $\mathcal{T}$. The **simplification** of $\mathcal{T}_M$ to remove dead-end analysis elements, noted* $\mathrm{Simplify}(\mathcal{T}_M)$*, is* $\langle \mathrm{Simplify}(\mathcal{T}), \diamondsuit, \delta' \rangle$*, where:*

- $\diamondsuit$ *does not change ;*
- $\delta' = \delta|_{Sp' \cup Ex}$ *is the restriction of the function δ to $Sp' \cup Ex$.*

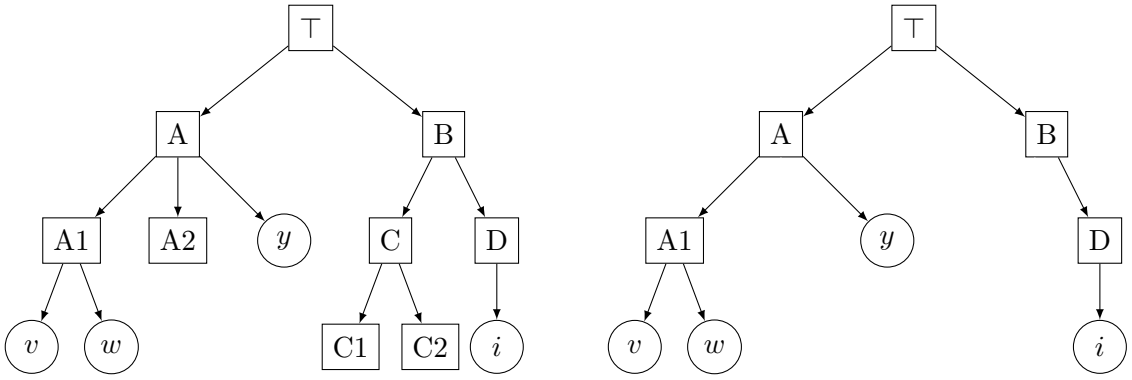


Figure 4.7: Simple example of a TSoR (left) and its simplification (right)

Example 10. The simplification of the TSoR on the left side of Figure 4.7 is shown on the left side of the same figure.

Property 1. *The simplification of a TSoR is a TSoR satisfying Recommendation 2. Similar results hold for an implemented TSoR and a TSoR-based measure.*

Proof. Let $\mathcal{T}$ be a TSoR. Let us show that $\mathrm{Simplify}(\mathcal{T})$ is a TSoR. First, $\mathcal{A}' \subseteq \mathcal{A}$, so it is a set of analysis elements, and Sp is well defined on $\mathcal{A}' \times \mathcal{A}'$. Next, note that if an analysis

element is removed, then so are all of its descendants. As a consequence, $\mathrm{Simplify}(\mathcal{T})$ remains connected. Since it is a subgraph of the tree $\mathcal{T}$, it is also a tree, and thus a TSoR. Without difficulty, the result extends to an implemented TSoR and a TSoR-based measure.

Let $\alpha \in \mathcal{A}'$, then by definition of $\mathrm{Simplify}$, $\exists r \in \mathcal{R}$ such that $r \in desc^{\mathcal{R}}(\alpha, \mathcal{T})$, and $r \in desc^{\mathcal{R}}(\alpha, \mathrm{Simplify}(\mathcal{T}))$. So either r is a child of α in $\mathrm{Simplify}(\mathcal{T})$, or there is an analysis $\alpha' \in children^{\mathcal{A}}(\alpha, \mathrm{Simplify}(\mathcal{T}))$ such that $r \in desc^{\mathcal{R}}(\alpha', \mathcal{T})$. This shows that $\mathrm{Simplify}(\mathcal{T})$ satisfies Recommendation 2. □

Similarly, we can easily remove all requirements without an implementation to satisfy Recommendations 5 (see page 76). We will skip the formal definition here, but the idea is to keep only requirements with an implementation (i.e., that is not $\emptyset$). To still satisfy Recommendation 2, one has to compose this latter operation with the simplification to remove dead-end analysis elements.

Recommendation 4 (see page 74) requires uniformity of child node types. There are several ways to solve this problem. The most brutal one would be to remove the problematic requirements. But, we can imagine that these are present for a good reason. Instead, we suggest adding a "buffer" analysis element between these requirements and their parent. This new analysis element will have only requirements, and its parent will have only analysis elements. The corresponding definition is given in Appendix A.4. Let us show how it works.

Example 11 (Modification of a TSoR to satisfy Recommendation 4). Let a TSoR-based measure be the one shown on the left side of Figure 4.8. Its modification for uniform children types is shown on the right side of the same figure. In the original TSoR, A has two analysis elements and two requirements among its children. So a new analysis element A' is added between A and the requirements it expects. The relative importance between y and z are preserved. To maintain their importance relatively to $\mathrm{A1}$ and $\mathrm{A2}$, the weight $(\mathrm{A}, \mathrm{A}')$ is 3, the sum of the weights of x and y.

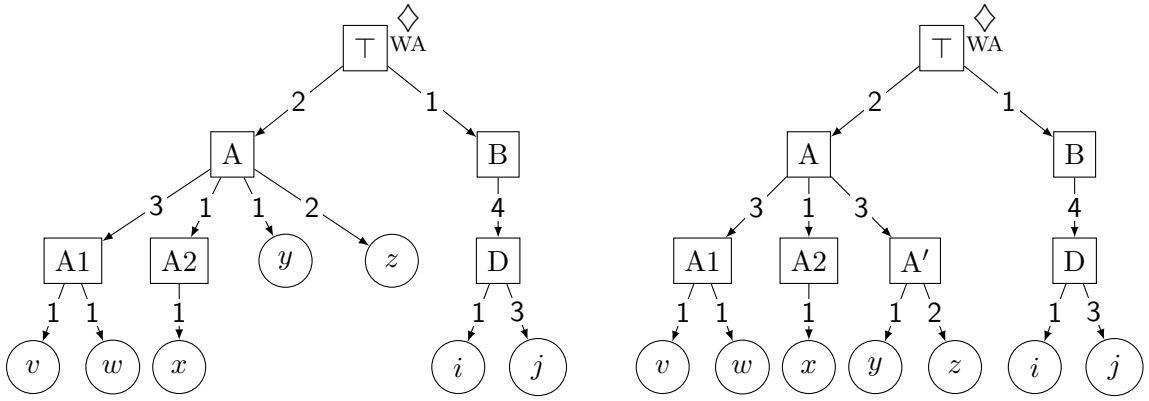


Figure 4.8: Simple example of a TSoR (left) and its modification to uniform children types (right)

Finally, two recommendations remain, and an operator could be defined for each of them. Recommendation 3 (page 73) demands that a requirement should not be expected by a node α if an equivalent requirement is expected by another node α' that is a descendant of α, i.e., at a more refined level. Therefore, the idea of another simplification operator would be to remove all such "duplicate" requirements. Another operator could be defined concerning the independence of siblings, to improve a TSoR regarding Recommendation 1, page 73. It would combine the equivalent siblings into a single node that has the children of both nodes. Defining these two operators formally would not be easy, as they require careful management of implementations and weights. Indeed, removing duplicate requirements or combining equivalent ones would require selecting which implementation to keep. Formally, it is possible to choose one arbitrarily, in practice it would be more interesting to put the users in the loop and let them choose. Similarly, defining the weight of the remaining requirement is not straightforward.

We choose to skip the definition of these operators and concentrate on construction operators. As a reminder, we decided to leave aside the definition of the classical tree operators to focus on more sophisticated operators that are specific to TSoRs and ITMs. So in the following sections, we aim to provide some tools to reuse existing ITMs to define new ones. To do so, we define two kinds of operators. The first ones consider a subpart of an ITM, while the second ones extend an ITM with another one.

4.4.2 Focusing on a subpart of an ITM

First, we introduce operators that allow to focus only on some analysis elements. There are several ways to do this, we present two of them. The first one focuses on a sub-structure: selected analysis elements, their ancestors (analysis elements) and all their child requirements are preserved. This reduces the scope of the analysis, but leaves the organization the same. The associated operator is called a *restriction*. The second focuses on the analysis elements: only the selected elements are preserved regardless of the hierarchy. More precisely it focuses on a set of concepts by combining the analysis elements equivalent to those selected. So the TSoR is flattened with the selected elements attached to the root, and the latter are linked to all requirements descending from them or their equivalents. The operator is a *reduction*. We first define a restriction operator on an ITM and then the reduction. All following construction operators are defined for TSoRs and extended to implemented TSoRs and TSoR-based measures. In this way, the operators are also specified for ITMs.

Restriction

A restriction of an ITM enables to focus on a part of its analysis structure. It is like a cut in the tree formed by the ITM. The part is specified by the set of analysis elements to keep. In practice, it is sufficient to specify the most precise ones (the deepest in the tree), the complete paths from the root to these elements being also kept. More precisely, the restriction is the smallest weighted subtree of the TSoR that contains (1) all of the specified elements, and (2) all of the requirements attached to a retained analysis element in the subtree. The implemented TSoR and the TSoR-based measure are easily restricted since their definitions only use functions that can be restricted to the subset of retained analysis elements, requirements, or relations.

Definition 16 (Restriction of a TSoR). *Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR. Let S be a non empty set such that, $S \subseteq \mathcal{A}$. The **restriction** of $\mathcal{T}$ with respect to S, noted $\mathrm{Restriction}(\mathcal{T}, S) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top \rangle$ is formally defined by:*

- $\mathcal{A}' = S \cup \{\alpha \in \mathcal{A} \mid \exists \alpha' \in S, \alpha' \in desc^{\mathcal{A}}(\alpha, \mathcal{T})\}$;
- $\mathcal{R}' = \bigcup_{\alpha \in \mathcal{A}'} children^{\mathcal{R}}(\alpha, \mathcal{T})$;
- $Sp' = Sp \cap (\mathcal{A}' \times \mathcal{A}')$;
- $Ex' = Ex \cap (\mathcal{A}' \times \mathcal{R}')$;
- $\top$ *is the same root as in* $\mathcal{T}$.

*Let $\mathcal{T}^I = \langle \mathcal{T}, i \rangle$ be an implemented TSoR of $\mathcal{T}$. The **restriction** of $\mathcal{T}^I$ with respect to S is* $\mathrm{Restriction}(\mathcal{T}^I, S) = \langle \mathrm{Restriction}(\mathcal{T}, S), i|_{\mathcal{R}'} \rangle$.

*Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ be a TSoR-based measure of $\mathcal{T}$. The **restriction** of $\mathcal{T}_M$ with respect to S is* $\mathrm{Restriction}(\mathcal{T}_M, S) = \langle \mathrm{Restriction}(\mathcal{T}, S), \diamondsuit, \delta|_{Sp' \cup Ex'} \rangle$.

Example 12. Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ be the TSoR-based measure shown on the left side of Figure 4.9. Its restriction with respect to $S = \{\mathrm{A1}, \mathrm{B}\}$ is given on the right side of the same figure. We observe that all the requirements of $\mathrm{A1}$ are preserved. Since $\mathrm{A1}$ is in S, then A is kept, as well as its requirement y. B has no requirement, but is mentioned in S, so it also appears in the result. $\mathrm{A2}$, C, and D are neither in S nor in the ancestors of the nodes in S, so they are removed, just like their requirements. It is as if these branches were cut off the tree. Finally, the weights are identical as before, indeed the relative importance of $\mathrm{A1}$ compared to y has no reason to change with the removal of $\mathrm{A2}$.

Property 2. *The restriction of a TSoR is a TSoR. Similarly, the restriction of an implemented TSoR is also an implemented TSoR. And the restriction of a TSoR-based measure is a TSoR-based measure.*

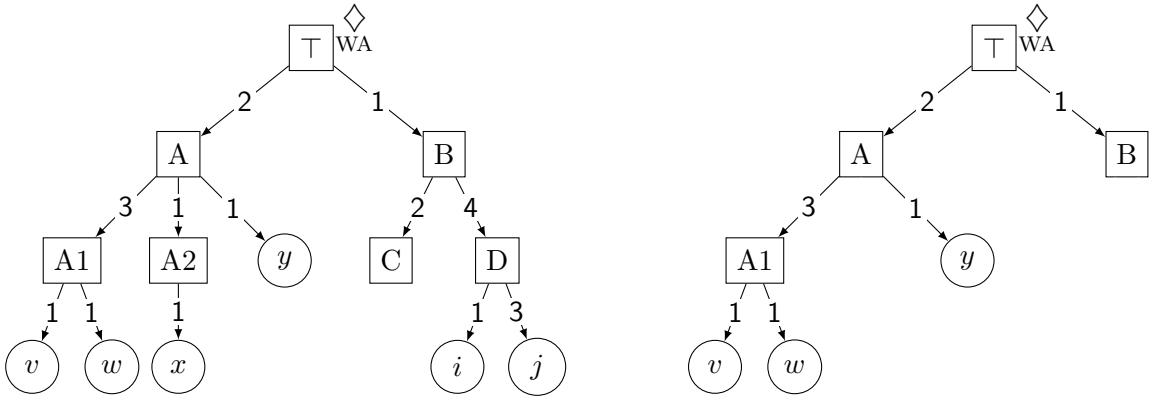


Figure 4.9: Simple example of a TSoR (left) and its restriction w.r.t. $\{\mathrm{A1}, \mathrm{B}\}$ (right)

Proof. Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR and $S \subseteq \mathcal{A}$ be a non-empty set of analysis elements. Let us show that $\mathrm{Restriction}(\mathcal{T}, S) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top \rangle$ is a TSoR. $\mathcal{A}', \mathcal{R}', Sp'$, and Ex' trivially satisfy the conditions of being a set of analysis elements, a set of requirements, and relations between them, respectively. In addition, since S is not empty, and $S \subseteq \mathcal{A}'$, then $\mathcal{A}'$ is not empty. Moreover, $\forall \alpha \in S, \alpha \in desc^{\mathcal{A}}(\alpha, \mathcal{T})$, so $\top \in \mathcal{A}'$.

The only thing to show is that $\mathcal{G} = \langle \mathcal{A}' \cup \mathcal{R}', Sp' \cup Ex', \top \rangle$ is a directed tree rooted by $\top$. Since $\mathcal{G}$ is a subgraph of $\langle \mathcal{A} \cup \mathcal{R}, Sp \cup Ex, \top \rangle$, which is a directed rooted tree, we only need to show that $\mathcal{G}$ is still a connected graph. Let $\alpha \in \mathcal{A}' \setminus \top'$. If $\alpha \in S$, then all its ancestors α' in $\mathcal{T}$ belong to $\mathcal{A}'$ because $\alpha \in desc^{\mathcal{A}}(\alpha', \mathcal{T})$ and by definition of $\mathcal{A}'$. So by construction of Sp, there exists a path from α to $\top$. Otherwise, if $\alpha \notin S$, then $\exists \alpha' \in S$ such that $\alpha' \in desc^{\mathcal{A}}(\alpha, \mathcal{T})$. Since all ancestors of α are ancestors of α', this means that all ancestors of α in $\mathcal{T}$ belong to $\mathcal{A}'$. Then by construction of Sp', there exists a path from α to $\top$. Therefore, $\langle \mathcal{A}', Sp', \top' \rangle$ defines a tree rooted at $\top$. Finally, since only requirements that are children of an analysis element in $\mathcal{A}'$ are preserved, all requirements of $\mathcal{R}'$ are connected to $\langle \mathcal{A}', Sp', \top \rangle$, and $\mathcal{G}$ is a directed tree rooted by $\top$.

As for the implemented TSoR and the TSoR-based measure, their restriction is defined by restricting the existing functions to the correct domains of definition. Therefore, they are trivially implemented TSoR and TSoR-based measure, respectively. □

This restriction operator may be useful to focus on a specific part of a measure, or of an analysis. For instance, the definition of the measure of accountability in Chapter 3 does not consider the entire LiQuID hierarchy. Indeed, one life cycle step (data processing) has been removed, as well as some question types (why, and some of the what) and all aspects of the information level except "description". These changes can be seen as the result of a restriction.

Reduction

Unlike the restriction, the reduction does not reflect the organization of the initial TSoR analysis. It aims to focus on a set of concepts (in the sense of equivalent analysis elements) and to provide a more succinct view of them and their associated requirements. Therefore, the number of levels of analysis elements is reduced to a single one, with all selected analysis elements attached to the root. Analysis elements are combined by equivalence. Thus, for each selected analysis element, all requirements descending from it or from one of its equivalent analysis elements are preserved and linked to it. In this way, it becomes possible to reduce a TSoR to a coherent whole by keeping only analysis elements of the same nature, e.g. only WH questions or the lifecycle steps. To do this, a user must provide the set S of analysis elements to study with two conditions: there must not be two equivalent analysis elements in S, and a descendant of an analysis element in S cannot be itself in S nor equivalent to a node in S.

Definition 17 (Reduction of a TSoR). *Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR and $S \subseteq \mathcal{A}$ such that:*

- $\nexists \alpha, \alpha' \in S$ *such that* $\alpha \equiv \alpha'$*, and*
- $\forall \alpha \in S, \forall \alpha' \in desc^{\mathcal{A}}(\alpha, \mathcal{T}), (\alpha' \notin S) \wedge (\nexists \alpha'' \in S, \alpha'' \equiv \alpha')$.

*The **reduction** of $\mathcal{T}$ w.r.t. S is* $\mathrm{Reduction}(\mathcal{T}, S) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top \rangle$ *where:*

- $\mathcal{A}' = S \cup \{\top\}$;
- $\mathcal{R}' = \{r \in \mathcal{R} \mid \exists \alpha \in S, r \in desc^{\mathcal{R}}(\alpha, \mathcal{T})\}$
  $\cup \{r \in \mathcal{R} \mid \exists (\alpha, \alpha') \in S \times \mathcal{A}, (\alpha \equiv \alpha') \wedge r \in desc^{\mathcal{R}}(\alpha', \mathcal{T})\}$;
- $Sp' = \{(\top, \alpha) \mid \alpha \in S\}$;
- $Ex' = \{(\alpha, r) \mid \alpha \in S, r \in desc^{\mathcal{R}}(\alpha, \mathcal{T})\}$
  $\cup \{(\alpha, r) \mid \exists (\alpha, \alpha') \in S \times \mathcal{A}, (\alpha \equiv \alpha') \wedge r \in desc^{\mathcal{R}}(\alpha', \mathcal{T})\}$;
- $\top$ *is the same root as in* $\mathcal{T}$.

The implemented TSoR can easily be reduced without changing the implementations. To reduce a TSoR-based measure, it is important to think about the weights. Calculating them in a way that preserves the importance of each element is not trivial, and there is no universal formula to do this: it depends on the aggregation function. Therefore, we propose to leave the definition of the weights to the user.

Definition 18 (Reduction of an implemented TSoR and a TSoR-based measure). *Let $\mathcal{T}^I = \langle \mathcal{T}, i \rangle$ be an implemented TSoR of $\mathcal{T}$. The **reduction** of $\mathcal{T}^I$ with respect to S is* $\mathrm{Reduction}(\mathcal{T}^I, S) = \langle \mathrm{Reduction}(\mathcal{T}, S), i|_{\mathcal{R}'} \rangle$.

Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ be a TSoR-based measure of $\mathcal{T}$. The reduction of $\mathcal{T}_M$ w.r.t. S is $\mathrm{Reduction}(\mathcal{T}_M, S) = \langle \mathrm{Reduction}(\mathcal{T}, S), \diamondsuit, \delta' \rangle$*, where:*

- *δ' depends on the aggregation function. The simplest way of defining it is to associate a weight of 1 to each analysis elements and to each requirement, and if necessary, redefine them by hand afterwards.*

Note that the result of a reduction may not satisfy Recommendation 1, which states that two sibling nodes should not be equivalent. Indeed, two equivalent requirements under two different but equivalent analysis element may be under the same analysis element and become siblings after a reduction.

Example 13. Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be the TSoR shown on the left side of Figure 4.10. $\mathrm{Reduction}(\mathcal{T}_M, S)$, where $S = \{\mathrm{A}, \mathrm{D}\}$, is given on the right side of the same figure. First, A and D satisfy the condition on S, since we assume that there is no equivalence in this TSoR, and since one is not a descendant of the other. Then, all the requirements below A have been kept and attached to it, i.e. y, its child, but also v, w, and x, the requirements of its children. Finally, D is directly connected to $\top$ with its requirements. Notice that k, which is the requirement expected by C, does not appear in the resulting TSoR, since none of its ancestors is in S.

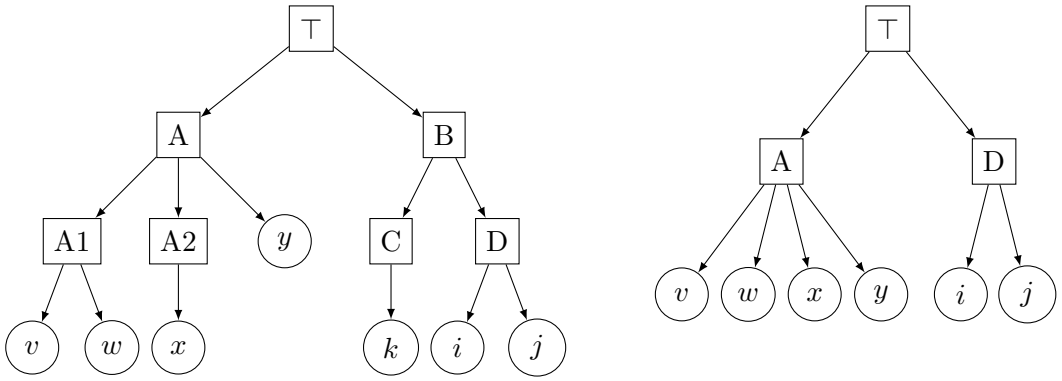


Figure 4.10: Simple example of a TSoR (left) and its reduction w.r.t. $\{\mathrm{A}, \mathrm{D}\}$ (right)

Property 3. *The reduction of a TSoR is a TSoR. Similarly, the reduction of an implemented TSoR is also an implemented TSoR. And the reduction of a TSoR-based measure is a TSoR-based measure.*

Proof. First, $\mathcal{A}'$ and $\mathcal{R}'$ are indeed sets of analysis elements and requirements respectively. The relation are also correctly defined, since $Sp' \subseteq \{\top\} \times \mathcal{A}' \subseteq \mathcal{A}' \times \mathcal{A}'$ and $Ex' \subseteq S \times \mathcal{R}' \subseteq \mathcal{A}' \times \mathcal{R}'$. Next, the conditions on S can be reformulated as follows: each requirement in $\mathcal{R}'$ has a unique

ancestor in $S \cup \{\alpha \in \mathcal{A} \mid \exists s \in S, \alpha \equiv s\}$. It follows that no cycle have been created in the graph $\langle \mathcal{A}' \cup \mathcal{R}', Sp' \cup Ex', \top \rangle$ which clearly is also connected. Hence, it is a directed rooted tree.

Without difficulty, the reduction of an implemented TSoR remains an implemented TSoR, and a TSoR-based measure remains such an element. □

To illustrate the interest of reduction, consider the KGAcc hierarchy (see Chapter 3), which contains many equivalent analysis elements. It is possible to focus only on the life cycle steps, or on the question types, without the other dimension. Let us first define the KGAcc TSoR and then two reductions of this TSoR.

Example 14 (KGAcc and its restrictions). Let $\mathcal{T}_{KGA} = \langle \mathcal{A}_{KGA}, \mathcal{R}_{KGA}, Sp_{KGA}, Ex_{KGA}, \top_{KGA} \rangle$ be the TSoR describing KGAcc, as shown in Figure 3.4, page 41. This TSoR is shown in Figure 4.11, where the analysis elements are referred to by their content for the sake of readability. More formally, it is defined as follows:

- $\mathcal{A}_{KGA} = \{\top_{KGA}, DC, DM, DU, DC\_Who, DC\_When, DC\_Where, DC\_How, DM\_Who, DM\_When, DM\_Where, DM\_How, DU\_Who, DU\_When, DU\_Where, DU\_How, DU\_What\}$;
- $\mathcal{R}_{KGA} = \{\mathrm{Q1}, \mathrm{Q2}, \mathrm{Q3}, \mathrm{Q4}, \mathrm{Q5}, \mathrm{Q6}, ...\}$ is the set of all KGAcc questions ;
- $Sp_{KGA} = \{(\top_{KGA}, DC), (\top_{KGA}, DM), (\top_{KGA}, DU), (DC, DC\_Who), (DC, DC\_When), (DC, DC\_Where), (DC, DC\_How), (DM, DM\_Who), (DM, DM\_When), (DM, DM\_Where), (DM, DM\_How), (DU, DU\_Who), (DU, DU\_When), (DU, DU\_Where), (DU, DU\_How), (DU, DU\_What)\}$;
- $Ex_{KGA} = \{(DC\_Who, \mathrm{Q1}), (DC\_When, \mathrm{Q2}), ...\}$ connects all the questions to their analysis elements.

The contents of analysis elements, given in the figure, are defined by: $\mathrm{content}(\top_{KGA}) =$ "Accountability", $\mathrm{content}(DC) =$ "Data Collection", $\mathrm{content}(DM) =$ "Data Maintenance", and $\mathrm{content}(DU) =$ "Data Usage". Then the content of the other nodes is quite transparent: $\mathrm{content}(DC\_Who) = \mathrm{content}(DM\_Who) = \mathrm{content}(DU\_Who) =$ "Who", $\mathrm{content}(DC\_When) =$ "When" and so on. Therefore, all analysis elements representing the same question type are equivalent.

Notice that this TSoR $\mathcal{T}_{KGA}$ follows all of the previous recommendations for the structure. Each analysis element is a parent node (Recommendation 2) of either an analysis element or a requirement, but not both (Recommendation 4). As a consequence, it also satisfies Recommendation 3, since no requirement has an equivalent one higher in the hierarchy.

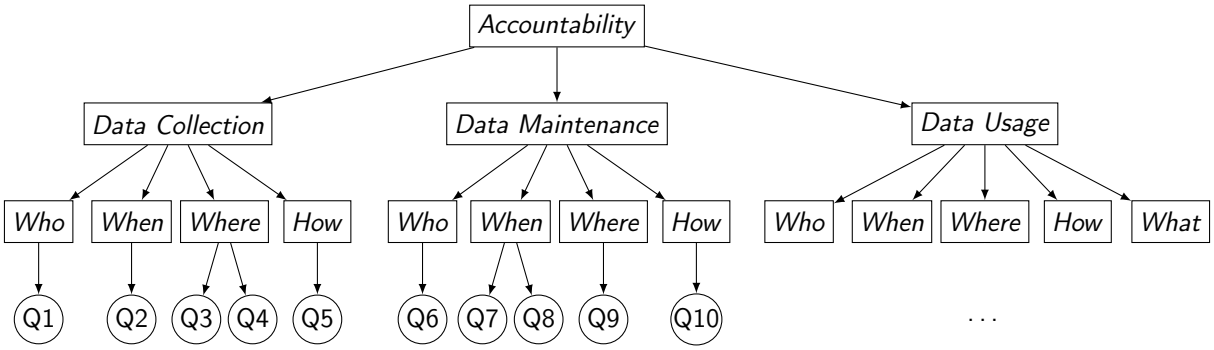


Figure 4.11: TSoR representing KGAcc framework

Let $S_{lifecycle} = \{DC, DM, DU\}$. Then, the reduction of $\mathcal{T}_{KGA}$ w.r.t. $S_{lifecycle}$, which we denote $\mathcal{T}_{KGA\_LC} = \text{Reduction}(\mathcal{T}_{KGA}, S_{lifecycle})$, is illustrated by Figure 4.12. The result is the same as if the second level of question types had been ignored.

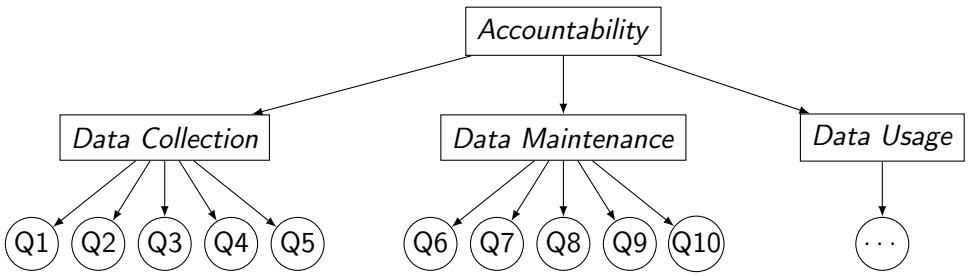


Figure 4.12: Reduction of the KGAcc TSoR with respect to the analysis elements representing life cycle steps

Let $S_{question} = \{DC\_Who, DC\_When, DC\_Where, DC\_How, DU\_What\}$. Then, the reduction of $\mathcal{T}_{KGA}$ w.r.t. $S_{question}$, $\mathcal{T}_{KGA\_Q} = \text{Reduction}(\mathcal{T}_{KGA}, S_{question})$ is illustrated by Figure 4.13. All the requirements are attached to the analysis element in $S_{question}$ equivalent to their original parent in $\mathcal{T}_{KGA}$.

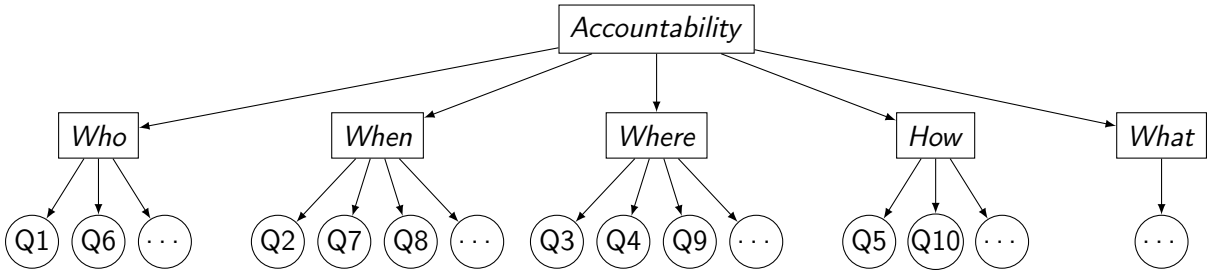


Figure 4.13: Reduction of the KGAcc TSoR with respect to the analysis elements representing question types

4.4.3 Extending an ITM with another

We consider the problem of adding a part of one TSoR to a given node (or several nodes) of another TSoR. Generally speaking, it can be useful during design to benefit from previous analysis works. For instance, one can consider adding the structural part from "generic" analysis structures, such as one based on the WH-questions, and then add requirements manually.

In a first step, we define copy operators, which add a given part of a TSoR at a single analysis element of another. In a second step, we define an extension operator, which copies a part of a TSoR at all leaves of another.

Copy of a TSoR into another

Let us denote $(\mathcal{T}_2, \alpha_2)$ the subtree rooted by an analysis element α_2 of a TSoR $\mathcal{T}_2$. Let α_1 be an analysis element of a TSoR $\mathcal{T}_1$. There are at least two ways to copy $(\mathcal{T}_2, \alpha_2)$ at α_1. A "full copy" attaches the entire subtree $(\mathcal{T}_2, \alpha_2)$ to node α_1, replacing α_2 with α_1 and preserving the whole hierarchy of analysis elements and requirements of $(\mathcal{T}_2, \alpha_2)$. We call this operator *copy with all requirements*, or simply $\mathrm{CopyAll}$. Since it is a rather classical operator, we provide its definition in Appendix A.5.

Performing a $\mathrm{CopyAll}$ means that all the added requirements are considered relevant in the context of $\mathcal{T}_1$ and α_1. However, that might not be the case, and one may want to keep only requirements that are explicitly relevant both in $(\mathcal{T}_2, \alpha_2)$ and under α_1. Hence, we define another operator $\mathrm{CopyFusion}$, which we call *copy with compatible requirements* or just *copy-fusion*, which relies on an important hypothesis. TSoRs $\mathcal{T}_1$ and $\mathcal{T}_2$ must be designed given a same set $\mathcal{R}$ of possible requirements, which can be used, or not, during the design. Therefore, a requirement $r \in \mathcal{R}$ that does not appear in $\mathcal{T}_1$ is a requirement that was analyzed in the context of that TSoR and was intentionally and explicitly not retained in that TSoR.

This operator acts only on the requirements *directly* under α_1 and structures them using $\mathcal{T}_2$. It leaves the other analysis elements (and their requirements) unchanged including those descending from α_1. Intuitively, at α_1, this operator keeps only the intersection of the requirements of $(\mathcal{T}_2, \alpha_2)$ and the children of α_1, this is why we call it a *copy-fusion*. The requirements are compared in terms of equivalence. The copy of the analysis structure remains the same as for the previous copy operator. Consequently, this copy-fusion is made to enrich the structure of a TSoR, not its requirements. For the definition of the copy-fusion for an implemented TSoR, the implementations are to be considered carefully. Since by hypothesis they are perfect, they should lead to the same result. In practice, that may not always be the case, so by default, we keep those from $(\mathcal{T}_2, \alpha_2)$ if the implementation is not empty. For a TSoR-based measure, a weight should be given to change uniformly the weights of $(\mathcal{T}_2, \alpha_2)$

compared to the existing children of α_1.

Definition 19 (Copy-fusion of a TSoR at some analysis elements of another). *Let $\mathcal{T}_1, \mathcal{T}_2$ be two TSoRs such that $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$. Let α_1 be an analysis elements of $\mathcal{A}_1$ and α_2 an analysis elements of $\mathcal{A}_2$ such that $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \cap \mathcal{A}_1 = \emptyset$.*

*The result of the **copy-fusion** of the subtree of $\mathcal{T}_2$ rooted by α_2 into the node α_1 $\mathcal{T}_1$ noted $\mathrm{CopyFusion}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top_1 \rangle$, is defined by:*

- $\mathcal{A}' = \mathcal{A}_1 \cup desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$
- *$\mathcal{R}'$ takes all the requirements of $\mathcal{R}_1$, except the children of α_1 which have no equivalent in the requirements to copy. Formally:* $\mathcal{R}' = \mathcal{R}_1 \setminus \{r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \mid \nexists r' \in desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2), r \equiv r'\}$
- $Sp' = Sp_1 \cup \left(Sp_2 \cap (desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2))^2\right) \cup \{(\alpha_1, \alpha) \mid \alpha \in children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)\}$
- *Ex' keeps the existing edges of Ex_1 that do not involve α_1, and it attaches all its children to the new analysis elements according to the elements of $\mathcal{T}_2$ that are copied. Formally:*

$$\begin{aligned} Ex' = & Ex_1 \setminus \{(\alpha_1, r) \mid r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1)\} \\ & \cup \{(\alpha_1, r) \mid r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \wedge \exists r' \in children^{\mathcal{R}}(\alpha_2, \mathcal{T}_2), r \equiv r'\} \\ & \cup \{(\alpha, r) \mid r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \wedge \alpha \in desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \wedge \exists r' \in children^{\mathcal{R}}(\alpha, \mathcal{T}_2), r \equiv r'\} \end{aligned}$$

*Let $\mathcal{T}^I{}_1 = \langle \mathcal{T}_1, i_1 \rangle, \mathcal{T}^I{}_2 = \langle \mathcal{T}_2, i_2 \rangle$ be two implemented TSoRs of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. The **copy-fusion** $\mathrm{CopyFusion}(\mathcal{T}^I{}_1, \alpha_1, (\mathcal{T}^I{}_2, \alpha_2))$ is $\langle \mathrm{CopyFusion}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)), i' \rangle$, where:*

$$\begin{aligned} i' : & \mathcal{R}' \to \mathit{IMP} \\ & r \mapsto \begin{cases} i_2(r) & \text{if } i_2(r) \neq \emptyset \\ i_1(r) & \text{otherwise} \end{cases} \end{aligned}$$

*Let $\mathcal{T}_{M1} = \langle \mathcal{T}_1, \diamondsuit, \delta_1 \rangle, \mathcal{T}_{M2} = \langle \mathcal{T}_2, \diamondsuit, \delta_2 \rangle$ be two TSoR-based measures of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, with the same aggregation function. Let $\omega \in \mathbb{R}^+$, be a multiplicative coefficient for changing the weights uniformly during the copy. Then the **copy-fusion** of a part of $\mathcal{T}_{M2}$ into $\mathcal{T}_{M1}$ is defined by $\mathrm{CopyFusion}(\mathcal{T}_{M1}, \alpha_1, (\mathcal{T}_{M2}, \alpha_2), \omega) = \langle \mathrm{CopyFusion}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)), \diamondsuit, \delta' \rangle$ with:*

- *Weight function is defined as follows:*

$$\delta' : Sp' \cup Ex' \to \mathbb{R}^+$$
$$(b, b') \mapsto \begin{cases} \delta_1(b, b') & \text{if } (b, b') \in Sp_1 \cup Ex_1 \\ \omega \times \delta_2(\alpha_2, b') & \text{if } b = \alpha_1 \text{ and } b' \in children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \\ \omega \times \delta_2(b, b') & \text{if } (b, b') \in Sp_2 \cup Ex_2 \end{cases}$$

We illustrate this copy with two examples, a first synthetic one to fully understand the implications of this operator, and a concrete example based on KGAcc.

Example 15. Let $\mathcal{T}_{M1}, \mathcal{T}_{M2}$ be two TSoR-based measures illustrated in Figure 4.14. $\mathcal{T}_{M2}$ is copied from its analysis element C into the element A of $\mathcal{T}_{M1}$. In addition, we want the children of C to be four times less important than $\mathrm{A1}$, so we apply a coefficient $1/4$. Therefore, $\mathrm{CopyFusion}(\mathcal{T}_{M1}, \mathrm{A}, (\mathcal{T}_{M2}, \mathrm{C}), 1/4)$ is shown at the bottom of the same figure.

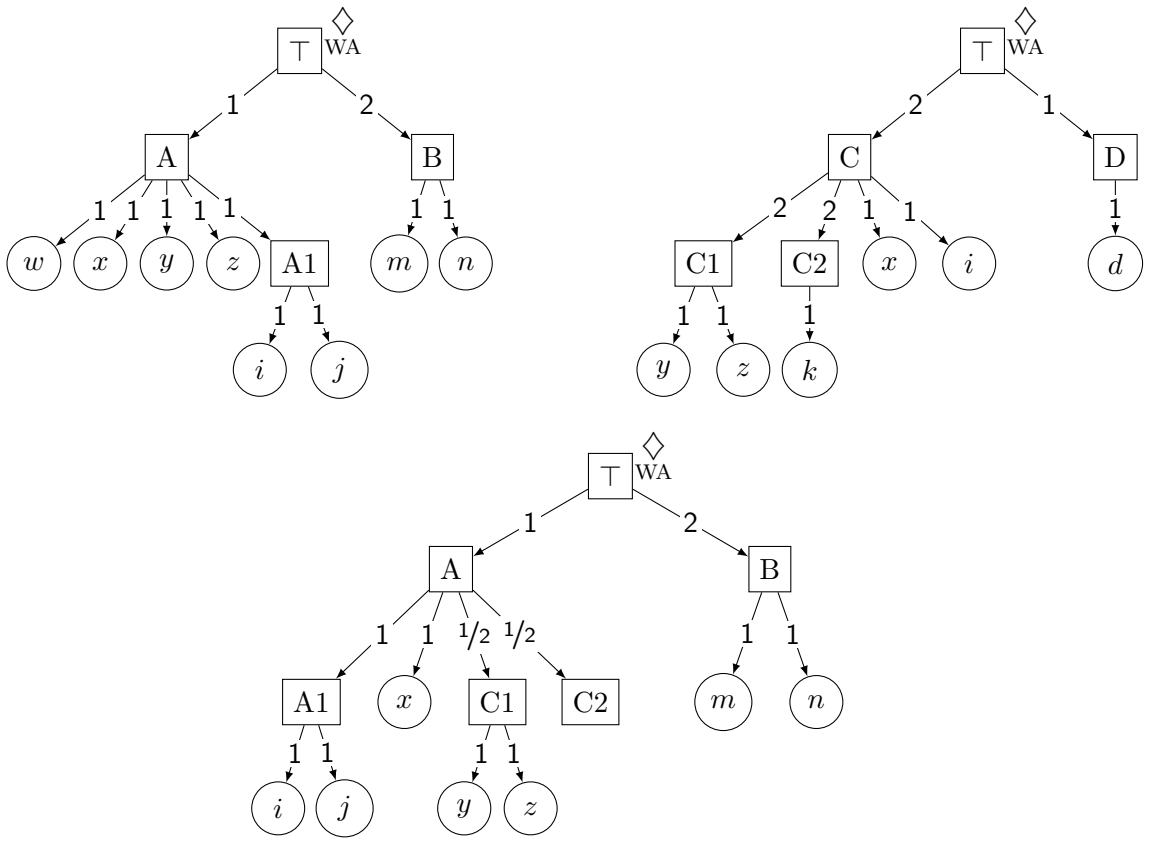


Figure 4.14: Two TSoR-based measures and the copy-fusion of one into the other

We observe several elements in this example. First, only children of A that are requirements can change. Hence, $\mathrm{A1}$, which specifies A, keeps its requirements i and j, even though i is the only one that appears in $\mathcal{T}_2$. Second, among the children of A, only the requirements that

appear in $(\mathcal{T}_{M2}, \mathrm{C})$ are kept, so w disappears in the result of the copy-fusion. Third, since k is in this subtree but not in the children of A, it is by hypothesis unsuitable in the context of A, and it does not appear in the result either, leaving $\mathrm{C2}$ as a dead-end analysis element. Finally, weights of $\mathrm{A1}$ and x are the same as in $\mathcal{T}_1$ while the new analysis elements to be linked to A have their original weights multiplied by the coefficient $1/4$.

As shown in the previous example, it is important to note that the result of this copy-fusion may lead to a TSoR that does not satisfy Recommendation 2 about no dead-end analysis elements. This is the case even if the two TSoRs initially comply with this recommendation.

Property 4. *The copy-fusion of a TSoR (an implemented TSoR, a TSoR-based measure) into another is a TSoR (respectively, an implemented TSoR, a TSoR-based measure).*

Proof. Let $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$ be two TSoRs. Let $\alpha_1 \in \mathcal{A}_1$ and $\alpha_2 \in \mathcal{A}_2$ such that $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \cap \mathcal{A}_1 = \emptyset$. Let us show that the result of $\mathrm{CopyFusion}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top' \rangle$ is a TSoR.

- Similar to the $\mathrm{CopyAll}$ operator, $\mathcal{A}'$ is a set of analysis elements and Sp' is a set of relations between elements of $\mathcal{A}'$.
- $\mathcal{R}' \subseteq \mathcal{R}_1$ which is a set of requirements, therefore $\mathcal{R}'$ is a set of requirements.
- Let us show that Ex' is properly defined on $\mathcal{A}' \times \mathcal{R}'$.
  Let $(\alpha, r) \in Ex_1 \setminus \{(\alpha_1, r) \mid r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1)\}$. Thus, $\alpha \in \mathcal{A}_1 \subseteq \mathcal{A}'$, and $r \notin children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1)$, therefore, $r \in \mathcal{R}'$.
  Let $(\alpha, r) \in \{(\alpha_1, r) \mid r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \wedge \exists r' \in children^{\mathcal{R}}(\alpha_2, \mathcal{T}_2), r \equiv r'\}$. So, $\alpha_1 \in \mathcal{A}_1 \subseteq \mathcal{A}'$. And since $r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \subseteq \mathcal{R}_1$ and $r' \in children^{\mathcal{R}}(\alpha_2, \mathcal{T}_2) \subseteq desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)$, then $r \in \mathcal{R}'$.
  Let $(\alpha, r) \in \{(\alpha, r) \mid r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \wedge \alpha \in desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \wedge \exists r' \in children^{\mathcal{R}}(\alpha, \mathcal{T}_2), r \equiv r'\}$. So, $\alpha \in desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \subseteq \mathcal{A}'$. Since $r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1) \subseteq \mathcal{R}_1$ and $r' \in children^{\mathcal{R}}(\alpha, \mathcal{T}_2) \subseteq desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)$. Therefore, r belongs to $\mathcal{R}'$.
  Hence, Ex' is a set of relations between elements of $\mathcal{A}'$ and $\mathcal{R}'$.
- $\top_1 \in \mathcal{A}_1 \subseteq \mathcal{A}'$. So the root is one of the analysis elements.

Finally, let us show that $\mathcal{G} = \langle \mathcal{A}' \cup \mathcal{R}', Sp' \cup Ex', \top' \rangle$ is a directed rooted tree. The analysis structure $\langle \mathcal{A}', Sp', \top' \rangle$ is the same as for the $\mathrm{CopyAll}$ operator. Therefore, as shown in the proof of the $\mathrm{CopyAll}$ operator, it is a directed rooted tree.

So, let us prove that each requirement of $\mathcal{G}$ has exactly one parent node. Let $r \in \mathcal{R}'$. Suppose that $r \notin children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1)$. Since $\mathcal{T}_1$ is a TSoR, r has a unique parent node α according to Ex_1. Since $r \notin children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1)$, $(\alpha, r) \in Ex'$, and r has no other parent node by the definition of Ex'.

Then, consider the other case where $r \in children^{\mathcal{R}}(\alpha_1, \mathcal{T}_1)$. By definition of $\mathcal{R}'$, $\exists r' \in desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)$ such that $r' \equiv r$. Since $\mathcal{T}_2$ is a TSoR, r' has only one parent node in $\mathcal{T}_2$, either α_2 or a descendant of α_2, but not both simultaneously. So, according to Ex', r has one, and only one parent, either α_1 or that descendant. Therefore, all the requirements of $\mathcal{G}$ have exactly one parent node. Hence, $\mathcal{G}$ is a directed rooted tree.

Without difficulty, the copy-fusion of an implemented TSoR into another remains an implemented TSoR. Finally, the weight is well defined for all elements in $Sp' \cup Ex'$. Therefore the copy-fusion of a TSoR-based measure into another remains a TSoR-based measure. □

In Example 14 (see page 91), we defined two TSoRs related to the KGAcc framework: $\mathcal{T}_{KGA\_LC}$ and $\mathcal{T}_{KGA\_Q}$ which share the same requirements. Let us show how we can merge them together by copying one into the other.

Example 16 (Copy-fusion of a reduction of KGAcc into another). Let $\mathcal{T}_{KGA\_LC}$ and $\mathcal{T}_{KGA\_Q}$ be the two TSoRs defined in Example 14. We define the copy-fusion of the entire TSoR $\mathcal{T}_{KGA\_Q}$ (i.e., starting from its root) into the node DC of $\mathcal{T}_{KGA\_LC}$. The common requirements between the children of DC in $\mathcal{T}_{KGA\_LC}$ and the requirements of $\mathcal{T}_{KGA\_Q}$ are Q1, Q2, Q3, Q4 and Q5. These are copied. The resulting TSoR, $\mathrm{CopyFusion}(\mathcal{T}_{KGA\_LC}, DC, (\mathcal{T}_{KGA\_Q}, \top_{KGA\_Q}))$, is given in Figure 4.15. We also observe that a new analysis element appears in the structure, the one with the content "What". It is a dead-end, so it could be removed with a cleaning operator.

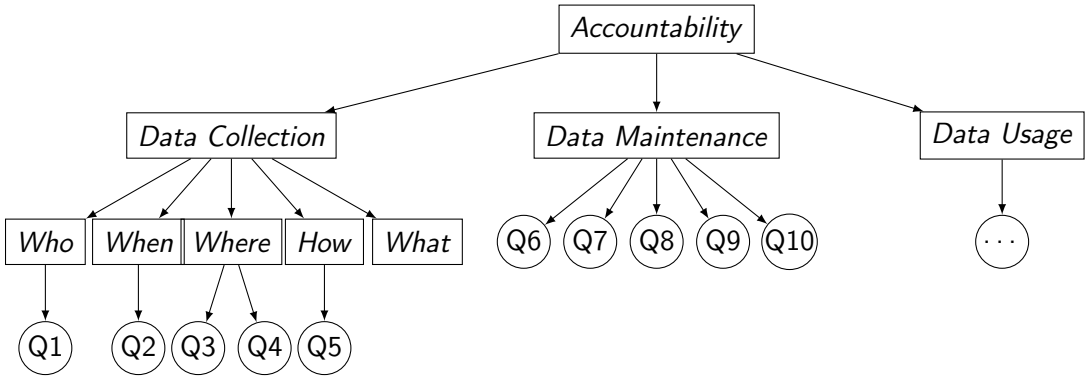


Figure 4.15: Copy-fusion of $\mathcal{T}_{KGA\_Q}$ into the node DC of $\mathcal{T}_{KGA\_LC}$

In this example, we can see how this operator can be used. However, it would be interesting to apply it to all leaf nodes of the analysis structure. This would allow to obtain the KGAcc hierarchy.

Extension of a TSoR with another

Following this previous idea, copying a part of a TSoR into all leaves of another TSoR enables a systematic approach by combining different analysis dimensions. For instance, the LiQuID

hierarchy [21] (see page 37) can be obtained by starting with an analysis dimension of the life cycle steps, by copying an analysis dimension of the WH-questions on all leaves of the first dimension, and finally doing the same with a last analysis dimension of what is called the information level. Extending a TSoR on all leaves can also be used in addition to the reduction operator to rearrange a TSoR differently.

Therefore, it is possible to extend a TSoR with another, or a part of another, by using a copy operator on all leaves of the analysis structure (i.e., all analysis elements that are not specified by any other). However, technically the constraints for using the copy operators require different analysis elements (and requirements for $\mathrm{CopyAll}$) between the TSoRs. These constraints would not be satisfied after the first copy. To get around them, we can duplicate a TSoR by replacing its nodes with new equivalent ones with exactly the same content. We will not define this $\mathrm{Duplicate}$ operator here, but it allows us to duplicate a TSoR, each time with different but equivalent analysis elements and requirements. Note that this strategy can also be used to bypass $\mathrm{CopyAll}$ constraints (see Appendix A.5).

As for the copy two extension operators can be defined. The first applies to TSoRs with different requirements and copies all of them, while the second takes TSoRs with requirements in common, and copies only those that are present in both TSoRs. For both, the principle is the same and consists in systematically copying a part of one TSoR into all leaves of the other TSoR to extend it. Therefore, we define only the second extend operator, based on $\mathrm{CopyFusion}$, which keeps only compatible requirements, but it can be adapted without any difficulty, by changing the copy operator. As for the copy-fusion, this operator is designed to be applied to TSoRs that have been built considering the same set of requirements. Therefore, it avoids placing requirements in contexts where they have not been retained during the conception of the first TSoR.

Definition 20 (Extension of a TSoR with another). *Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ and $\mathcal{T}' = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top' \rangle$ be two TSoRs. Let α' be an analysis element of $\mathcal{A}'$, the result of the* ***extension*** *of $\mathcal{T}$ with a subtree of $\mathcal{T}'$ is denoted $\mathrm{Extend}(\mathcal{T}, (\mathcal{T}', \alpha'))$.*

Let $\{\alpha_1, ..., \alpha_n\}$, with $n \geq 1$, be the analysis elements of $\mathcal{T}$ that are leaves of the analysis structure, i.e. that are not specified by any analysis element. Then $\mathrm{Extend}(\mathcal{T}, (\mathcal{T}', \alpha')) = \mathcal{T}_n$, where $\mathcal{T}_n$ is defined recursively by:

- $\mathcal{T}_0 = \mathcal{T}$
- $\forall i \in \{1, ..., n\}, \mathcal{T}_i = \mathrm{CopyFusion}(\mathcal{T}_{i-1}, \alpha_i, (\mathrm{Duplicate}(\mathcal{T}'), \alpha'))$

Let $\mathcal{T}^I = \langle \mathcal{T}, i \rangle, \mathcal{T}^{I'} = \langle \mathcal{T}', i' \rangle$ be two implemented TSoRs of $\mathcal{T}$ and $\mathcal{T}'$, respectively. The $\mathrm{Extend}(\mathcal{T}^I, (\mathcal{T}^{I'}, \alpha')) = {\mathcal{T}^I}_n$, where ${\mathcal{T}^I}_n$ is defined in the same way by recursion:

- ${\mathcal{T}^I}_0 = \mathcal{T}^I$

- $\forall i \in \{1, ..., n\}, \mathcal{T}^I{}_i = \mathrm{CopyFusion}(\mathcal{T}^I{}_{i-1}, \alpha_i, (\mathrm{Duplicate}(\mathcal{T}^{I'}), \alpha'))$

Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle, \mathcal{T}_M{}' = \langle \mathcal{T}', \diamondsuit, \delta' \rangle$ be two TSoR-based measures of $\mathcal{T}$ and $\mathcal{T}'$, with the same aggregation function. Let $\omega \in \mathbb{R}^{+*}$. The $\mathrm{Extend}(\mathcal{T}_M, (\mathcal{T}_M{}', \alpha'), \omega) = \mathcal{T}_{Mn}$, where:

- $\mathcal{T}_{M0} = \mathcal{T}_M$
- $\forall i \in \{1, ..., n\}, \mathcal{T}_{Mi} = \mathrm{CopyFusion}(\mathcal{T}_{Mi-1}, \alpha_i, (\mathrm{Duplicate}(\mathcal{T}_M{}'), \alpha'), \omega)$

Notice that in practice it is not necessary to duplicate the entire TSoR, but only the part that is being copied.

Property 5. *The extension of a TSoR (an implemented TSoR, a TSoR-based measure) by another is a TSoR (respectively, an implemented TSoR, a TSoR-based measure).*

Proof. Provided that the duplicate operator creates new analysis elements and new requirements each time it is used, then the constraints specified for the use of the copy operator are always satisfied. Since the copy-fusion of a TSoR is a TSoR, the extension of a TSoR is also a TSoR. The same reasoning applies to an implemented TSoR and a TSoR-based measure. □

In Example 16 (see page 97), we started to rebuild the KGAcc framework based on its reduction over the different aspects: $\mathcal{T}_{KGA\_LC}$ and $\mathcal{T}_{KGA\_Q}$, defined in Example 14 (see page 91). Let us show how to completely reconstruct the KGAcc framework with the extension.

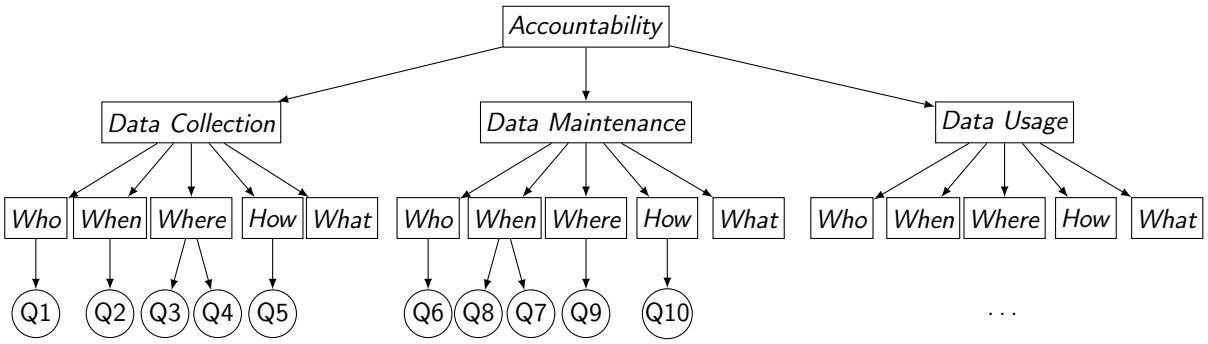


Figure 4.16: Extension of $\mathcal{T}_{KGA\_LC}$ with $\mathcal{T}_{KGA\_Q}$

Example 17 (Extension of a reduction of KGAcc with another). Let $\mathcal{T}_{KGA\_LC}$ and $\mathcal{T}_{KGA\_Q}$ be the two TSoRs defined in Example 14, page 91 and appearing in Figure 4.17. We define the extension of the TSoR $\mathcal{T}_{KGA\_LC}$ with the whole TSoR $\mathcal{T}_{KGA\_Q}$ (i.e., starting from its root). The first step of the extension is illustrated in Example 16, by the copy of $\mathcal{T}_{KGA\_Q}$ into $\mathcal{T}_{KGA\_LC}$, except that in the context of Extend, $\mathcal{T}_{KGA\_Q}$ has first been duplicated, keeping the same contents. Therefore, extending $\mathcal{T}_{KGA\_LC}$ consists in continuing to copy (a duplicate of) $\mathcal{T}_{KGA\_Q}$ on the other leaf nodes. The resulting TSoR, $\mathrm{Extend}(\mathcal{T}_{KGA\_LC}, (\mathcal{T}_{KGA\_Q}, \top_{KGA\_Q}))$, is given in Figure 4.16.

To recover the original KGAcc framework $\mathcal{T}_{KGA}$, it is necessary to remove the dead-end analysis elements by using the cleaning operator $\mathrm{Simplify}$. Hence, we obtain $\mathcal{T}_{KGA} = \mathrm{Simplify}(\mathrm{Extend}(\mathcal{T}_{KGA\_LC}, (\mathcal{T}_{KGA\_Q}, \top_{KGA\_Q})))$. This cycle is illustrated in Figure 4.17.

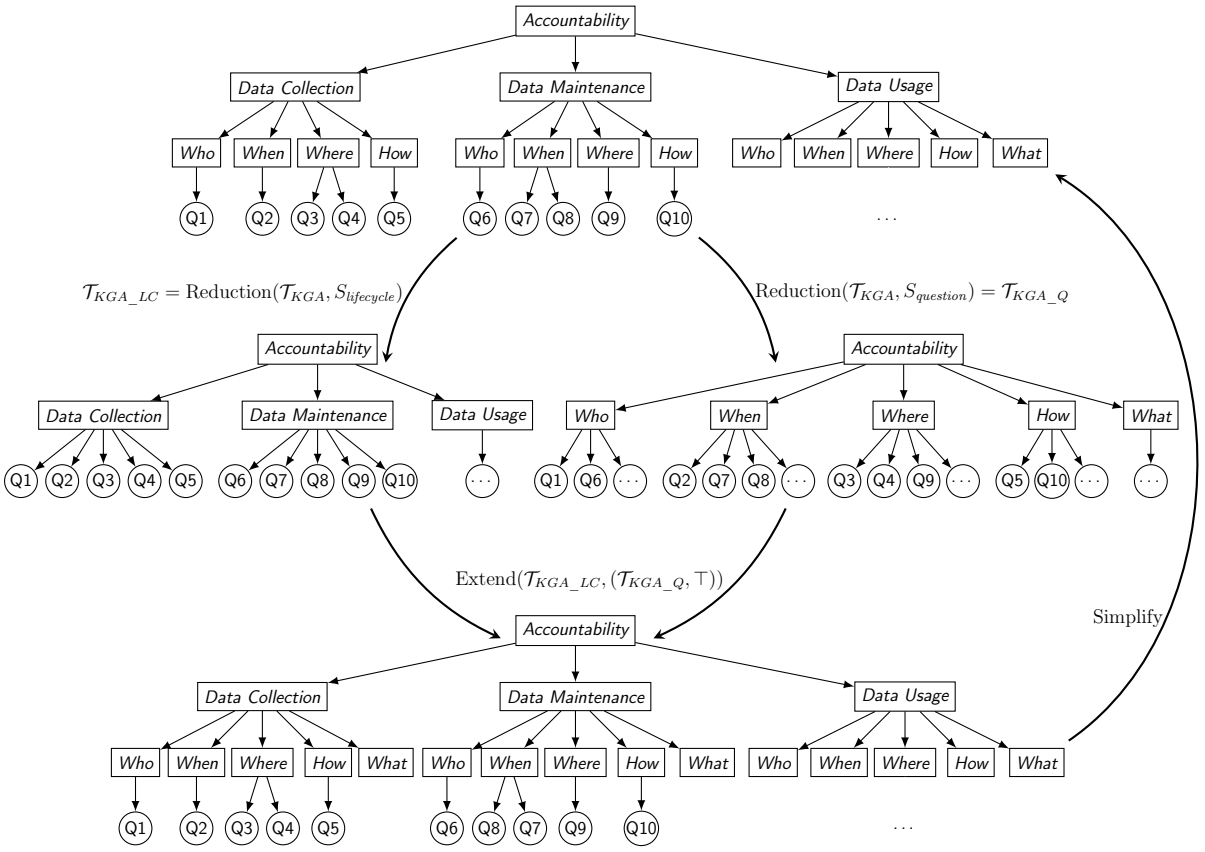


Figure 4.17: Operators on KGAcc hierarchy

4.4.4 Conclusion

We have defined a set of operators that can be used to create, modify, reuse, and complete TSoRs. Based on the results obtained, they can be slightly modified in terms of structure or weights. The operators open many possibilities. For instance, they enable the extraction of a chosen part of an existing measure for reuse in another, possibly more specific, context. They can also be used to reduce the number of dimensions studied in a TSoR in order to simplify it. Conversely, the number of dimensions can be increased by relying on generators, i.e., TSoRs with a single level of analysis element. Altogether, these operators simplify the design of a TSoR, especially if it is possible to rely on a library of existing TSoRs to extract, further develop or detail the analysis structure.

4.5 Comparison of TSoRs

In the previous sections, we described what a TSoR is and explained how to manipulate, modify, and combine TSoRs. We also showed that some usual compound measures can be expressed with a TSoR by illustrating it with a data quality measure or with KGAcc. The representation of measures in the same formalism also facilitates their comparison. So in this section, we study methods for comparing such measures, with the aim of identifying the common metrics, the common structuring elements, to quantify how much they share... Our goal is to provide a precise analysis of the measures, to highlight their most important aspects, and to guide users in choosing one measure over another.

There are several ways to compare measures. It can be an experimental comparison, which consists in studying the correlation between the results obtained by the evaluation of many digital resources according to the measures to be compared. Or it can be a theoretical comparison, which studies and compares each element that constitutes the measures. The problem with the experimental comparison is that it requires a large number of evaluations. Whereas once the measures are expressed in a common formalism, it becomes easier to compare them theoretically. Therefore, by using the TSoR to express the measures, we opt for this second option.

However, several elements can hinder this theoretical comparison. It may be extremely difficult to compare requirements expressed in natural language, since their interpretation and implementation may vary from person to person. A human expertise is required to compare two nodes that do not have exactly the same content. In addition, comparing two metrics based on their implementation may be very difficult and time-consuming due to differences in the languages used and the complexity of the programs. Therefore, we limit our comparison to the structure of the TSoRs, leaving the implementations out of the scope, and we only compare similar measures or measures built on the same basis, such as the FAIR measures. It is still possible to compare very different measures, but we warn that determining equivalence between nodes requires human expertise and may be subjective.

In this section, we first present a qualitative comparison that aims to identify the common and specific nodes between two TSoRs. Then, we propose a more quantitative comparison that highlights the importance of each element on the global score. We illustrate all the notions introduced with FAIR measures.

4.5.1 Identification of common and specific nodes

The easiest way to compare TSoRs is to identify the nodes they share and the nodes that are specific to one of the TSoRs. Then it is possible to determine the coverage rate of one TSoR by another.

Definition 21 (Common nodes between two TSoRs). *Let $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle, \mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$ be two TSoRs. The common analysis elements, in terms of equivalence, are denoted $\mathrm{common}^{\mathcal{A}}(\mathcal{T}_1, \mathcal{T}_2)$, and the common requirements are $\mathrm{common}^{\mathcal{R}}(\mathcal{T}_1, \mathcal{T}_2)$. They are defined as follows:*

- $\mathrm{common}^{\mathcal{A}}(\mathcal{T}_1, \mathcal{T}_2) = \{\alpha_1 \in \mathcal{A}_1 \mid \exists \alpha_2 \in \mathcal{A}_2, \alpha_1 \equiv \alpha_2\}$;
- $\mathrm{common}^{\mathcal{R}}(\mathcal{T}_1, \mathcal{T}_2) = \{r_1 \in \mathcal{R}_1 \mid \exists r_2 \in \mathcal{R}_2, r_1 \equiv r_2\}$.

Notice that in both cases, if there are equivalent nodes in $\mathcal{A}_1$, there may be equivalent nodes in the resulting set. However, we prefer to highlight each of the common nodes rather than keeping only one of the equivalent nodes for each equivalence.

Now let us define the nodes that are specific to one TSoR compared to another TSoR, i.e. the nodes that only appear in the first TSoR.

Definition 22 (Specific node of a TSoR compared to another). *Let be two TSoRs: $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle$, $\mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$. The set of nodes that are specific to $\mathcal{T}_1$ compared to $\mathcal{T}_2$ in terms of the equivalence is denoted $\mathrm{specific}^{\mathcal{A}}$ for analysis element and $\mathrm{specific}^{\mathcal{R}}$ for requirements. They are defined by:*

- $\mathrm{specific}^{\mathcal{A}}(\mathcal{T}_1, \mathcal{T}_2) = \{\alpha_1 \in \mathcal{A}_1 \mid \nexists \alpha_2 \in \mathcal{A}_2, \alpha_1 \equiv \alpha_2\}$;
- $\mathrm{specific}^{\mathcal{R}}(\mathcal{T}_1, \mathcal{T}_2) = \{r_1 \in \mathcal{R}_1 \mid \nexists r_2 \in \mathcal{R}_2, r_1 \equiv r_2\}$.

We illustrate these notions with an example of a FAIR-based measure. Let us first define the analysis structure given by the original FAIR principles and then the TSoR of an existing measure.

Example 18 (Common and specific nodes of the FAIR principles compared to FAIR-Checker). Let $\mathcal{T}_{FAIR} = \langle \mathcal{A}_{FAIR}, \emptyset, Sp_{FAIR}, \emptyset, \mathsf{FAIR} \rangle$ be the TSoR illustrated in Figure 4.18. This TSoR represents the FAIR principles. Since all these principles are stated in very general terms and are always reformulated and refined to define a measure, we consider all principles and sub-principles to be analysis elements. Therefore, it is reduced to an analysis dimension (i.e., TSoR without requirements).

Let $\mathcal{T}_{check} = \langle \mathcal{A}_{check}, \mathcal{R}_{check}, Sp_{check}, Ex_{check}, \mathsf{FAIR} \rangle$ be the TSoR representing the FAIR-Checker measure [15] and shown in Figure 4.19 (weights will be discussed later). It has new requirements with respect to the original FAIR principles and fewer analysis elements.

In the case of FAIR and FAIR-Checker, it is easy to determine their common and specific nodes because they rely on the same set of analysis elements. Therefore, the common analysis elements are $\mathrm{common}^{\mathcal{A}}(\mathcal{T}_{FAIR}, \mathcal{T}_{check}) = \{\mathsf{FAIR}, \mathsf{Findable}, \mathsf{Accessible}, \mathsf{Interoperable}, \mathsf{Reusable}, \mathsf{F1}, \mathsf{F2}, \mathsf{A1}, \mathsf{A1.1}, \mathsf{A1.2}, \mathsf{I1}, \mathsf{I2}, \mathsf{I3}, \mathsf{R1}, \mathsf{R1.1}, \mathsf{R1.2}, \mathsf{R1.3}\}$. Then the analysis elements that are specific to FAIR compared to FAIR-Checker are $\mathrm{specific}^{\mathcal{A}}(\mathcal{T}_{FAIR}, \mathcal{T}_{check}) = \{\mathsf{F3}, \mathsf{F4}, \mathsf{A2}\}$.

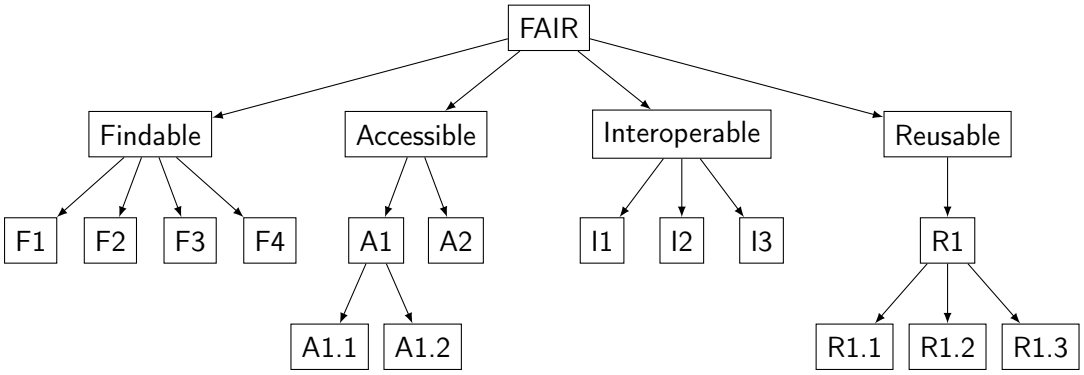


Figure 4.18: Analysis dimension of the FAIR principles

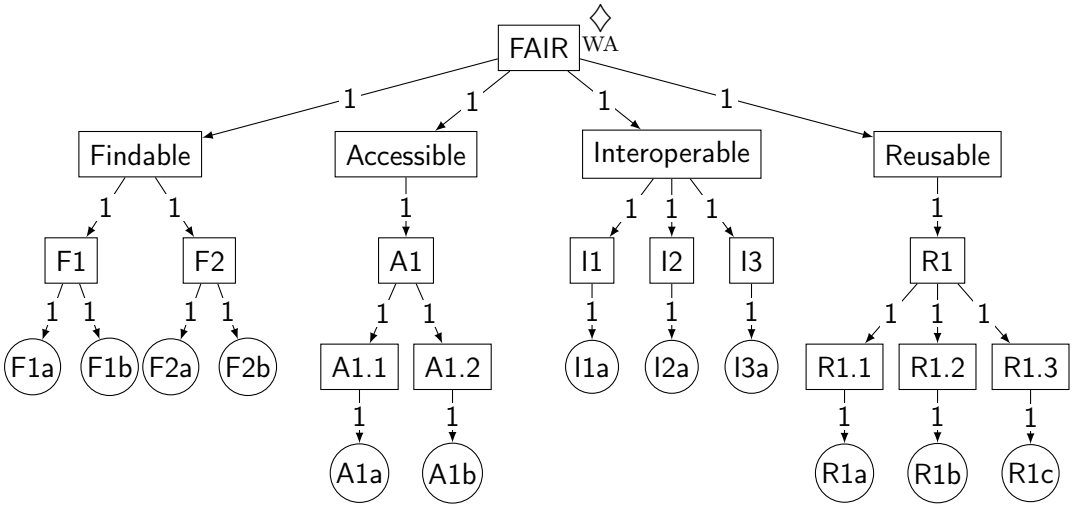


Figure 4.19: TSoR-based measure of FAIR-Checker

Therefore, FAIR-Checker does not consider all aspects of the FAIR principles. Conversely, $\text{specific}^{\mathcal{A}}(\mathcal{T}_{check}, \mathcal{T}_{FAIR}) = \emptyset$: FAIR-Checker is fully covered by FAIR, it does not bring any additional analysis element. However, $\text{specific}^{\mathcal{R}}(\mathcal{T}_{check}, \mathcal{T}_{FAIR}) = \mathcal{R}_{check}$, which means that all the requirements in FAIR-Checker are new.

To quantify how much they share, we can define the coverage rate of one TSoR by another, either in terms of analysis elements or of requirements. Let us focus on the former.

Definition 23 (Coverage rate of a TSoR with respect to analysis elements)**.** *Let $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle$, $\mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$ be two TSoRs. The coverage rate of the analysis elements of $\mathcal{T}_1$ by those of $\mathcal{T}_2$ is the part of the analysis elements of $\mathcal{A}_1$ that have an equivalent in $\mathcal{A}_2$. Formally it is defined by:*

$$\text{coverage}^{\mathcal{A}}(\mathcal{T}_1, \mathcal{T}_2) = \frac{|\,\text{common}^{\mathcal{A}}(\mathcal{T}_1, \mathcal{T}_2)|}{|\mathcal{A}_1|}.$$

Example 19 (Coverage rate of FAIR by FAIR-Checker). The coverage rate of FAIR by FAIR-Checker is $\text{coverage}^{\mathcal{A}}(\mathcal{T}_{FAIR}, \mathcal{T}_{check}) = \frac{|\text{common}^{\mathcal{A}}(\mathcal{T}_{FAIR}, \mathcal{T}_{check})|}{|\mathcal{A}_1|} = \frac{17}{20} = 85\%$.

4.5.2 Quantitative comparison based on the impact

We can compare TSoRs in a more quantitative way. Using the information provided by the TSoR-based measures, we can compare the impact of their nodes and identify which TSoR gives the most importance to which elements. As a reminder, the impact of a node, introduced in Definition 13, page 82, refers to its importance in the definition of the whole measure, i.e. the score obtained if all requirements descending from it are evaluated as 1 and all others are evaluated as 0.

To illustrate how to use the impact to compare two TSoR-based measures, let us first define the TSoR-based measures of two existing FAIR measures: FAIR-Checker [15] and O'FAIRe [16], and then compare the impact of their nodes.

Example 20 (Comparison of the impact of analysis elements over FAIR-Checker and O'FAIRe). Let $\mathcal{T}_{check}$ be the TSoR representing the FAIR-Checker measure defined in Example 18, and illustrated in Figure 4.19. FAIR-Checker do not mention any way to compute a global score, so we arbitrarily choose a weighted average and weights of 1 on each edge. Let $\mathcal{T}_{M\,check} = \langle \mathcal{T}_{check}, \underset{\text{WA}}{\diamondsuit}, \delta_{check} \rangle$ be this TSoR-based measure of FAIR-Checker.

Let $\mathcal{T}_{oFe}$ be the TSoR representing the O'FAIRe measure [16], with new requirements for each of the original FAIR principles. O'FAIRe defines weights for each requirement based on existing studies. It also computes the global "$FAIRScore$" as the weighted sum. Therefore, the TSoR-based measure of O'FAIRe is $\mathcal{T}_{M\,oFe} = \langle \mathcal{T}_{oFe}, \underset{\text{WS}}{\diamondsuit}, \delta_{oFe} \rangle$. It is illustrated in Figure 4.20, where all the requirements under the same node are displayed in the same ellipse to keep it readable. The weights indicated on their edge to the node is the sum of the weight of each requirement.

Now that these two TSoR-based measures are defined, we can compare the impact of each of their nodes. They both rely on the same set of analysis elements, so it is easy to compare them. However, we do not compare their requirements because they are both expressed in natural language and they are different.

Let $\mathcal{T}_{M\,oFe}$ be the TSoR-based measure representing O'FAIRe and $\mathcal{T}_{M\,check}$ representing FAIR-Checker. For each of the analysis elements of $\mathcal{A}_{FAIR}$, we compute its impact on the two TSoRs according to Definition 13. For O'FAIRe, we compute the normalized impact, i.e. the impact divided by the maximum achievable score. The analysis elements that are not in $\mathcal{T}_{M\,check}$ give an impact of 0.

The impacts are listed in Table 4.1. Since FAIR-Checker considers fewer requirements

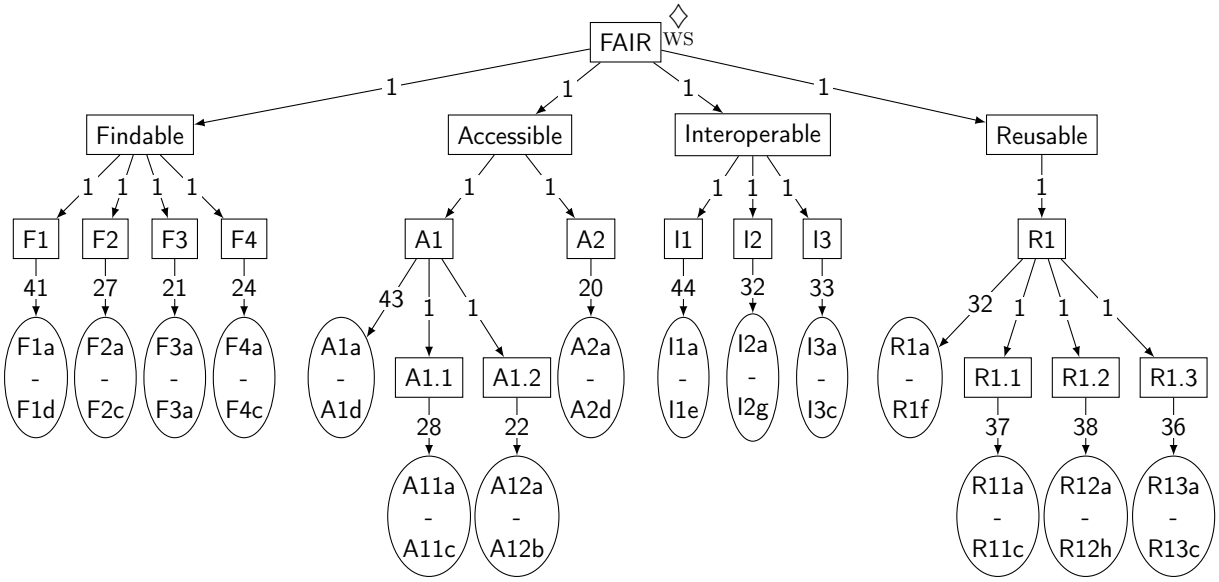


Figure 4.20: TSoR-based measure of O'FAIRe

than O'FAIRe, most of its analysis elements have a greater impact. We observe that on the second level of the hierarchy (F1, F2, etc.) of O'FAIRe, two analysis elements are much more important than the others. They correspond to the two sub-principles that have both requirements and sub-analysis elements: A1 and R1. Indeed, with the weighted sum, their impact is the cumulative impact of all the elements under them.

The first definitions for the comparison allow to identify differences in the structuring elements and in the requirements. Then, the comparison of the impacts focuses more on the weighting elements and on how the score is computed. It also highlights which elements are the most important. These provide some keys for comparing TSoRs. However, they may not be sufficient and one may want to dig deeper into the implementations to compare not only the theoretical objectives but also their application. This could also be complemented by an experimental comparison.

4.6 Representation of an ITM in Semantic Web

To facilitate the use and sharing of measures in our formalism, we use semantic web technologies to represent the concept of a TSoR and its components in RDF. We define the vocabulary CMD-v, which stands for Compound Measure Description vocabulary, as illustrated in Figure 4.21. As a reminder, a TSoR is defined as follows: $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$. Hence, a TSoR consists of a set of analysis elements ($\mathcal{A}$) and requirements ($\mathcal{R}$) that form a tree structure. The elements of the tree are linked by the relations "is specified by" (Sp) and "expects" (Ex).

Table 4.1: Comparison of the normalized impact of each requirement for FAIR-Checker and O'FAIRe, ordered by level on the hierarchy.

Analysis element	Impact on FAIR-Checker	Normalized impact on O'FAIRe
Findable	0.25	0.236
Accessible	0.25	0.236
Interoperable	0.25	0.228
Reusable	0.25	0.299
F1	0.125	0.086
F2	0.125	0.056
F3	0	0.044
F4	0	0.050
A1	0.25	0.195
A2	0	0.042
I1	0.083	0.092
I2	0.083	0.067
I3	0.083	0.069
R1	0.25	0.299
A1.1	0.125	0.059
A1.2	0.125	0.046
R1.1	0.083	0.077
R1.2	0.083	0.079
R1.3	0.083	0.075

In addition, they are associated with a content (name, description...) through the relation "hasContent". The content of an analysis element is of type `skos:Concept`, and a requirement is intended to measure a specific aspect, therefore its content is of type `dqv:Metric`. Note that analysis elements and requirements can only structure a single TSoR. However, their contents, concepts or metrics, can appear in multiple TSoRs.

A TSoR requires additional information to make it a TSoR-based measure ($\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$) and an implemented TSoR ($\mathcal{T}^I = \langle \mathcal{T}, i \rangle$). Therefore, an aggregation function is associated with a TSoR. For the sake of simplicity, weights are not placed at the edges of the tree, but directly at the nodes. Following the previous constraint that analysis elements and requirements can only be associated with one TSoR, this does not change anything on the definition nor on what the weights represent. Implementations are also linked to requirements. Other information can be added to enhance the transparency of the TSoR thus defined, such as creators, creation and modification dates, etc. The definition of the vocabulary is given in Appendix A.6 and is published online[5].

Regarding the use of operators with this representation, since analysis elements and

[5] https://w3id.org/cmd

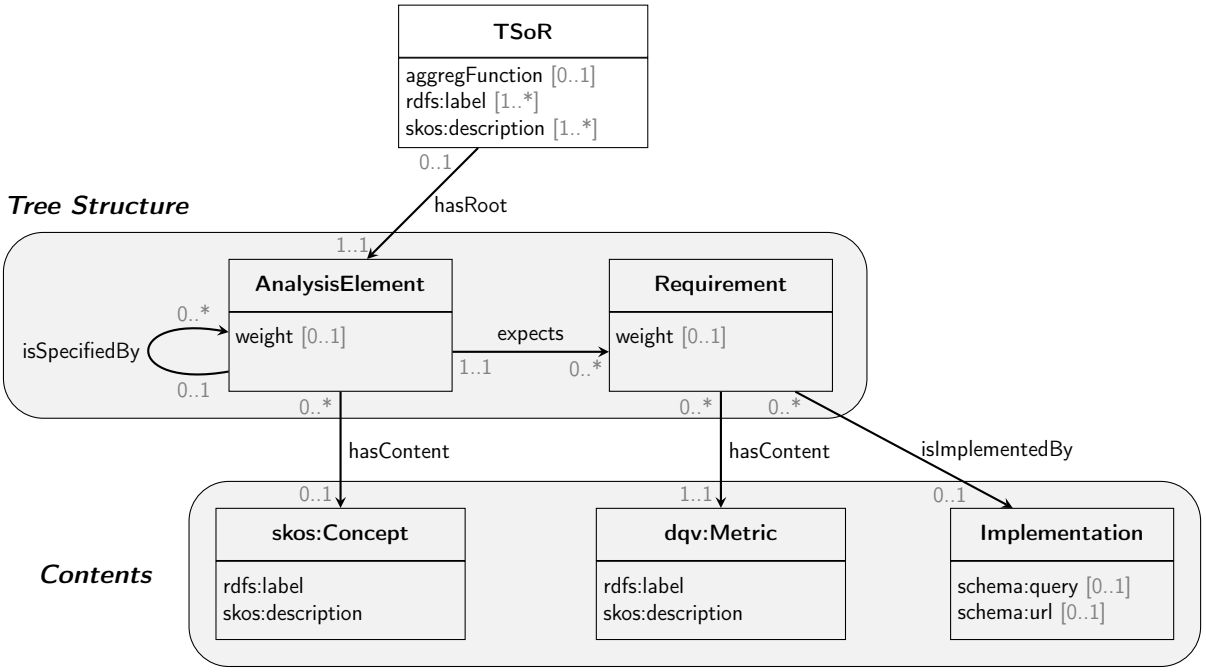


Figure 4.21: CMD-v: a vocabulary to express Implemented TSoR-based Measure

requirements can only structure a single TSoR, it is necessary to duplicate a TSoR when using the copy and extend operators, or when the purpose is to keep both the original TSoR and the resulting one.

In addition, SHACL constraints[6] ensure that a TSoR expressed with this vocabulary is properly defined. Some of these are shown in Figure 4.21 with the cardinalities in gray. These express constraints on the tree structure: a TSoR has exactly one root, an analysis element can specify only one other analysis element (and zero for the root). This constraint is illustrated by Listing 4.2. Similarly, a requirement must be expected by exactly one analysis element. And each analysis element and each requirement must have a content (concept or metric respectively), except for the root. In addition, there can be only one aggregation function per TSoR, one weight per analysis element and requirement, and one implementation per requirement. Some other constraints represent recommendations. They check whether an analysis element is either specified by another element or expects a requirement (no dead-end analysis element), but not both (uniformity of children type). They also check recommendations about complete implementation. Failing to satisfy a constraint on the tree structure is a violation, and a warning only for recommendation constraints. To validate both the vocabulary and the SHACL constraints, the KGAcc framework has been expressed[7] using CMD-v.

[6] https://github.com/Jendersen/TSoR-vocab/blob/main/cmd_shacl.ttl
[7] https://github.com/Jendersen/TSoR-vocab/blob/main/example.ttl

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix cmd: <https://w3id.org/cmd#> .

:AnalysisElementShape a sh:NodeShape ;
  sh:targetClass cmd:AnalysisElement ;
  # An analysis element must satisfy exactly one of these 2 shapes.
  sh:xone (
    # An analysis element specifies another element (is a child of)
    [ sh:property [
        sh:path [ sh:inversePath cmd:isSpecifiedBy ] ;
        sh:class cmd:AnalysisElement ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:name "Tree Structure of analysis elements: specifies." ;
        sh:severity sh:Violation ;
        sh:message "An AnalysisElement either specifies exactly one
    AnalysisElement (isSpecifiedBy), or is the root of exactly one
   TSoR (hasRoot)." ;
    ] ]
    # An analysis element is the root of a TSoR.
    [ sh:property [
        sh:path [ sh:inversePath cmd:hasRoot ] ;
        sh:class cmd:TSoR ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:name "Tree Structure of analysis elements: is root." ;
        sh:severity sh:Violation ;
        sh:message "An AnalysisElement either specifies exactly one
    AnalysisElement (isSpecifiedBy), or is the root of exactly one
   TSoR (hasRoot)." ;
    ] ]
  ) .
```

Listing 4.2: SHACL constraint representing that each analysis element should specify one analysis element or be the root of a TSoR

Comparison with data quality vocabulary

The Data Quality Vocabulary (DQV) [45] is a vocabulary for describing the data quality of datasets in RDF. Several types of quality information can be described: measurements, annotations (feedback or certificates), conformance to standards or data policies. The data model is represented in Figure 4.22. It extends the DCAT vocabulary and uses other vocabularies, among which PROV-O to express the provenance of quality information or SKOS to express concepts grouping the metrics according to the characteristic they evaluate.

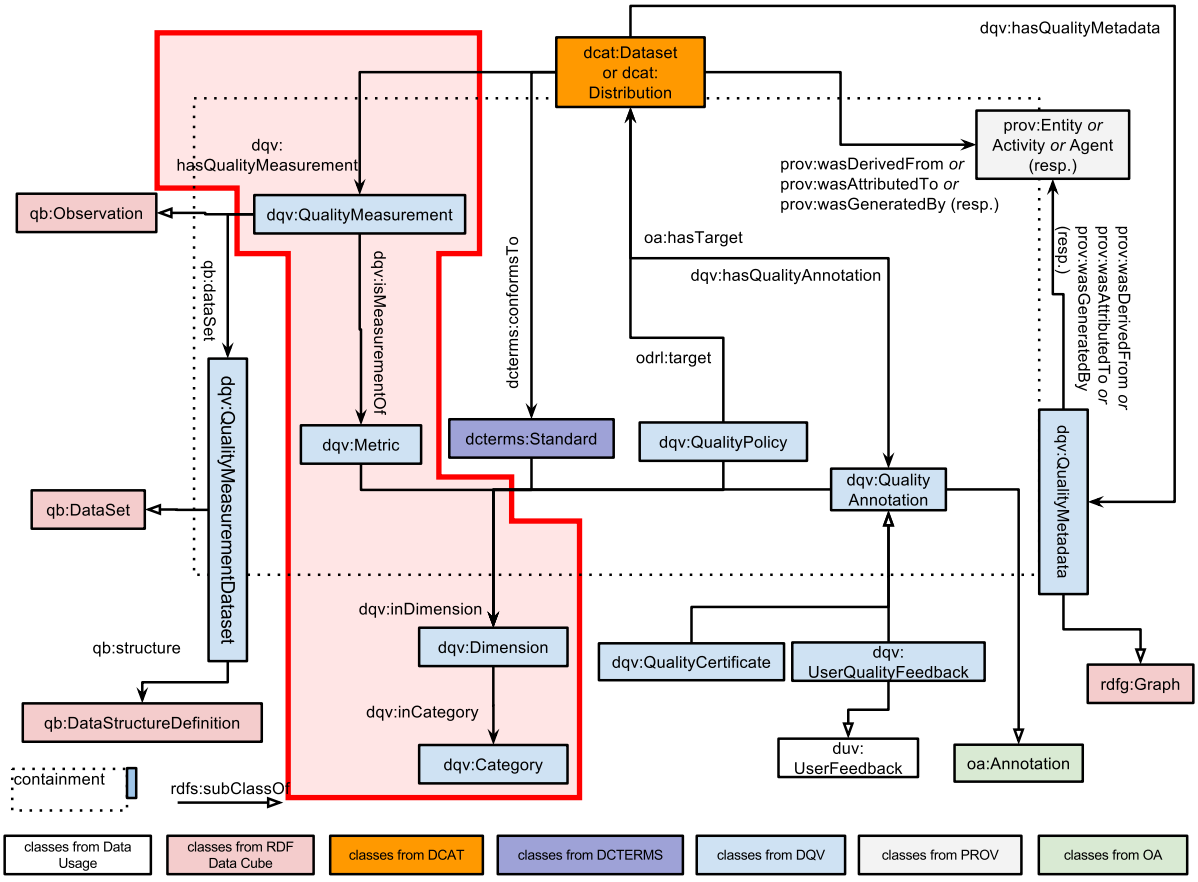


Figure 4.22: DQV data model [45] and the part that is related to CMD-v surrounded in red

DQV has some common points with our CMD vocabulary, they are highlighted in red in the figure: from *Dataset* to *QualityMeasurement*, *Metric*, *Dimension*, and *Category*. According to DQV, a quality measurement is an "evaluation of a given dataset against a specific quality metric". A metric represents "a procedure for measuring a data quality dimension, which is abstract, by observing a concrete quality indicator". Dimensions represent "a characteristic of a dataset relevant for assessing quality", such as availability or interoperability. They are grouped into categories, for instance, the accessibility category includes three dimensions: availability, licensing, and performance.

Let us compare DQV and CMD-v. First, these two models differ in their main objective. On the one hand, DQV aims at expressing the quality of a dataset that *has already been* measured. On the other hand, CMD-v expresses and describes a compound measure itself, meaning the measure against which a dataset *will be* evaluated. It aims to be explicit about the relative importance of the different elements and thus about how metrics will be aggregated by assigning a weight to the requirements and analysis elements. Therefore, CMD-v is not concerned with the expression of the result of evaluations.

However, both models express some quality indicators (metrics or requirements) and a way to structure them, in this way they have similarities. DQV indicates the score of a dataset on given metrics, and the central element of CMD-v is the requirement, which represents, i.e. has for content, a metric. For instance, a requirement may have as content the "downloadURLAvailabilityMetric" described by "Check if dcat:downloadURL is available and if its value is dereferenceable." [69].

In DQV, metrics are organized by dimensions, and a dimension must be associated with at least one metric. Similarly, it is recommended to associate an analysis element with at least one requirement. Then, categories represent a "group of quality dimensions", while analysis elements are organized with other analysis elements. Therefore, both dimensions and categories can be considered as analysis elements. In particular, for us, dimensions are leaf analysis and vice versa, hence, categories correspond to the other analysis. The hierarchical organization proposed by CMD-v is more flexible than that of DQV, indeed its model is limited to two levels whereas CMD-v allows to define deeper hierarchies, or a shallower ones. A possible solution mentioned by DQV [45] is to use SKOS and the relations `skos:narrower` and `skos:broader`, but this has not been studied in more detail nor explicitly added to the definition of the vocabulary. The relations between the two vocabularies are illustrated in Figure 4.23.

Obviously, DQV allows to express more types of quality information that are beyond the scope of CMD-v. Regarding compound measures, which are addressed by both vocabularies, it appears that CMD-v is designed to describe a wider range of hierarchies. Indeed, all metric organizations expressed in DQV can be expressed in CMD-v without loss of information. Listing 4.3 shows how to transform a Metric-Dimension-Category hierarchy in DQV into a TSoR in CMD-v. However, the class `dqv:QualityMeasurement`, i.e. the result of an evaluation of a given metric, has no equivalent in CMD-v. In fact, the latter does not aim at expressing elements of evaluation, for this part, DQV is better suited.

For the rest, CMD-v is more precise, requiring the implementation associated with each requirement. This is only partially covered by the use of PROV-O in DQV which is less precise or certain. And it provides more information such as relative importance, i.e. weights. Therefore, it explicitly provides the parameters that will be used to aggregate the results on

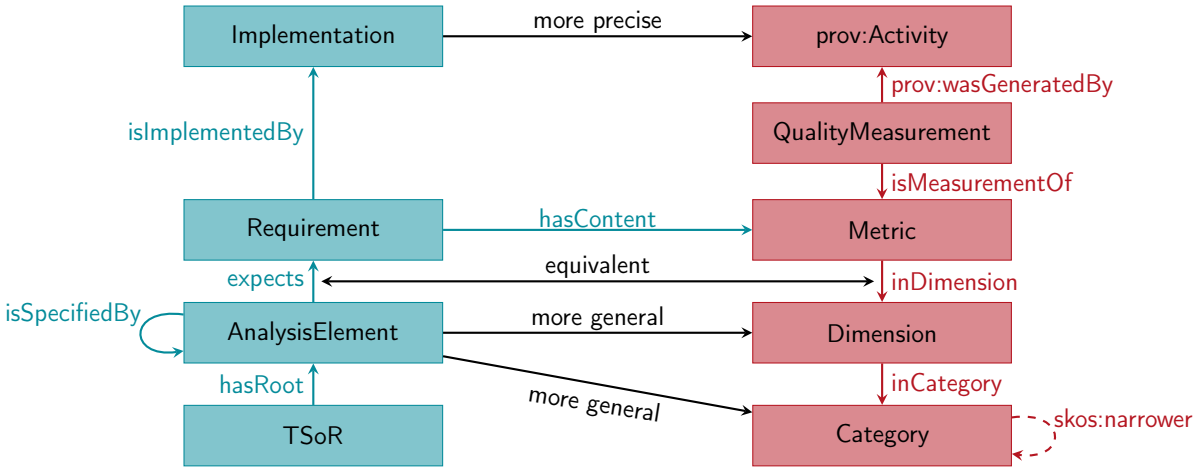


Figure 4.23: CMD-v (in blue) and DQV (in red), and relations between them (in black)

each analysis element. Finally, the structure of a TSoR allows to group analysis elements and requirements and to consider several TSoRs simultaneously. Therefore, DQV and CMD-v are complementary: CMD-v should be used to detail and share the measures, while DQV should continue to express any type of quality information, including the results of evaluations of the previously defined measures.

```
PREFIX dqv: <http://www.w3.org/ns/dqv#>
PREFIX cmd: <https://w3id.org/cmd#> .
PREFIX skos: <http://www.w3.org/2004/02/skos/core#> .

CONSTRUCT {
    ?cmd_met a cmd:Requirement ;
        cmd:hasContent ?met .
    ?met a dqv:Metric ;
        skos:prefLabel ?met_lab ;
        skos:definition ?met_desc .

    ?cmd_dim a cmd:AnalysisElement ;
        cmd:hasContent ?dim ;
        cmd:expects cmd_met .
    ?dim a skos:Concept ;
        skos:prefLabel ?dim_lab ;
        skos:definition ?dim_desc .

    ?cmd_cat a cmd:AnalysisElement ;
        cmd:hasContent ?cat ;
        cmd:isSpecifiedBy cmd_dim .
    ?cat a skos:Concept ;
        skos:prefLabel ?cat_lab ;
        skos:definition ?cat_desc .
```

```
25
26      ?root a cmd:AnalysisElement ;
27          cmd:isSpecifiedBy ?cmd_cat .
28      ?tsor a cmd:TSoR ;
29          cmd:hasRoot ?root .
30  } WHERE {
31      GRAPH ?g {
32          ?met a dqv:Metric .
33          OPTIONAL { ?met skos:definition ?met_desc . }
34          OPTIONAL { ?met skos:prefLabel ?met_lab . }
35
36          OPTIONAL { ?met dqv:inDimension ?dim . }
37          OPTIONAL { ?dim dqv:inCategory ?cat . }
38
39          OPTIONAL { ?dim a dqv:Dimension . }
40          OPTIONAL { ?dim skos:definition ?dim_desc . }
41          OPTIONAL { ?dim skos:prefLabel ?dim_lab . }
42
43          OPTIONAL { ?cat a dqv:Category . }
44          OPTIONAL { ?cat skos:prefLabel ?cat_lab . }
45          OPTIONAL { ?cat skos:definition ?cat_desc . }
46      }
47      BIND ( IRI(CONCAT(STR(?g), STR(?met))) AS ?cmd_met )
48      BIND ( IRI(CONCAT(STR(?g), STR(?dim))) AS ?cmd_dim )
49      BIND ( IRI(CONCAT(STR(?g), STR(?cat))) AS ?cmd_cat )
50      BIND ( IRI(CONCAT(STR(?g), "-root")) AS ?root )
51      BIND ( IRI(CONCAT(STR(?g), "-TSoR")) AS ?tsor )
52  }
```

Listing 4.3: Query to transform DQV hierarchy into a CMD-v TSoR

4.7 Conclusion

In this chapter, we have formalized some notions for expressing and describing compound measures. The central element is the TSoR (Tree-Structure of Requirements), which enables to define a measure based on requirements expressed in natural language and structured by analysis elements. The TSoR can be extended by implementations for the evaluation of a digital resource on each requirement and by weights for computing a global score.

These notions facilitate the reuse of existing measures and the construction of new ones, as well as their comparison. Indeed, we have defined some operators to build new TSoRs that focus on certain aspects of existing TSoRs, or to combine them to create TSoRs that are richer in information or in requirements. TSoRs can be compared in terms of their structure, by identifying the specific analysis elements or requirements of one TSoR compared to another, or by comparing the importance they give to each of these aspects. However, the comparison

is limited due to the difficulty of comparing natural language expressions.

Finally, we have proposed a semantic web representation of a TSoR to make it possible and easier to share and reuse them. The second objective is also to provide an explicit way of describing a measure to annotate data quality measurements and make them more transparent.

While this chapter is very theoretical, we have put these notions into practice by developing a web application dedicated to the construction of TSoRs, their comparison, the evaluation of knowledge graphs with them, and the visualization of the results. This is the subject of the next chapter.

5

A Web Application for Measures and KGs Evaluation

Contents

CHAPTER 4 presents formal bases for a web application that we present in this chapter. It enables the creation of measures in the form of Tree-Structure of Requirements (TSoR) and Implemented TSoR-based Measure (ITM): from the weighted structure, to the requirements expressed in natural language, to the implementation. It also provides a functionality to compare two ITMs. When the implementations are provided as SPARQL queries, the application allows the measure to be used to evaluate knowledge graphs through their SPARQL endpoint and to visualize the results.

5.1 Specifications and context of development

The existence of many measures based on a hierarchical structure and the difficulty of comparing and reusing them made us propose a formal framework for defining these measures in Chapter 4. To go further, we aim to develop a web application that allows anyone to explicitly define their measure, share it, and also to compare it with other measures defined in this application. This application is not limited to measures on knowledge graphs, but if a measure is implemented with SPARQL queries, we intend to provide the possibility to evaluate knowledge graphs directly *via* the application, based on the IndeGx framework [17], and to visualize the results.

The development of the application was initially centered around two functionalities, that were further extended. The first one consists in **enabling users to create an ITM**, and thus to:

- declare an analysis structure, as a tree of analysis elements ;
- define requirements and their links with analysis elements ;
- define queries and their links with requirements ;
- build an ITM based on existing ITMs (by implementing the corresponding operators) ;
- generate the RDF file containing the user-defined ITM, as well as the files required for

an evaluation with IndeGx.

The second one focuses on **visualizing the results of an evaluation campaign conducted with IndeGx.** It should enable to:

- compute the scores obtained by a KG globally and on its nodes of the ITM ;
- compare KGs according to their overall score ;
- compare KGs according to one or more analysis elements ;
- detail the results obtained by a KG on the analysis elements, or even on the requirements ;
- show additional information explaining the result obtained and suggestions for improvement.

The need for comparison arrived later in the development of the application.

The development of this application was proposed as the subject of an internship. Thus, it was realized by Arthur Durand, student at INSA Lyon, for his end of study project. This internship lasted five months, from April to August 2023. It was supervised by Philippe Lamarre and myself, with regular meetings to refine our needs, guide the technical choices, and check the progression.

Arthur Durand was free in his choice of technologies, with a few constraints imposed by the technical division of the laboratory in terms of reliability and security. Thus, the web application was developed using Go for the back-end, JavaScript and Bootstrap for the front-end. and MariaDB for the database. It is accessible at the following address: http://dekalog.liris.cnrs.fr:8080/.

5.2 Functionalities

The application has four main functionalities, illustrated in Figure 5.1. The first allows users to define their own measure as an ITM in the application. The second allows users to compare ITMs, i.e. some measures defined in the application. The third functionality uses an ITM to evaluate KGs through their SPARQL endpoint by using the IndeGx framework. Finally, the results obtained by evaluating the KGs can be visualized with the fourth functionality.

5.2.1 Creating measures as ITMs

The first objective of this application is to enable users to define their own measure in the form of an ITM. To do so, they start from a root and add the different elements that make up the measure: the analysis elements to form the structure, the requirements expressed in

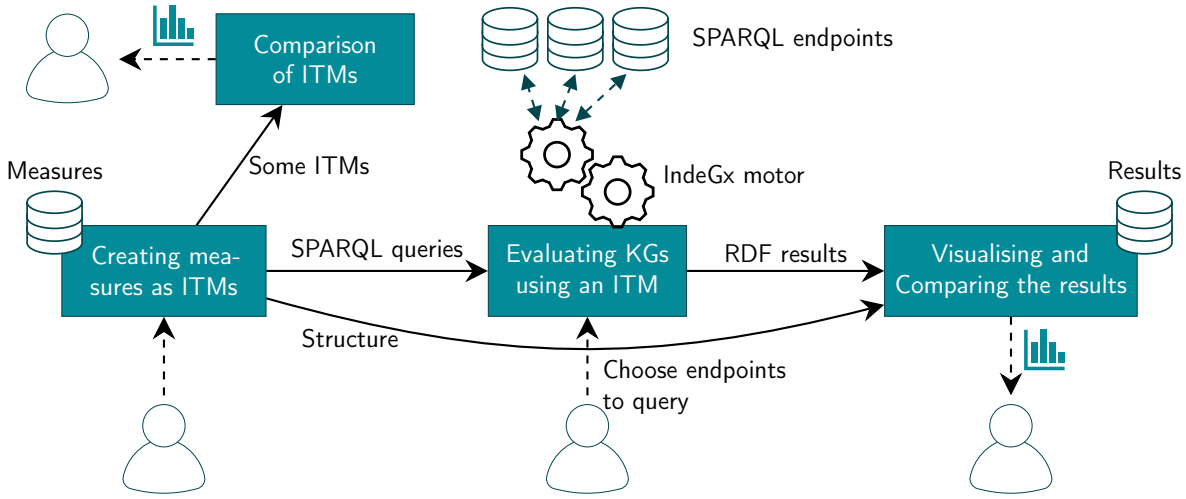


Figure 5.1: Main functionalities of the web application

natural language, their implementation (directly the implementation itself, such as a SPARQL query, or a URL pointing to the actual implementation), and the weighting elements. Currently, only the weighted average is fully supported by all functionalities of the application, but it is possible to choose a weighted average, a weighted sum, a min and a max.

Figure 5.2 illustrates the creation of the KGAcc measure from Chapter 3. The box in the upper left corner enables to show and modify specific parts of the measure. The measure itself is composed of the analysis elements (represented as rectangles, i.e., the first three levels), the requirements (represented as circles), and the implementation (the last level). Each element contains additional information, such as a longer description, which can be seen by clicking on it.

It is possible to visualize other measures already defined in the tool and even to reuse them as a basis for a new measure. For instance, the FAIR principles were defined as just a structure of analysis elements and then reused to define FAIR-Checker and O'FAIRe. When adding an analysis element or a requirement, it can be selected from a list of already defined elements, allowing comparison between two measures.

Currently, only one of the building operators defined in Chapter 4 is partially implemented: it is possible to copy the analysis structure of one measure into the node of another measure. However, unlike the copy operator, only the analysis elements are copied and not the requirements below them.

5.2.2 Comparing ITMs on their impacts

Once multiple measures have been defined as ITMs in the application, it becomes possible to compare them in terms of impact, as defined in Definition 13, page 82. For instance, the

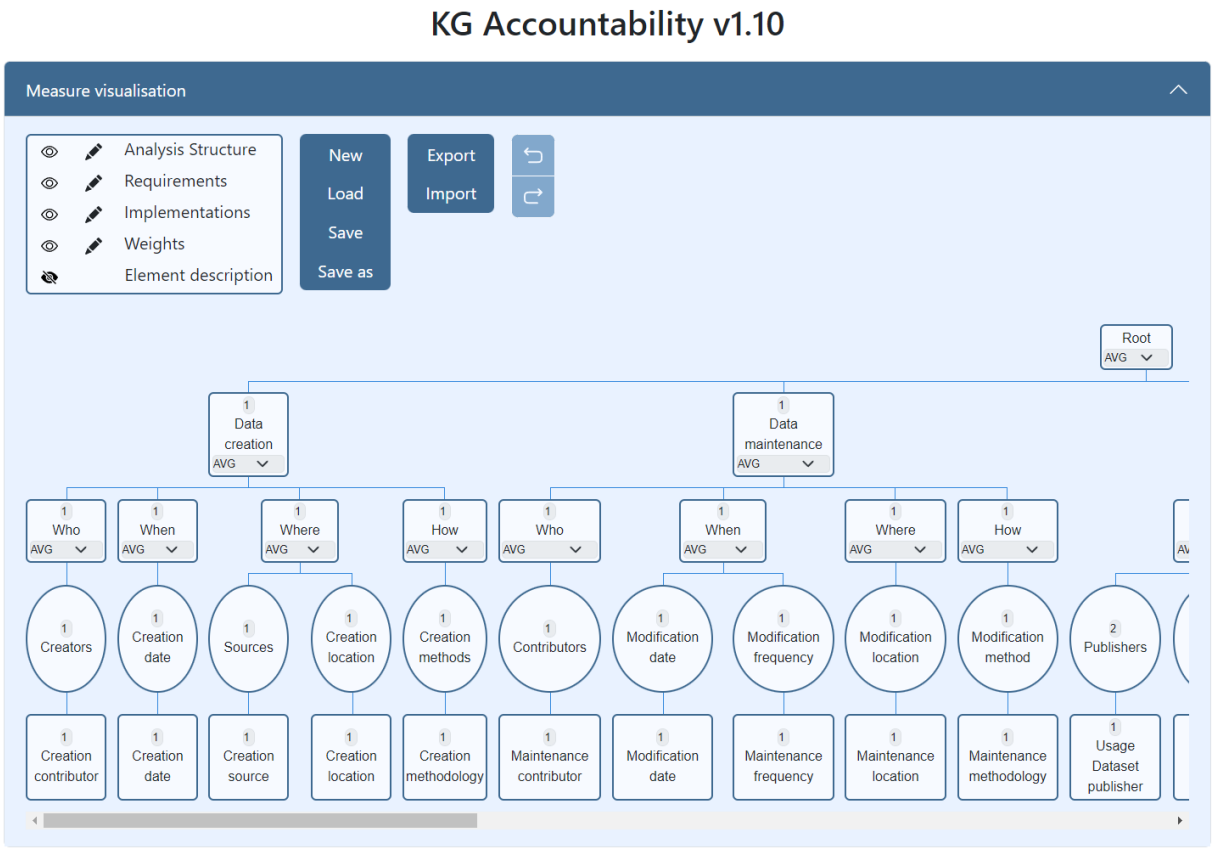


Figure 5.2: Screenshot of the measure creation of KGAcc measure in the web application

original FAIR principles [14], FAIR-Checker [15], and O'FAIRe [16] have been added to the application. The FAIR principles are represented as an ITM reduced to an analysis structure (i.e., without requirement), with a weight of 1 on each analysis element. This ITM was then refined with the specific requirements of FAIR-Checker and O'FAIRe. Indeed, to enable the comparison, the ITMs must be defined on the same basis, either by reusing an existing ITM or by choosing the same analysis elements among those already defined. The weights for the former have been arbitrarily chosen as 1 for each analysis element leading to a requirement and 0 for the others. The weights for the latter are based on the weights already defined by O'FAIRe. Therefore, Figure 5.3 shows the comparison between the impacts of the common analysis elements of the original FAIR principles, FAIR-Checker and O'FAIRe.

When there are equivalent nodes in the same measure, it is possible to compare impacts based on the cumulative impact of the equivalent nodes, or to compare the impact of a specific node identified by its path in the tree. Finally, the impact of the elements of one measure can also be seen without comparing it to another measure to better understand the importance of each element.

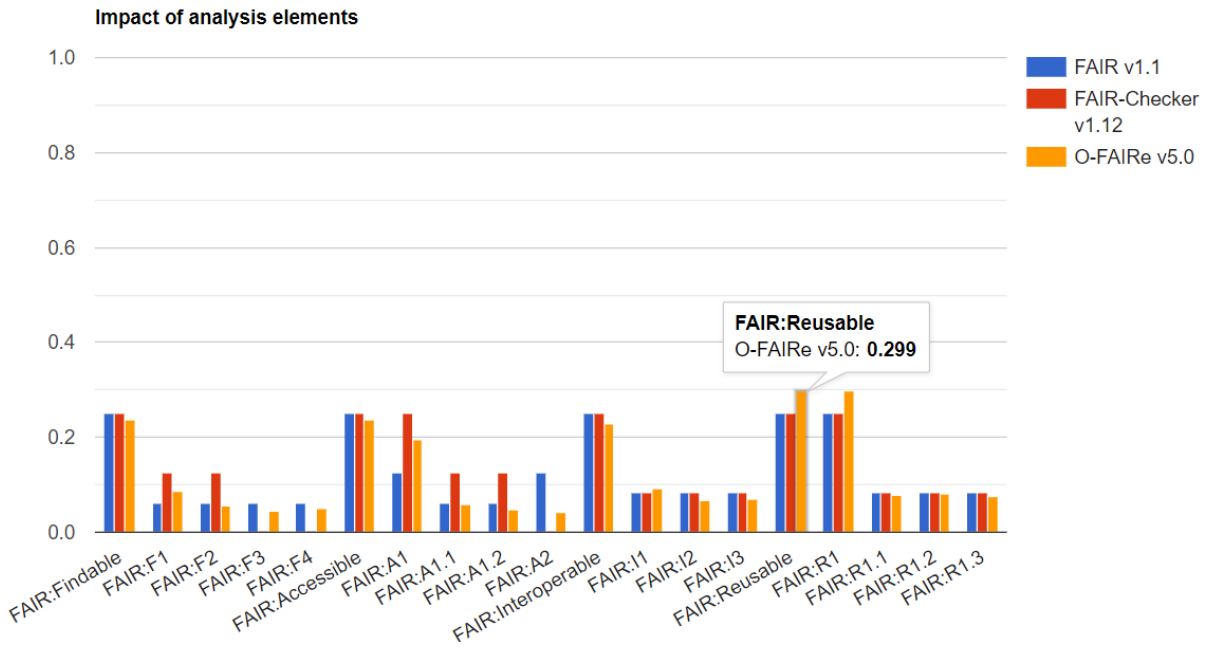


Figure 5.3: Screenshot of the measure comparison of three versions of the FAIR principles in the web application

5.2.3 Evaluating knowledge graphs using an ITM

The creation of an ITM is very permissive regarding the implementation. It can either be a URL of the actual implementation, or a text with no control over the language used. However, if the implementations are filled with SPARQL queries, then the measure can be used to evaluate knowledge graphs.

To do so, users can select the measure and enter the URL of the SPARQL endpoints to be evaluated with it. Then, the SPARQL queries of the measure are transformed into the input files needed by IndeGx, and IndeGx[1] is executed using these files on the specified SPARQL endpoints. The resulting RDF file is stored in the database and the results can be visualized using the last functionality.

There is currently an issue with character encoding when creating implementations as SPARQL queries. Therefore, it is not yet possible to use this functionality, even though it is already implemented.

5.2.4 Visualizing and comparing the results

Once some SPARQL endpoints have been evaluated on a given measure, it is possible to visualize the results and compare the evaluated endpoints. This is done by computing the global score

[1]Currently, the first version of IndeGx is used: https://github.com/Wimmics/dekalog

of the endpoints from the resulting RDF file, as well as the scores obtained for each analysis element. Detailed scores provide a precise view and understanding of the results. To go further, users can select exactly the nodes on which to visualize the results. For instance, on KGAcc, it is possible to focus only on the "data usage" step and to detail the score for each requirement.

It is also possible to change the weights on the structure on the fly (without changing the ITM itself) to change the importance of some elements. This is especially useful if the user analyzing the result is different from the user designing the ITM.

Since the previous step is not working, it is currently not possible to use this functionality. However, this was the first to be implemented, so the visualization has been tested and works properly. Figure 5.4 shows an old screenshot of the visualization.

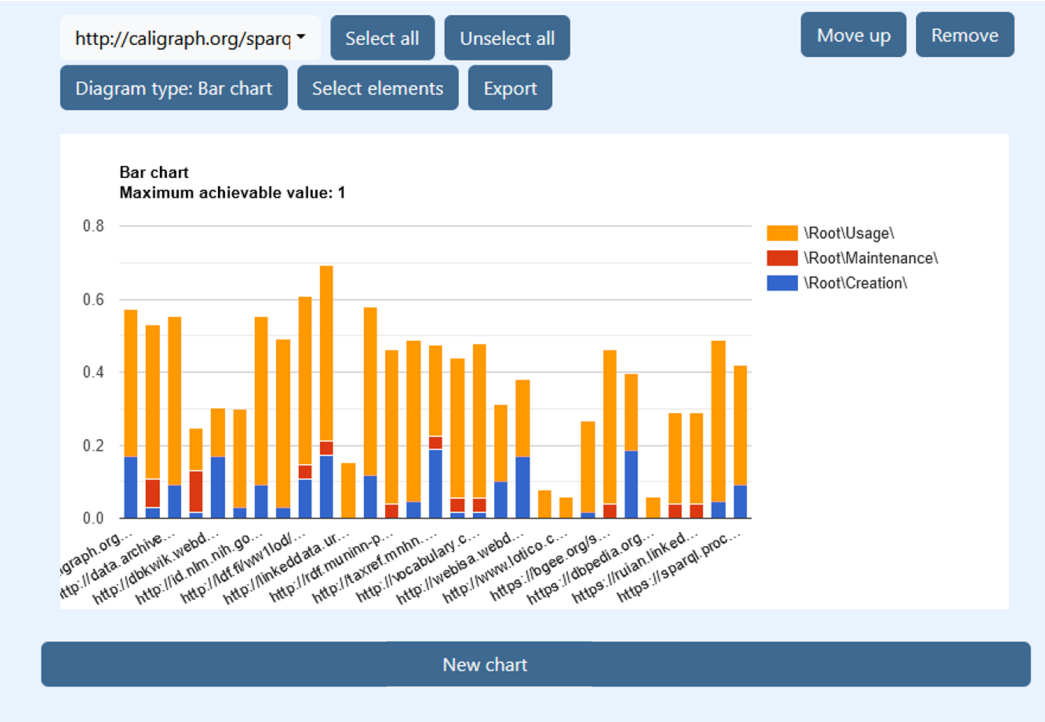


Figure 5.4: Screenshot of the web application for the visualization of the results of an evaluation campaign on the KGAcc measure

5.3 Future works and conclusion

As mentioned above, the web application is not fully functional yet. Therefore, we plan to fix it so that the evaluation can work properly. In addition, the application does not manage any form of account, which is a serious limitation for collaborative use. Indeed, currently anyone can modify an existing measure. This is something we would like to improve. Then, we

aim to implement more of the operators defined in the previous chapter. Finally, to improve transparency, sharing, and reuse of measures and evaluations, we would like to provide additional exports in RDF of both the measure using the vocabulary defined in Section 4.6, the material to run IndeGx based on the measure, and the results. Ultimately, it would be interesting to provide another access to the measures through a SPARQL endpoint for instance.

This web application aims to be a library to explore, understand, define, compare and reuse existing measures, and additionally to evaluate KGs. This objective is partially achieved, with the possibility to create and compare some measures. Despite some limitations to overcome, we believe in its potential and are working on it to achieve its main goals and share this application to a wider audience.

6

Conclusion

Contents

6.1 Research summary

The main and first objective of this thesis was to study the transparency of knowledge graphs. To do so, we first studied the concept of transparency and its companion concepts to try to understand it and to clarify the boundaries of this notion applied to a data source. We ended up with a new definition of transparency that emphasizes the contextual aspect of transparency: *A source's transparency w.r.t. a given need of observation consists in giving any observer access to all the suitable metadata regarding this need.* Together with the lack of more precise requirements for transparency, we concluded that evaluating transparency in absolute terms is neither possible nor desirable.

In addition to improving our understanding of transparency, the analysis of the companion concepts enabled us to identify an interesting notion with clearer requirements for datasets. Indeed, accountability, which is sometimes considered synonymous with transparency, is the subject of a detailed study and a metadata model, named LiQuID [21]. While this study is not specific to any type of dataset, we adapted it and its questions to propose a measure of the accountability of knowledge graphs. We applied this measure to evaluate knowledge graphs by querying their SPARQL endpoints in two campaigns. They show that only a few knowledge graphs provide metadata to describe themselves within their data. Indeed, we were able to identify the IRI of the evaluated KGs for less than ten percent of them. In this context, the evaluations also highlight the difficulty of identifying the description of a KG within its data. However, for those in which we found metadata, they provide different levels of accountability information, showing that our measure allows to discriminate between them. Finally, to analyze the relevance and novelty of this measure, we compared it with several measures of FAIRness and of data quality of knowledge graphs. It appears that the measure most similar to ours is

the O'FAIRe measure [16]. The latter looks for many properties similar to those required by our measure. However, neither O'FAIRe nor the other measures cover all aspects of accountability.

In the context of the ANR project DeKaloG, of which this thesis is a part, one of the objectives was to help users to choose a KG among others by providing an indicator of the transparency of KGs. In this regard, we used the IndeGx framework [17], developed by other members of the project, to conduct our experimental campaigns. Therefore, our measure of accountability gives a first approximation of transparency that can be used along with the index of KGs to provide additional information about them. This led to a joint publication with these members of the project [110].

By analyzing and comparing our accountability measure with others, we noticed that most of them have a similar structure, and that their comparison and reuse was difficult. This gave us the idea of a formal framework to define different measures in a common formalism. Therefore, we defined a TSoR, i.e. a Tree-Structure of Requirements, which promotes an explicit definition of the measure and both its structure and its criteria, or requirements, expressed in natural language. Additional information can be included, with implementations and elements describing how to compute an overall score. A measure fully specified in this way becomes an ITM: an Implemented TSoR-based Measure. We believe that exposing the structure and requiring both the requirements expressed in natural language and their implementation makes it easier to understand the measure. Then, we proposed several operators to help users create new ITMs based on existing ones. They enable to extract a part of an ITM to reduce the scope of the measure or to simplify its structure, or to combine several ITMs to enrich the structure or the requirements. We also defined some operators to compare TSoRs by identifying the common elements and their relative importance in each TSoR. However, the comparison requires knowing whether two elements are equivalent or not, which is a very difficult task when elements are expressed in natural language. In addition, it is common for some elements to be neither equivalent nor distinct. This can make comparison very difficult. Finally, coming back to the semantic web, we proposed a vocabulary to express measures following this framework.

To put this into practice, we specified a web application that enables users to manage their measures as TSoRs and ITMs. A first version of this tool has been implemented and already shows the comparison between two ITMs. It is also intended to evaluate knowledge graphs based on the measures defined in it, and to visualize the results. In the future, we hope that it will make it easier to reuse measures, that it will provide different quality indicators of KGs with just a few clicks on measures defined by different authors.

6.2 Discussions and perspectives

In this thesis, we identified several aspects that can be improved or further explored. Here are the main ones. The first concerns the KGs themselves, to overcome the small proportion that provide a self-description. The second aims to improve our measure of accountability. And the last one focuses on the comparison between measures.

First, the results of the evaluation of the KGs accountability showed that only a few of them have a findable self-description. This can be explained by several factors. There are no clear recommendations on what KGs should provide. In this respect, we believe that our accountability measure and the emergence of FAIRness measures can only help to improve KGs. In the same way, evaluations, indexes or our web application can serve as an incentive for KG providers, with a detailed view of the results and points of improvement. In addition, there is a lack of a uniform way to identify KG descriptions. The recommendation to make the description accessible via the `/.well-known/void` mechanism is a very good idea, as it is both very easy to find and machine-readable, but it is rarely followed by KG providers, and they are not directly accessible by querying the SPARQL endpoint. Therefore, we believe it would be interesting to introduce a new property or class that unambiguously indicates that such an entity represents the KG itself or the description of the queried KG. Alternatively, some KGs, such as CaLiGraph[1], identify it by including "void" or even ".well-known/void" in the IRI of the KG. This practice should also be encouraged.

Once such good practices have been adopted, we can take the measure of accountability a step further by covering more aspects of the LiQuID model. To do this, we will need to propose a new vocabulary to overcome the lack of the existing vocabularies.

Finally, we have also defined the formalization to facilitate comparison. However, a major limitation of this comparison is the difficulty of comparing elements expressed in natural language or implementations. To go further in the comparison, we will have to propose new strategies. For instance, it might be interesting to consider not only equivalence, but also inclusion. By enforcing the use of SPARQL queries for implementations, we can think of relying on the declarative aspect to compare property by property what is required. This would make the comparison much more precise, independent of natural language expressions, and therefore much more reliable.

[1] http://caligraph.org Accessed: 17 march 2024

A

Appendices

Contents

A.1 List of vocabularies used

Table A.1: Vocabularies used and their prefix

Full Name (Short Name)	**Prefix**	**Namespace**	**Type**
Creative Commons Rights Expression Language (ccRel)	cc	http://creativecommons.org/ns#	License
DataID	dataid	http://dataid.dbpedia.org/ns/core#	Dataset
Data Catalog Vocabulary (DCAT)	dcat	http://www.w3.org/ns/dcat#	Dataset
Dublin Core Metadata ElementSet (DC Elements)	dce	http://purl.org/dc/elements/1.1/	General
DCMI Type Vocabulary (DCMI Type)	dcmitype	http://purl.org/dc/dcmitype/	General
DCMI Metadata Terms (DC Terms)	dct	http://purl.org/dc/terms/	General
Description of a Project vocabulary (DOAP)	doap	http://usefulinc.com/ns/doap#	License
Data Quality Vocabulary (DQV)	dqv	http://www.w3.org/ns/dqv#	Quality
Evaluation and Report Language (EARL)	earl	http://www.w3.org/ns/earl#	Quality
Friend of a Friend vocabulary (FOAF)	foaf	http://xmlns.com/foaf/0.1/	General
IndeGx ontology	kgi	http://ns.inria.fr/kg/index#	IndeGx
Test case manifest vocabulary (Manifest)	mf	http://www.w3.org/2001/sw/DataAccess/tests/test-manifest#	IndeGx
NEPOMUK Information Element Core Ontology (NIE)	nie	http://www.semanticdesktop.org/ontologies/2007/01/19/nie#	License
The OWL 2 Schema vocabulary (OWL)	owl	http://www.w3.org/2002/07/owl#	Standard
Provenance, Authoring and Versioning (PAV)	pav	http://purl.org/pav/	Provenance
Protocol for Web Description Resources (POWDER-S)	powder-s	http://www.w3.org/2007/05/powder-s#	Metadata
W3C PROVenance Interchange (PROV-O)	prov	http://www.w3.org/ns/prov#	Provenance
The RDF Concepts Vocabulary (RDF)	rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Standard
The RDF Schema vocabulary (RDFS)	rdfs	http://www.w3.org/2000/01/rdf-schema#	Standard
schema.org	schema	http://schema.org/	General
SPARQL 1.1 Service Description (SPARQL-SD)	sd	http://www.w3.org/ns/sparql-service-description#	Dataset
Shapes Constraint Language Vocabulary (SHACL)	sh	http://www.w3.org/ns/shacl#	Standard
Simple Knowledge Organization System (SKOS)	skos	http://www.w3.org/2004/02/skos/core#	General
Industry 4.0 Knowledge Graph Ontology (STO)	sto	https://w3id.org/i40/sto#	License
Vocabulary of Interlinked Datasets (VoID)	void	http://rdfs.org/ns/void#	Dataset
XHTML Vocabulary (XHV)	xhv	http://www.w3.org/1999/xhtml/vocab#	License
XML Schema (XSD)	xsd	http://www.w3.org/2001/XMLSchema#	Standard

A.2 From LiQuID questions to KGAcc questions

Table A.2: Accountability requirements concerning **Data Collection**: Original questions from LiQuID and the adapted ones in the KGAcc framework with their weight (W)

Type	Questions from LiQuID	KGAcc Questions	W
Why	Why was the data set created?	(1)	
Who	Who (people, organizations) was involved in the data collection process? Provide all information relevant to their identification, their role in the data collection process, all information necessary to assess their qualifications to fulfill this role, and all characteristics which could have an influence on the data set.	Who are the creators of the KG and their role in this process? For all creators, indicates whether they are a person or an organization, provide information to identify them (name and point of contact such as email, or phone number, or address, or homepage), provide their qualifications, provide all characteristics which could have an influence on the KG.	1
When	On what date(s) or time frame(s) has the data been collected/ created? It must also be possible to place the data in a temporal context.	What is/are the creation date(s) of the KG?	1
Where	Where was the data set collected (country, place, website, ...)? It must also be possible to place the data in a spatial context.	From what original source(s) were the data collected or derived?	1/2
		From what physical location (state, country, continent, ...) was the KG created?	1/2
How	What was the methodology/ procedure for data collection?	Which methods or tools were used for data creation?	1
	Which methods and tools were exactly used in each step and what was the (technical) environment?	(1)	
What	What data was collected?	(2)	
	What concepts does it cover?	(2)	
	What is a general description of the data set?	(2)	
	What are the characteristics/ profile of the data set (dependent on data type)?	(2)	
	What is the quality of the data set (quality metrics depend on data type)?	(2)	

(1) LiQuID question not considered because vocabularies miss expressivity to consider the question.
(2) LiQuID question not considered because vocabularies miss expressivity to distinguish between collected data and published data (cf. Table A.4)

Table A.3: Accountability requirements concerning **Data Maintenance**: Original questions from LiQuID and the adapted ones in the KGAcc framework with their weight (W)

Type	Questions from LiQuID	KGAcc Questions	W
Why	Why will the dataset be further maintained?	(1)	
Who	Who (people, organizations) will be involved in the data maintenance? Provide all information relevant to their identification, their role in the data maintenance, all information necessary to assess their qualifications to fulfill this role, and all characteristics which could have an influence on the data set.	Who are the maintainers of the KG and their role in this process? For all maintainers, indicates whether they are a person or an organization, provide information to identify them (name and point of contact such as email, or phone number, or address, or homepage), provide their qualifications, provide all characteristics which could have an influence on the KG.	1
When	On what date(s) or time frame(s) will the data be maintained?	When was the KG last maintained/modified?	1
	With which frequency?	With which frequency is the KG maintained?	1
Where	Where will the data set be maintained (country, place, website, . . .)?	From what physical location (state, country, continent, ...) is or will the KG be maintained?	1
How	What will be the methodology/ procedure for data maintenance?	What will be the methodology/ procedure for data maintenance?	1
	Which methods and tools will exactly be used in each step and what will be the (technical) environment?	(1)	
What	What data will be the result of the data maintenance?	(2)	
	What concepts does it cover?	(2)	
	What is a general description of the data set?	(2)	
	What are the characteristics/ profile of the data set (dependent on data type)?	(2)	
	What is the quality of the data set (quality metrics depend on data type)?	(2)	

(1) LiQuID question not considered because vocabularies miss expressivity to consider the question.
(2) LiQuID question not considered because vocabularies miss expressivity to distinguish between collected data and published data (cf. Table A.4)

Table A.4: Accountability requirements concerning **Data Usage**: Original questions from LiQuID and the adapted ones in the KGAcc framework with their weight (W)

Type	Questions from LiQuID	KGAcc Questions	W
Why	What has the data set been used for?	(1)	
	For which other purposes can the published data set be used for?	(1)	
Who	Who publishes this data set?	Who publishes this KG?	1
	Who has used/ can use the published data set?	Who has the right to use the published KG?	1/2
		Who is intended to use the published KG?	1/2
When	When can/ was the published data set be used?	Since when was the KG available?	1
	When is it available?	Until when is the KG available?	1
	Until what point in time is it valid?	Until when is the KG valid?	1
Where	Where is the data set published/ available?	What is the webpage presenting the KG and/or allowing to gain access to it?	1/2
		Where to access the KG (either through a dump or a SPARQL endpoint)?	1/2
	Where (place, geographically) can the published data set be used?	In what physical location can the KG be used?	1
How	What is a recommended process for using the published data set?	What is the license of the KG?	1/3
		How to access the KG? Provide a SPARQL endpoint or a dump if they are freely accessible, or the procedure of access, and the characteristics of the endpoint if provided.	1/3
		How to use, reuse or integrate the KG?	1/3
	What are recommended methods, tools, and technical environments where the published data set can be used?	What are the requirements to use the KG?	1
What	What data is published for use?	What are examples of the published data?	1
	What concepts does it cover?	What concepts, topics or subjects does the KG cover?	1
	What is a general description of the data set?	What is a general description of the KG?	1
	What are the characteristics/ profile of the data set (dependent on data type)?	How many triples are there in the KG?	1/3
		How many entities, properties and classes are there in the KG?	1/3
		What RDF serialization formats does the KG support?	1/3
	What is the quality of the data set (quality metrics depend on data type)?	What is the quality of the KG?	1

(1) LiQuID question not considered because vocabularies miss expressivity

A.3 Properties used in KGAcc

The following table contains the list of properties for each query, as used in the second experiment. Each query is satisfied if at least one of the listed properties can be found associated with the IRI of the KG under study. In the table, properties that are in the same cell are considered equivalent when qualifying a knowledge graph in the context of measuring accountability only. Note that we did not check if the objects of the given "equivalent" properties are of compatible types, since our current objective is only to evaluate existing KGs, not to store and reuse the information elsewhere.

Table A.5: Properties (or property paths) used for each queries. Properties and property paths that are in the same cell are considered equivalent when qualifying a knowledge graph in the context of measuring accountability.

Query	Required Properties (or Property paths) and Equivalences	
DC Who: Creator	dce:creator	dct:creator
	foaf:maker	pav:createdBy
	pav:authoredBy	schema:author
	schema:creator	
	prov:wasGeneratedBy/prov:wasAssociatedWith/prov:actedOnBehalfOf* *(where the object is a person or an organization, plus constraints on the type of activity)*	
	dce:contributor	dct:contributor
	pav:contributedBy	schema:contributor
	schema:editor	
	prov:wasGeneratedBy/prov:wasAssociatedWith/prov:actedOnBehalfOf* *(where the object is a person or an organization, plus constraints on the type of activity)*	
	pav:curatedBy	
DC Who: Creator Additional queries	*See* `https://github.com/Jendersen/KG_accountability/blob/v2.0/rules/creation/contributor/manifest.ttl`	
DC When: Date	dct:created	pav:createdOn
	prov:generatedAtTime	schema:dateCreated
	prov:wasGeneratedBy/(prov:startedAtTime|prov:endedAtTime)	
	pav:curatedOn	
DC Where: Source	dce:source	dct:source
	pav:derivedFrom	pav:importedFrom
	pav:retrievedFrom	prov:hadPrimarySource
	prov:wasDerivedFrom	schema:isBasedOn

Query	Required Properties (or Property paths) and Equivalences
DC Where: Location	pav:createdAt schema:locationCreated prov:wasGeneratedBy/prov:atLocation
DC How: Methodology	pav:createdWith pav:importedBy pav:retrievedBy prov:wasGeneratedBy/prov:wasAssociatedWith *(where the object is a prov:SoftwareAgent, plus constraints on type of activity)*
	dct:accrualMethod
DM Who: Contributor	dce:contributor dct:contributor pav:contributedBy schema:contributor schema:editor prov:wasGeneratedBy/prov:wasAssociatedWith/prov:actedOnBehalfOf* *(where the object is a person or an organization, plus constraints on the type of activity)*
	schema:maintainer
DM Who: Contributor Additional queries	*See* https://github.com/Jendersen/KG_accountability/blob/v2.0/rules/maintenance/contributor/manifest.ttl
DM When: Date	dct:modified pav:contributedOn pav:lastUpdateOn schema:dateModified prov:wasGeneratedBy/(prov:startedAtTime|prov:endedAtTime)
DM When: Frequency	dct:accrualPeriodicity
DM Where: Location	prov:wasGeneratedBy/prov:atLocation
DM How: Method	dct:accrualMethod
	prov:wasGeneratedBy/prov:wasAssociatedWith *(where the object is a prov:SoftwareAgent)*
DU Who: Publisher	dce:publisher dct:publisher pav:providedBy schema:publisher schema:sdPublisher prov:wasGeneratedBy/prov:wasAssociatedWith/prov:actedOnBehalfOf* *(where the object is a person or an organization, and the activity a prov:Publish)*

Query	Required Properties (or Property paths) and Equivalences	
DU Who: Rights	dce:rights	dct:accessRights
	dct:rights	
	dct:license	cc:license
	doap:license	nie:license
	schema:license	schema:sdLicense
	sto:license	xhv:license
	dataid:openness	
DU Who: Audience	dct:audience	dataid:usefulness
	schema:audience	
DU When: Start availability	dct:available	dct:issued
	schema:datePublished	schema:sdDatePublished
	prov:wasGeneratedBy/(prov:startedAtTime|prov:endedAtTime)	
	(where the activity must be a prov:Publish)	
DU When: End availability	prov:invalidatedAtTime	schema:expires
DU When: End validity	dct:valid	schema:expires
DU Where: Webpage	dcat:landingPage	foaf:homepage
	schema:url	dcat:distribution/dcat:accessURL
DU Where: Access address	schema:contentURL	void:sparqlEndpoint
	dcat:accessService/dcat:endpointURL	
	dcat:accessService/sd:endpoint	
	^dcat:servesDataset/dcat:endpointURL	
	^dcat:servesDataset/sd:endpoint	
	void:dataDump	
	dcat:distribution/dcat:downloadURL	
	schema:distribution/schema:contentUrl	
DU Where: Location	dce:rights	dct:accessRights
	dct:rights	
	dct:license	cc:license
	doap:license	nie:license
	schema:license	schema:sdLicense
	sto:license	xhv:license
	dataid:openness	
DU How: License	dct:license	cc:license
	doap:license	nie:license
	schema:license	schema:sdLicense
	sto:license	xhv:license

Query	Required Properties (or Property paths) and Equivalences	
DU How: Access	schema:contentURL	void:sparqlEndpoint
	dcat:accessService/dcat:endpointURL	
	dcat:accessService/sd:endpoint	
	^dcat:servesDataset/dcat:endpointURL	
	^dcat:servesDataset/sd:endpoint	
	void:dataDump	
	dcat:distribution/dcat:downloadURL	
	schema:distribution/schema:contentUrl	
	dcat:distribution/dataid:accessProcedure	
DU How: Access Additional queries	*See* https://github.com/Jendersen/KG_accountability/blob/v2.0/rules/usage/how/manifest.ttl	
DU How: Reuse	dataid:reuseAndIntegration	
DU How: Requirements	dcat:distribution/dataid:softwareRequirement	
	schema:distribution/dataid:softwareRequirement	
DU What: Examples	schema:workExample	skos:example
	void:exampleResource	
DU What: Concepts	dcat:keyword	schema:keywords
	dcat:theme	dce:subject
	dct:subject	foaf:primaryTopic
	foaf:topic	schema:about
DU What: Description	dataid:dataDescription	dce:description
	dct:description	owl:comment
	powder-s:text	rdfs:comment
	schema:description	skos:note
DU What: Triples	void:triples	
DU What: Entities	void:entities	
	void:properties	
	void:classes	
DU What: Serialization	dce:format	dct:format
	schema:encodingFormat	void:feature
DU What: Quality	dqv:hasQualityAnnotation	schema:review
	dqv:hasQualityMeasurement	
	schema:award	

A.4 Additional cleaning operator

Recommendation 4 requires uniformity of child node type, i.e., the children of an analysis element should be either analysis elements or requirements, but not both. We formalize a cleaning operator that modifies a TSoR to conform to this recommendation: $\mathrm{UniformChildren}$ by adding "buffer" analysis elements between the faulty analysis elements and their requirements. We then show that the result is a TSoR and that it satisfies Recommendation 4.

Definition 24 (Uniform children type of a TSoR (Recommendation 4))**.** $\mathrm{UniformChildren}$ *modifies a TSoR to* ***make its children types uniform****. Let* $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ *be a TSoR. Let* $\alpha \in \mathcal{A}$ *such that* $children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset$ *and* $children^{\mathcal{R}}(\alpha, \mathcal{T}) \neq \emptyset$*. We define* $buffer(\alpha) \notin \mathcal{A}$ *that is a new analysis element that copies the content of* α*, adding the precision "other requirements".* $\mathrm{UniformChildren}(\mathcal{T}) = \langle \mathcal{A}', \mathcal{R}, Sp', Ex', \top \rangle$ *is defined by:*

- $\mathcal{A}' = \mathcal{A} \cup \{buffer(\alpha) \mid \alpha \in \mathcal{A}, children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset \wedge children^{\mathcal{R}}(\alpha, \mathcal{T}) \neq \emptyset\}$;
- $\mathcal{R}$ *is the same set of requirements ;*
- $Sp' = Sp \cup \{(\alpha, buffer(\alpha)) \mid \alpha \in \mathcal{A}, children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset \wedge children^{\mathcal{R}}(\alpha, \mathcal{T}) \neq \emptyset\}$;
- $Ex' = \big(Ex \setminus \{(\alpha, r) \mid (\alpha, r) \in Ex \wedge children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset\}\big)$
  $\cup \{(buffer(\alpha), r) \mid (\alpha, r) \in Ex \wedge children^{\mathcal{A}}(\alpha, \mathcal{T}) \neq \emptyset\}$;
- $\top$ *is the same root as in* $\mathcal{T}$*.*

Let $\mathcal{T}^I = \langle \mathcal{T}, i \rangle$ *be an implemented TSoR of* $\mathcal{T}$*. A modification of* $\mathcal{T}^I$ *for* ***uniform children types****, noted* $\mathrm{UniformChildren}(\mathcal{T}^I) = \langle \mathrm{UniformChildren}(\mathcal{T}), i \rangle$ *with no change in* i*.*

Let $\mathcal{T}_M = \langle \mathcal{T}, \diamondsuit, \delta \rangle$ *be a TSoR-based measure of* $\mathcal{T}$*. A modification of* $\mathcal{T}_M$ *for* ***uniform children types****, noted* $\mathrm{UniformChildren}(\mathcal{T}_M) = \langle \mathrm{UniformChildren}(\mathcal{T}), \diamondsuit, \delta' \rangle$ *is defined as:*

- $\mathrm{UniformChildren}(\mathcal{T}) = \langle \mathcal{A}', \mathcal{R}, Sp', Ex', \top \rangle$ *uniform children types of* $\mathcal{T}$;
- $\diamondsuit$ *does not change ;*
- δ' *depends on* $\diamondsuit$ *and their is no universal formula to define it. Let us detail it for two aggregation functions.*

If $\diamondsuit = \underset{\text{WA}}{\diamondsuit}$, *then*

$$\delta' : Sp' \cup Ex' \to \mathbb{R}^+$$
$$(b, b') \mapsto \begin{cases} \delta(b, b') & \text{if } (b, b') \in Sp \cup Ex \\ \sum_{r \in children^{\mathcal{R}}(b, \mathcal{T})} \delta(b, r) & \text{if } (b, b') \in Sp' \setminus Sp \quad (\text{i.e. } b' = buffer(b)) \\ \delta(b_1, b') & \text{if } (b, b') \in Ex' \setminus Ex, \text{ with } b = buffer(b_1) \end{cases}$$

If $\diamondsuit = \underset{\text{WS}}{\diamondsuit}$, *then*

$$\delta' : Sp' \cup Ex' \to \mathbb{R}^+$$
$$(b, b') \mapsto \begin{cases} \delta(b, b') & \text{if } (b, b') \in Sp \cup Ex \\ 1 & \text{if } (b, b') \in Sp' \setminus Sp \quad (\text{i.e. } b' = buffer(b)) \\ \delta(b_1, b') & \text{if } (b, b') \in Ex' \setminus Ex, \text{ with } b = buffer(b_1) \end{cases}$$

Property 6. *The modification* $\mathrm{UniformChildren}(\mathcal{T})$ *of a TSoR* $\mathcal{T}$ *for uniform children types is a TSoR. Similarly,* $\mathrm{UniformChildren}(\mathcal{T}^I)$ *of an implemented TSoR* $\mathcal{T}^I$ *is an implemented TSoR and* $\mathrm{UniformChildren}(\mathcal{T}_M)$ *of a TSoR-based measure* $\mathcal{T}_M$ *is a TSoR-based measure.*

The modification $\mathrm{UniformChildren}(\mathcal{T})$ *of a TSoR* $\mathcal{T}$ *for uniform children types satisfies Recommendation 4, which states that the children of an analysis element should all be of the same type.*

Proof. Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{R}, Sp, Ex, \top \rangle$ be a TSoR. Let us show that $\mathrm{UniformChildren}(\mathcal{T}) = \langle \mathcal{A}', \mathcal{R}, Sp', Ex', \top \rangle$ is also a TSoR. First, $\mathcal{A}'$, Sp', and Ex' satisfies the conditions of being respectively, a set of analysis elements, and relations between them and requirements.

So let us show that $\mathcal{G} = \langle \mathcal{A}' \cup \mathcal{R}, Sp' \cup Ex', \top \rangle$ is a directed tree rooted by $\top$. Since $\top \in \mathcal{A} \subset \mathcal{A}'$ and Sp' and Ex' are directed relations, we need to show that $\mathcal{G}$ is a tree (connected and acyclic).

Let $r \in \mathcal{R}$, then in $\mathcal{T}$, there exists a unique $\alpha \in \mathcal{A}$ such that $(\alpha, r) \in Ex$. Let α be such an element. Either α has only children of type requirements, then it is the only analysis element expecting r in $\mathrm{UniformChildren}(\mathcal{T})$: $(\alpha, r) \in Ex'$. Or, α also has children of type analysis elements, then $(\alpha, r) \notin Ex'$ and $buffer(\alpha)$ is the only analysis element expecting r.

Let $\alpha \in \mathcal{A}' \setminus \{\top\}$, if $\alpha \in \mathcal{A}$, then $\exists \alpha' \in \mathcal{A} \subset \mathcal{A}'$ such that $(\alpha', \alpha) \in Sp$, and α is not a "buffer". So α' is the only parent node of α. If $\alpha \notin \mathcal{A}$, then $\exists \alpha' \in \mathcal{A}, \alpha = buffer(\alpha')$, so $(\alpha', \alpha) \in Sp$, and α' is the only parent node of α.

Therefore, for each node in $\mathcal{A}' \cup \mathcal{R} \setminus \{\top\}$, there exists a unique parent node in $\mathcal{G}$. Therefore, $\mathcal{G}$ is a tree, and $\mathrm{UniformChildren}(\mathcal{T})$ is a TSoR. With no difficulty, $\mathrm{UniformChildren}$ of an implemented TSoR and a TSoR-based measure remain such elements, assuming the weights are well defined w.r.t. the aggregation function.

For all analysis elements whose children were both analysis elements and requirements, their relationships with requirements have been removed in favor of a new relationship with an analysis element. In addition, the analysis elements created are only parents of requirements. Therefore, the result of $\mathrm{UniformChildren}$ satisfies Recommendation 4 about a unique children type. □

A.5 Additional copy operator

In this section we define a copy operator. It enables to copy a part of a TSoR into another, by keeping all requirements of the copied TSoR. To copy a part of $\mathcal{T}_2$ into $\mathcal{T}_1$, it is necessary to specify a first analysis element α_2, which specifies the subtree $(\mathcal{T}_2, \alpha_2)$ to be copied. This subtree is connected to an analysis element α_1 of the TSoR $\mathcal{T}_1$. This connection is made such that α_1 replaces α_2, which is therefore not copied into $\mathcal{T}_1$. Hence, children and descendants of α_2 are copied into $\mathcal{T}_1$ and become children and descendants of α_1. For a TSoR-based measure, a weight should be given to indicate the relative importance of the children of α_2 compared to the existing children of α_1.

Definition 25 (Copy a TSoR into another with all requirements). *Let $\mathcal{T}_1, \mathcal{T}_2$ be two TSoRs such that $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$. Let α_1 be an analysis element of $\mathcal{A}_1$ and α_2 be an analysis element of $\mathcal{A}_2$ such that $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \cap \mathcal{A}_1 = \emptyset$, and $desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2) \cap \mathcal{R}_1 = \emptyset$. The result of the **copy with all requirements** of the subtree $(\mathcal{T}_2, \alpha_2)$ into $\mathcal{T}_1$, noted $\mathrm{CopyAll}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top \rangle$, is defined by:*

- *$\mathcal{A}'$ takes all elements of $\mathcal{A}_1$ and all descendants of α_2 in $\mathcal{T}_2$. Formally: $\mathcal{A}' = \mathcal{A}_1 \cup desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$;*
- *$\mathcal{R}'$ takes all requirements of $\mathcal{R}_1$ and all descendants of α_2 in $\mathcal{T}_2$. Formally: $\mathcal{R}' = \mathcal{R}_1 \cup desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)$;*
- *Sp' keeps the existing edges of Sp_1, the edges between descendants of α_2 and connects all children of α_2 to α_1. Formally:*

$$Sp' = Sp_1 \cup \left(Sp_2 \cap (desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2))^2\right) \cup \{(\alpha_1, \alpha) \mid \alpha \in children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)\}$$

- *Ex' keeps the existing edges of Ex_1, the edges between descendants of α_2 and connects all children of α_2 to α_1. Formally:*

$$Ex' = Ex_1 \cup \left(Ex_2 \cap (desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \times desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2))\right) \cup \{(\alpha_1, r) \mid r \in children^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)\}$$

*Let $\mathcal{T}^I{}_1 = \langle \mathcal{T}_1, i_1 \rangle, \mathcal{T}^I{}_2 = \langle \mathcal{T}_2, i_2 \rangle$ be two implemented TSoRs of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. A **copy with all requirements** of a part of $\mathcal{T}^I{}_2$ into $\mathcal{T}^I{}_1$ is defined by $\mathrm{CopyAll}(\mathcal{T}^I{}_1, \alpha_1, (\mathcal{T}^I{}_2, \alpha_2)) = \langle \mathrm{CopyAll}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)), i' \rangle$, with:*

- $i' = i_1 \cup i_2|_{desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)}$.

Let $\mathcal{T}_{M1} = \langle \mathcal{T}_1, \diamondsuit, \delta_1 \rangle, \mathcal{T}_{M2} = \langle \mathcal{T}_2, \diamondsuit, \delta_2 \rangle$ be two TSoR-based measures of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, such that the aggregation functions are the same. Let $\omega \in \mathbb{R}^{+*}$. A ***copy with all requirements*** *of a part of $\mathcal{T}_{M2}$ into $\mathcal{T}_{M1}$, noted $\mathrm{CopyAll}(\mathcal{T}_{M1}, \alpha_1, (\mathcal{T}_{M2}, \alpha_2), \omega) = \langle \mathrm{CopyAll}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)), \diamondsuit, \delta' \rangle$, is defined by:*

- *weight function is defined as follows:*

$$\delta' : Sp' \cup Ex' \to \mathbb{R}^+$$

$$(b, b') \mapsto \begin{cases} \delta_1(b, b') & \text{if } (b, b') \in Sp_1 \cup Ex_1 \\ \omega \times \delta_2(\alpha_2, b') & \text{if } b = \alpha_1 \text{ and } b' \in children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \\ \omega \times \delta_2(\alpha_2, b') & \text{if } b = \alpha_1 \text{ and } b' \in children^{\mathcal{R}}(\alpha_2, \mathcal{T}_2) \\ \delta_2(b, b') & \text{if } (b, b') \in Sp_2 \cup Ex_2 \end{cases}$$

Example 21. Let $\mathcal{T}_{M1}, \mathcal{T}_{M2}$ be two TSoR-based measures illustrated in Figure A.1. We define a new TSoR which is the copy of $\mathcal{T}_{M2}$ from its node C into the node A of $\mathcal{T}_{M1}$. In addition, we assert that the children of C are four times less important than x. This TSoR, $\mathrm{CopyAll}(\mathcal{T}_{M1}, A, (\mathcal{T}_{M2}, C), 1/4)$, is shown at the bottom of the same figure.

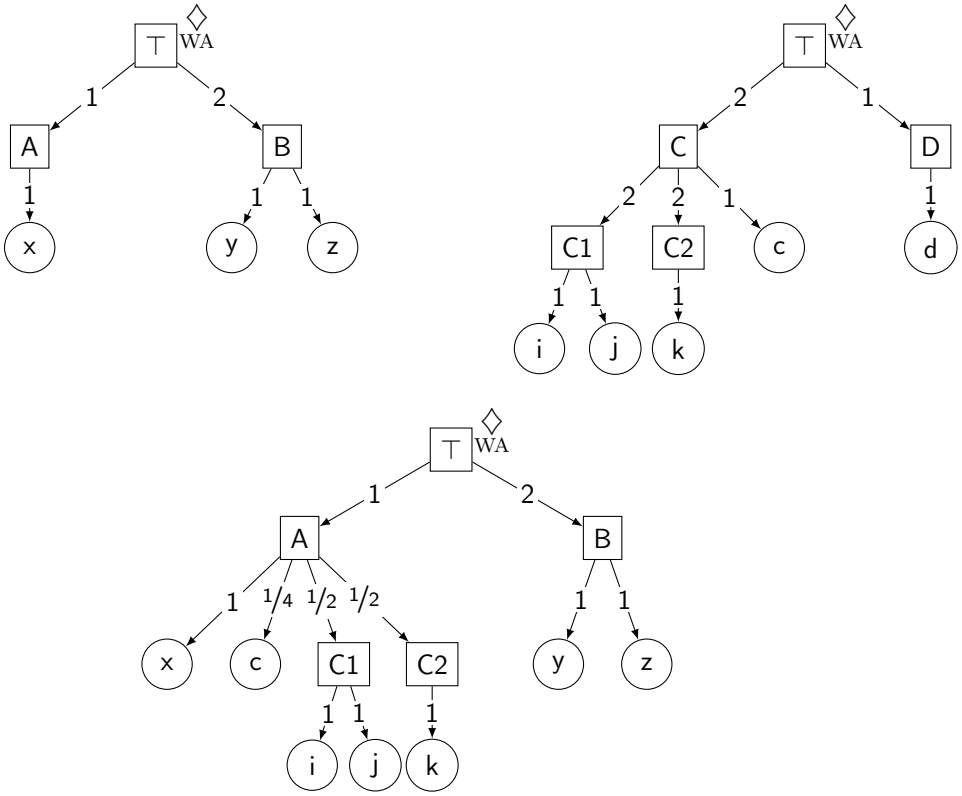


Figure A.1: Two TSoR-based measures and the copy with all requirements of one into the other

Property 7. *The copy with all requirements of a TSoR (an implemented TSoR, or a TSoR-based measure) into another is a TSoR (respectively, an implemented TSoR, a TSoR-based measure).*

Proof. Let $\mathcal{T}_1 = \langle \mathcal{A}_1, \mathcal{R}_1, Sp_1, Ex_1, \top_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}_2, \mathcal{R}_2, Sp_2, Ex_2, \top_2 \rangle$ be two TSoRs. Let $\alpha_1 \in \mathcal{A}_1$ and $\alpha_2 \in \mathcal{A}_2$ such that $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \cap \mathcal{A}_1 = \emptyset$, and $desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2) \cap \mathcal{R}_1 = \emptyset$. Let us show that $\mathrm{CopyAll}(\mathcal{T}_1, \alpha_1, (\mathcal{T}_2, \alpha_2)) = \langle \mathcal{A}', \mathcal{R}', Sp', Ex', \top' \rangle$ is a TSoR.

- $\mathcal{A}_1$ and $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$ are sets of analysis elements, therefore $\mathcal{A}'$ is a set of analysis elements.
- $\mathcal{R}_1$ and $desc^{\mathcal{R}}(\alpha_2, \mathcal{T}_2)$ are sets of requirements, so $\mathcal{R}'$ is a set of requirements.
- $\mathcal{A}_1 \subset \mathcal{A}'$, therefore $Sp_1 \subset \mathcal{A}' \times \mathcal{A}'$ and $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \subset \mathcal{A}'$, so $Sp_2 \cap (desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2))^2 \subset \mathcal{A}' \times \mathcal{A}'$. And, $\alpha_1 \in \mathcal{A}_1$ and $children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2) \subset desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$, so $\{(\alpha_1, \alpha) \mid \alpha \in children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)\} \subset \mathcal{A}' \times \mathcal{A}'$. Hence, Sp' is a set of relations between elements of $\mathcal{A}'$.
- Following exactly the same reasoning with requirements, we can show that, Ex' is a set of relations between elements of $\mathcal{A}'$ and of $\mathcal{R}'$.
- $\top_1 \in \mathcal{A}_1 \subset \mathcal{A}'$. So the root is one of the analysis elements.

Finally, let us show that $\mathcal{G} = \langle \mathcal{A}' \cup \mathcal{R}', Sp' \cup Ex', \top' \rangle$ is a directed rooted tree.

Let $\alpha \in \mathcal{A}_1 \setminus \{\top'\}$. Since $\mathcal{T}_1$ is a TSoR, then there exists a unique parent α' in $\mathcal{A}_1$ of α, and it is also a parent according to Sp'. In addition, $\alpha \notin desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$ by the condition on the two TSoRs. Therefore, α is not a child of another node according to the definition of Sp'. Finally, α has one, and only one, parent in $\mathcal{A}'$.

Let $\alpha \in desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$. By the condition on the two TSoRs, $\alpha \notin \mathcal{A}_1$, therefore, α does not appear in Sp_1. Either $\alpha \in children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$, so α_2 is its parent node, and it has no parent in $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$. Or $\alpha \notin children^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$, since $\mathcal{T}_2$ is a TSoR, it has exactly one parent node in $desc^{\mathcal{A}}(\alpha_2, \mathcal{T}_2)$ and α_1 is not its parent node. In both cases, α has exactly one parent in $\mathcal{A}'$. Therefore, all analysis elements of $\mathcal{G}$, except the root $\top'$, have exactly one parent node.

Following the same reasoning, we prove that every requirement of $\mathcal{G}$ has exactly one parent node. Hence, $\mathcal{G}$ is a directed rooted tree.

Without difficulty, copy an implemented TSoR into another remains an implemented TSoR. Finally, the weight is well defined for all elements in $Sp' \cup Ex'$, therefore copy of TSoR-based measure into another remains a TSoR-based measure. □

A.6 RDF vocabulary to describe a TSoR

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix vann: <http://purl.org/vocab/vann/> .
@prefix sw: <http://www.w3.org/2003/06/sw-vocab-status/ns#> .

# Vocabulary
<https://w3id.org/cmd#>
  a owl:Ontology ;
  a <http://purl.org/vocommons/voaf#Vocabulary> ;
  rdfs:label "Compound Measure Description"@en ;
  skos:altLabel "Complex Measure Definition"@en, "CMD-v" ;
  rdfs:comment "This document is a vocabulary to describe compound
   measures, i.e. measures with several metric or item that are
   organized with serveral dimensions. The description of such a
   measure relies on a Tree-Structure of Requirement (TSoR): a set
   of requirements structured hierarchicaly with analysis element.
   A TSoR represents the main measure. Several information may be
   added to explicitely indicate how the overall score on the
   measure should be calculated based on the hierarchy, relative
   importance of the node of the hierarchy and an aggregation
   function."@en ;
  dc:creator "Jennie Andersen" ;
  dc:contributor "Sylvie Cazalens", "Philippe Lamarre" ;
  dc:created "2022-25-08"^^xsd:date ;
  dc:modified "2024-03-01"^^xsd:date ;
  dc:date "2024-03-01"^^xsd:date ;
  owl:versionInfo 2.0 ;
  vann:preferredNamespacePrefix "cmd" ;
  vann:preferredNamespaceUri "https://w3id.org/cmd#" ;
  sw:term_status "UnderDevelopment" .


```

```
28 ##################################
29 #            Classes             #
30 ##################################
31
32 # TSoR
33 <https://w3id.org/cmd#TSoR>
34   a rdfs:Class ;
35   rdfs:label "TSoR: Tree-Structure of Requirements"@en ;
36   rdfs:comment "Defines a compound measure with a set of
      requirements as well as a structuration of these requirements
      through the use of analysis elements."@en .
37
38 # AnalysisElement
39 <https://w3id.org/cmd#AnalysisElement>
40   a rdfs:Class ;
41   rdfs:label "Analysis Element"@en ;
42   rdfs:comment "Node of the tree structure to classify requirements
      ."@en .
43
44 # Requirement
45 <https://w3id.org/cmd#Requirement>
46   a rdfs:Class ;
47   rdfs:label "Requirement"@en ;
48   rdfs:comment "Node of the tree structure representing a given
      metric to evaluate a concrete element."@en .
49
50 # Implementation
51 <https://w3id.org/cmd#Implementation>
52   a rdfs:Class ;
53   rdfs:label "Implementation"@en ;
54   rdfs:comment "An implementation describing a procedure and/or an
      executable document. Can either be expressed in a query language
      , or be refering to an executable file, or be precisly
      describing the procedure."@en .
55
56 ##################################
57 #           Properties           #
58 ##################################
```

```
# hasRoot
<https://w3id.org/cmd#hasRoot>
  a owl:ObjectProperty ;
  rdfs:label "has root"@en ;
  rdfs:comment "The TSoR has a given root among the analysis
   element."@en ;
  rdfs:domain <https://w3id.org/cmd#TSoR> ;
  rdfs:range <https://w3id.org/cmd#AnalysisElement> .

# isSpecifiedBy
<https://w3id.org/cmd#isSpecifiedBy>
  a owl:ObjectProperty ;
  rdfs:label "is specified by"@en ;
  rdfs:comment "Structures analysis elements through this relation.
    An analysis element is specified by one or more other analysis
   elements that detail the analysis."@en ;
  rdfs:domain <https://w3id.org/cmd#AnalysisElement> ;
  rdfs:range <https://w3id.org/cmd#AnalysisElement> .

# expects
<https://w3id.org/cmd#expects>
  a owl:ObjectProperty ;
  rdfs:label "expects"@en ;
  rdfs:comment "Associate a requirement with an analysis element:
   an analysis element expects a requirement."@en ;
  rdfs:domain <https://w3id.org/cmd#AnalysisElement> ;
  rdfs:range <https://w3id.org/cmd#Requirement> .

# hasContent
<https://w3id.org/cmd#hasContent>
  a owl:ObjectProperty ;
  rdfs:label "hasContent"@en ;
  rdfs:comment "A node of a TSoR has as content a given concept or
   metric."@en .

# isImplementedBy
<https://w3id.org/cmd#isImplementedBy>
  a owl:ObjectProperty ;
```

```
92    rdfs:label "is implemented by"@en ;
93    rdfs:comment "A requirement is implemented by an implementation."
       @en ;
94    rdfs:domain <https://w3id.org/cmd#Requirement> ;
95    rdfs:range <https://w3id.org/cmd#Implementation> .
96
97  # weight
98  <https://w3id.org/cmd#weight>
99    a owl:DatatypeProperty ;
100   rdfs:label "weight"@en ;
101   rdfs:comment "Number representing the relative importance of one
       node (analysis element or requirement) of a TSoR compared to its
        siblings."@en ;
102   rdfs:range xsd:double .
103
104 # aggregFunction
105 <https://w3id.org/cmd#aggregFunction>
106   a owl:DatatypeProperty ;
107   rdfs:label "aggregation function"@en ;
108   rdfs:comment "Definition of a function expressing how to compute
       a unique global score based on the result obtained on each
       requirements and their weight."@en ;
109   rdfs:domain <https://w3id.org/cmd#TSoR> .
```

Listing A.1: RDF description of the vocabulary to describe TSoR

Glossary

Assessment Framework
: An assessment (or evaluation) framework is an implemented tool that allows to evaluate datasets based on a single measure, or that can be used or extended with several measures.

Assessment Score
: An assessment score is the overall score obtained on the measure and computed based on the measurement of each metric using a scoring function.

Dataset
: In the context of RDF graphs, it may be either the whole RDF graph or a subgraph of it.

Dump
: A dump, or data dump, or RDF dump, is an RDF export of the dataset.

Knowledge Graph
: In this thesis, it refers to an RDF graph.

Measure
: A measure is an abstract concept that consists of an aggregated set of metric.

Measurement
: A measurement is the evaluation of a dataset (or a knowledge graph) with respect to a given metric, it is associated with the value obtained by the dataset on the metric.

Metric
: A metric refers to a procedure for measuring a given criterion, a single quality characteristic, or a requirement, it is a concrete and usually implemented quality indicator.

SPARQL endpoint
: A SPARQL endpoint is a web service that provides access to the data via SPARQL queries.

References

[1] H. Paulheim, "Knowledge graph refinement: a survey of approaches and evaluation methods", *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[2] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: the story so far", in *Semantic services, interoperability and web applications: emerging concepts*, IGI global, 2011, pp. 205–227.

[3] C. Bizer, M.-E. Vidal, and H. Skaf-Molli, "Linked open data", 2018.

[4] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase", *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: a nucleus for a web of open data", in *international semantic web conference*, Springer, 2007, pp. 722–735.

[6] G. Kobilarov *et al.*, "Media meets semantic web–how the bbc uses dbpedia and linked data to make connections", in *The Semantic Web: Research and Applications: 6th European Semantic Web Conference, ESWC 2009 Heraklion, Crete, Greece, May 31–June 4, 2009 Proceedings 6*, Springer, 2009, pp. 723–737.

[7] C. Stadler, J. Lehmann, K. Höffner, and S. Auer, "Linkedgeodata: a core for a web of spatial open data", *Semantic Web*, vol. 3, no. 4, pp. 333–354, 2012.

[8] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs.", *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, no. 1-4, p. 2, 2016.

[9] T. R. Gruber, "A translation approach to portable ontology specifications", *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[10] P. L. Whetzel *et al.*, "Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications", *Nucleic acids research*, vol. 39, no. suppl_2, W541–W545, 2011.

[11] R. Matheus and M. Janssen, "Transparency dimensions of big and open linked data", in *Conference on e-Business, e-Services and e-Society*, Springer, 2015, pp. 236–246.

[12] D. Wyatt, "The many dimensions of transparency: a literature review", *Helsinki Legal Studies Research Paper*, no. 53, 2018.

[13] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer, "Quality assessment for linked data: a survey", *Semantic Web*, vol. 7, no. 1, pp. 63–93, 2016.

[14] M. D. Wilkinson *et al.*, "The FAIR guiding principles for scientific data management and stewardship", *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[15] A. Gaignard, T. Rosnet, F. De Lamotte, V. Lefort, and M.-D. Devignes, "FAIR-checker: supporting digital resource findability and reuse with knowledge graphs and semantic web standards", *Journal of Biomedical Semantics*, vol. 14, no. 1, pp. 1–14, 2023.

[16] E. Amdouni, S. Bouazzouni, and C. Jonquet, "O'FAIRe makes you an offer: metadata-based automatic FAIRness assessment for ontologies and semantic resources", *International Journal of Metadata, Semantics and Ontologies*, vol. 16, no. 1, pp. 16–46, 2022.

[17] P. Maillot, O. Corby, C. Faron, F. Gandon, and F. Michel, "IndeGx: a model and a framework for indexing RDF knowledge graphs with SPARQL-based test suits", *Journal of Web Semantics*, p. 100 775, 2023.

[18] M. Ben Ellefi *et al.*, "RDF dataset profiling–a survey of features, methods, vocabularies and applications", *Semantic Web*, vol. 9, no. 5, pp. 677–705, 2018.

[19] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, "Information accountability", *Communications of the ACM*, vol. 51, no. 6, pp. 82–87, 2008.

[20] M. Basereh, A. Caputo, and R. Brennan, "AccTEF: a transparency and accountability evaluation framework for ontology-based systems", *International Journal of Semantic Computing*, vol. 16, no. 01, pp. 5–27, 2022.

[21] S. Oppold and M. Herschel, "Accountable data analytics start with accountable data: the liquid metadata model.", in *ER Forum/Posters/Demos*, 2020, pp. 59–72.

[22] E. Bertino, "The quest for data transparency", *IEEE Security & Privacy*, vol. 18, no. 3, pp. 67–68, 2020.

[23] T. Berners-Lee, J. Hendler, and O. Lassila, "A new form of web content that is meaningful to computers will unleash a revolution of new possibilities", *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001.

[24] A. Singhal *et al.*, "Introducing the knowledge graph: things, not strings", *Official google blog*, May 2012. [Online]. Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[25] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge", in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[26] X. L. Dong, "Challenges and innovations in building a product knowledge graph", in *Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining*, 2018, pp. 2869–2869.

[27] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger, "Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago", *Semantic Web*, vol. 9, no. 1, pp. 77–129, 2018.

[28] M. Lanthaler, D. Wood, and R. Cyganiak, "RDF 1.1 concepts and abstract syntax", W3C, W3C Recommendation, Feb. 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[29] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL", *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 3, pp. 1–45, 2009.

[30] S. Harris, A. Seaborne, and E. Prud'hommeaux, "SPARQL 1.1 query language", *W3C recommendation*, vol. 21, no. 10, p. 778, 2013.

[31] R. Guha and D. Brickley, "RDF schema 1.1", W3C, W3C Recommendation, Feb. 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[32] "OWL 2 web ontology language document overview (second edition)", W3C, W3C Recommendation, Dec. 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[33] P.-Y. Vandenbussche, G. A. Atemezing, M. Poveda-Villalón, and B. Vatant, "Linked open vocabularies (LOV): a gateway to reusable semantic vocabularies on the web", *Semantic Web*, vol. 8, no. 3, pp. 437–452, 2017.

[34] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao, "Describing linked datasets with the VoID vocabulary", *W3C Note. W3C*, 2011. [Online]. Available: https://www.w3.org/TR/void/.

[35] P.-Y. Vandenbussche, J. Umbrich, L. Matteis, A. Hogan, and C. Buil-Aranda, "SPARQLES: monitoring public SPARQL endpoints", *Semantic web*, vol. 8, no. 6, pp. 1049–1065, 2017.

[36] A. Hasnain, Q. Mehmood, S. S. e Zainab, and A. Hogan, "SPORTAL: profiling the content of public SPARQL endpoints", *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 12, no. 3, pp. 134–163, 2016.

[37] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach, "LOD laundromat: a uniform way of publishing other people's dirty data", in *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*, Springer, 2014, pp. 213–228.

[38] D. Browning, R. Albertoni, A. Perego, A. Gonzalez Beltran, S. Cox, and P. Winstanley, "Data catalog vocabulary (DCAT) - version 2", W3C, W3C Recommendation, Feb. 2020. [Online]. Available: https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/.

[39] M. Brümmer, C. Baron, I. Ermilov, M. Freudenberg, D. Kontokostas, and S. Hellmann, "DataID: towards semantically rich metadata for complex datasets", in *Proceedings of the 10th International Conference on Semantic Systems*, 2014, pp. 84–91.

[40] G. T. Williams, "SPARQL 1.1 service description", *W3C Recommendation. W3C*, 2013. [Online]. Available: https://www.w3.org/TR/sparql11-service-description/.

[41] A. J. Gray *et al.*, "Dataset descriptions: HCLS community profile", 2015.

[42] S. Bechhofer and A. Miles, "SKOS simple knowledge organization system reference", W3C, W3C Recommendation, Aug. 2009. [Online]. Available: https://www.w3.org/TR/2009/REC-skos-reference-20090818/.

[43] T. Lebo *et al.*, "PROV-o: the PROV ontology", W3C, W3C Recommendation, Apr. 2013. [Online]. Available: https://www.w3.org/TR/prov-o/.

[44] P. Ciccarese, S. Soiland-Reyes, K. Belhajjame, A. J. Gray, C. Goble, and T. Clark, "PAV ontology: provenance, authoring and versioning", *Journal of biomedical semantics*, vol. 4, no. 1, pp. 1–22, 2013.

[45] A. Isaac and R. Albertoni, "Data on the web best practices: data quality vocabulary", W3C, W3C Note, Dec. 2016. [Online]. Available: https://www.w3.org/TR/2016/NOTE-vocab-dqv-20161215/.

[46] T. Berners-Lee, *Linked data-design issues*, 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html (visited on 01/14/2024).

[47] J. Andersen, S. Cazalens, and P. Lamarre, "On the way to measure KG transparency: formalizing transparency - Requirements and first models", LIRIS UMR 5205, INSA Lyon, Tech. Rep., Dec. 2021. [Online]. Available: https://hal.science/hal-03986511.

[48] D. Schwabe, C. Laufer, and P. Casanovas, "Knowledge graphs: trust, privacy, and transparency from a legal governance approach", *Law in Context. A Socio-legal Journal*, vol. 37, no. 1, pp. 1–19, 2020.

[49] P. B. Brandtzaeg, A. Følstad, and M. Á. Chaparro Domínguez, "How journalists and social media users perceive online fact-checking and verification services", *Journalism practice*, vol. 12, no. 9, pp. 1109–1129, 2018.

[50] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences", *Genome biology*, vol. 11, no. 8, pp. 1–13, 2010.

[51] E. Bertino, S. Merrill, A. Nesen, and C. Utz, "Redefining data transparency: a multidimensional approach", *Computer*, vol. 52, no. 1, pp. 16–26, 2019.

[52] D. Firmani, L. Tanca, and R. Torlone, "Ethical dimensions for data quality", *Journal of Data and Information Quality (JDIQ)*, vol. 12, no. 1, pp. 1–5, 2019.

[53] S. Abiteboul, P. Bourhis, and V. Vianu, "Explanations and transparency in collaborative workflows", in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2018, pp. 409–424.

[54] N. Huijboom and T. Van den Broek, "Open data: an international comparison of strategies", *European journal of ePractice*, vol. 12, no. 1, pp. 4–16, 2011.

[55] E. Huaman and O. Paniasuk, "Definition of an Assessment Framework - MindLab D21", UIKB, Tech. Rep., 2020. [Online]. Available: https://drive.google.com/file/d/1YCwtuhsAbJUVw1WCaKJejDXXosyQgn4s/view.

[56] P. Senellart, "Provenance in databases: principles and applications", in *Reasoning Web Summer School (RW'19)*, 2019, pp. 104–109.

[57] S. C. Boulakia *et al.*, "Scientific workflows for computational reproducibility in the life sciences: status, challenges and opportunities", *Future Gener. Comput. Syst.*, vol. 75, pp. 284–298, 2017. [Online]. Available: https://doi.org/10.1016/j.future.2017.01.012.

[58] A. Gaignard, H. Skaf-Molli, and A. Bihouée, "From scientific workflow patterns to 5-star linked open data", in *8th USENIX Workshop on the Theory and Practice of Provenance, TaPP 2016, Washington, D.C., USA, June 8-9, 2016*, S. C. Boulakia, Ed., USENIX Association, 2016. [Online]. Available: https://www.usenix.org/conference/tapp16/workshop-program/presentation/gaignard.

[59] L. Moreau and P. Groth, "PROV-overview. an overview of the PROV family of documents", W3C, W3C Note, Apr. 2013. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/.

[60] P. Hayes and P. Patel-Schneider, "RDF 1.1 semantics", W3C, W3C Recommendation, Feb. 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

[61] D. Hernández, A. Hogan, and M. Krötzsch, "Reifying RDF: what works well with wikidata?", *SSWS@ ISWC*, vol. 1457, pp. 32–47, 2015.

[62] L. Moreau, P. Groth, J. Cheney, T. Lebo, and S. Miles, "The rationale of PROV", *Journal of Web Semantics*, vol. 35, pp. 235–257, 2015.

[63] G. Michener and K. Bersch, "Identifying transparency", *Information Polity*, vol. 18, no. 3, pp. 233–242, 2013.

[64] N. Geoscience, "Towards transparency", *Nature Geoscience*, vol. 7, no. 11, p. 777, 2014.

[65] ACM, *Artifact Review and Badging - Current*, Aug. 2020. [Online]. Available: https://www.acm.org/publications/policies/artifact-review-and-badging-current (visited on 03/19/2021).

[66] F. Sayre and A. Riegelman, "The reproducibility crisis and academic libraries", *College & Research Libraries*, vol. 79, no. 1, p. 2, 2018.

[67] B. Haibe-Kains *et al.*, "Transparency and reproducibility in artificial intelligence", *Nature*, vol. 586, no. 7829, E14–E16, 2020.

[68] M. Herschel, R. Diestelkämper, and H. B. Lahmar, "A survey on provenance: what for? what form? what from?", *The VLDB Journal*, vol. 26, no. 6, pp. 881–906, 2017.

[69] J. Debattista, C. Lange, S. Auer, and D. Cortis, "Evaluating the quality of the LOD cloud: an empirical investigation", *Semantic Web*, vol. 9, no. 6, pp. 859–901, 2018.

[70] X. Wang *et al.*, "Knowledge graph quality control: a survey", *Fundamental Research*, vol. 1, no. 5, pp. 607–626, 2021.

[71] B. Xue and L. Zou, "Knowledge graph quality management: a comprehensive survey", *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[72] R. Y. Wang and D. M. Strong, "Beyond accuracy: what data quality means to data consumers", *Journal of management information systems*, vol. 12, no. 4, pp. 5–33, 1996.

[73] S. Yumusak, E. Dogdu, H. Kodaz, A. Kamilaris, and P.-Y. Vandenbussche, "SpEnD: linked data SPARQL endpoints discovery using search engines", *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 4, pp. 758–767, 2017.

[74] Y. Yamamoto, A. Yamaguchi, and A. Splendiani, "Yummydata: providing high-quality open life science data", *Database*, vol. 2018, 2018.

[75] D. S. Katz, M. Gruenpeter, and T. Honeyman, "Taking a fresh look at FAIR for research software", *Patterns*, vol. 2, no. 3, 2021.

[76] C. Goble *et al.*, "FAIR computational workflows", *Data Intelligence*, vol. 2, no. 1-2, pp. 108–121, 2020.

[77] E. Schultes, B. Magagna, K. M. Hettne, R. Pergl, M. Suchánek, and T. Kuhn, "Reusable FAIR implementation profiles as accelerators of FAIR convergence", in *International Conference on Conceptual Modeling*, Springer, 2020, pp. 138–147.

[78] W. Hugo, Y. Le Franc, G. Coen, J. Parland-von Essen, and L. Bonino, *D2.5 FAIR semantics recommendations second iteration*, version 1.0, Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5362010.

[79] T. Rosnet, F. de Lamotte, M.-D. Devignes, V. Lefort, and A. Gaignard, "FAIR-checker - supporting the findability and reusability of digital life science resources", 2021.

[80] E. Amdouni and C. Jonquet, "Une méthodologie et un outil d'évaluation du niveau de "FAIRness" pour les ressources sémantiques: le cas d'agroportal", in *Jourrnées francophones d'Ingénierie des Connaissances (IC 2021)*, 2021.

[81] M. D. Wilkinson, S.-A. Sansone, E. Schultes, P. Doorn, L. O. Bonino da Silva Santos, and M. Dumontier, "A design framework and exemplar metrics for FAIRness", *Scientific data*, vol. 5, no. 1, pp. 1–4, 2018.

[82] B. RDA FAIR Data Maturity Model Working Group *et al.*, "FAIR data maturity model: specification and guidelines", *Res. Data Alliance*, vol. 10, 2020.

[83] D. Garijo, O. Corcho, and M. Poveda-Villalón, "FOOPS!: an ontology pitfall scanner for the FAIR principles.", in *ISWC (Posters/Demos/Industry)*, 2021.

[84] A. Devaraju and R. Huber, "An automated solution for measuring the progress toward FAIR research data", *Patterns*, vol. 2, no. 11, 2021.

[85] J. Brase, "Datacite-a global registration agency for research data", in *2009 fourth international conference on cooperation and promotion of information resources in science and technology*, IEEE, 2009, pp. 257–261.

[86] C. Sun, V. Emonet, and M. Dumontier, "A comprehensive comparison of automated FAIRness evaluation tools", in *13th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences*, 2022, pp. 44–53.

[87] M. Moser, J. Werheid, T. Hamann, A. Abdelrazeq, and R. H. Schmitt, "Which FAIR are you? a detailed comparison of existing FAIR metrics in the context of research data management", in *Proceedings of the Conference on Research Data Infrastructure*, vol. 1, 2023.

[88] S. Carroll *et al.*, "The CARE principles for indigenous data governance", *Data science journal*, vol. 19, 2020.

[89] D. Lin *et al.*, "The TRUST principles for digital repositories", *Scientific Data*, vol. 7, no. 1, pp. 1–5, 2020.

[90] S. Issa, O. Adekunle, F. Hamdi, S. S.-S. Cherfi, M. Dumontier, and A. Zaveri, "Knowledge graph completeness: a systematic literature review", *IEEE Access*, vol. 9, pp. 31 322–31 339, 2021.

[91] P. N. Mendes, H. Mühleisen, and C. Bizer, "Sieve: linked data quality assessment and fusion", in *Proceedings of the 2012 joint EDBT/ICDT workshops*, 2012, pp. 116–123.

[92] J. Debattista, S. Auer, and C. Lange, "Luzzu - a methodology and framework for linked data quality assessment", *Journal of Data and Information Quality (JDIQ)*, vol. 8, no. 1, pp. 1–32, 2016.

[93] E. Amdouni, S. Bouazzouni, and C. Jonquet, "O'FAIRe: ontology FAIRness evaluator in the agroportal semantic resource repository", in *ESWC 2022-19th Extended Semantic Web Conference, Poster and demonstration*. 2022.

[94] J. Andersen, S. Cazalens, and P. Lamarre, "Assessing knowledge graphs accountability", in *The Semantic Web: ESWC 2023 Satellite Events*, Heraklion, Greece: Springer Nature Switzerland, 2023, pp. 37–42. [Online]. Available: https://hal.science/hal-04169706.

[95] J. Andersen, S. Cazalens, P. Lamarre, and P. Maillot, "A framework to assess knowledge graphs accountability", in *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, IEEE, Venice, Italy: IEEE, Oct. 2023, pp. 213–220. [Online]. Available: https://hal.science/hal-04372234.

[96] M. Rowe and J. Butters, "Assessing trust: contextual accountability.", in *SPOT@ ESWC*, 2009.

[97] I. Naja, M. Markovic, P. Edwards, and C. Cottrill, "A semantic framework to support ai system accountability and audit", in *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, Springer, 2021, pp. 160–176.

[98] S. Oppold and M. Herschel, "LiQuID supplemental material - Model overview", Tech. Rep., 2020. [Online]. Available: https://www.ipvs.uni-stuttgart.de/departments/de/resources/mds/Information_Overview.pdf.

[99] K. Eckert and D. Garijo, "Dublin core to PROV mapping", W3C, W3C Note, Apr. 2013. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-dc-20130430/.

[100] B. Dutta, D. Nandini, and G. K. Shahi, "MOD: metadata for ontology description and publication", in *Proceedings of the International Conference on Dublin Core and Metadata Applications*, Dublin Core Metadata Initiative, 2015.

[101] E. Amdouni and C. Jonquet, "FAIR or FAIRer? an integrated quantitative FAIRness assessment grid for semantic resources and ontologies", in *Research Conference on Metadata and Semantics Research*, Springer, 2021, pp. 67–80.

[102] C. Jonquet, A. Toulet, B. Dutta, and V. Emonet, "Harnessing the power of unified metadata in an ontology repository: the case of agroportal", *Journal on Data Semantics*, vol. 7, no. 4, pp. 191–221, 2018.

[103] P. Maillot, O. Corby, C. Faron, F. Gandon, and F. Michel, "Metadatamatic: a web application to create a dataset description", in *Companion Proceedings of the ACM Web Conference 2023*, 2023, pp. 123–126.

[104] J. Andersen, S. Cazalens, and P. Lamarre, "Improving KG completeness evaluation considering information need as first-class citizen", LIRIS UMR 5205, INSA Lyon, Tech. Rep., Dec. 2022, p. 37. [Online]. Available: https://hal.science/hal-03986309.

[105] Q. Shi, J. Wang, J. Z. Pan, and G. Cheng, "VOYAGE: a large collection of vocabulary usage in open RDF datasets", in *International Semantic Web Conference*, Springer, 2023, pp. 211–229.

[106] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach", *Encyclopedia of software engineering*, pp. 528–532, 1994.

[107] B. Behkamal, M. Kahani, E. Bagheri, and Z. Jeremic, "A metrics-driven approach for quality assessment of linked open data", *Journal of theoretical and applied electronic commerce research*, vol. 9, no. 2, pp. 64–79, 2014.

[108] T. L. Saaty, "A scaling method for priorities in hierarchical structures", *Journal of mathematical psychology*, vol. 15, no. 3, pp. 234–281, 1977.

[109] S. Kubler, J. Robert, S. Neumaier, J. Umbrich, and Y. Le Traon, "Comparison of metadata quality in open data portals using the analytic hierarchy process", *Government Information Quarterly*, vol. 35, no. 1, pp. 13–29, 2018.

[110] P. Maillot, J. Andersen, S. Cazalens, C. Faron, F. Gandon, P. Lamarre, and F. Michel, "An open platform for quality measures in a linked data index", in *WWW'24 Companion: Companion Proceedings of the ACM Web Conference 2024*, Singapore, Singapore: Association for Computing Machinery, May 2024.

Publications

International conferences with peer review

- P. Maillot, J. Andersen, S. Cazalens, C. Faron, F. Gandon, P. Lamarre, and F. Michel, "An open platform for quality measures in a linked data index", in *WWW'24 Companion: Companion Proceedings of the ACM Web Conference 2024*, Singapore, Singapore: Association for Computing Machinery, May 2024.
- J. Andersen, S. Cazalens, P. Lamarre, and P. Maillot, "A framework to assess knowledge graphs accountability", in *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, IEEE, Venice, Italy: IEEE, Oct. 2023, pp. 213–220. [Online]. Available: https://hal.science/hal-04372234.
- J. Andersen, S. Cazalens, and P. Lamarre, "Assessing knowledge graphs accountability", in *The Semantic Web: ESWC 2023 Satellite Events*, Heraklion, Greece: Springer Nature Switzerland, 2023, pp. 37–42. [Online]. Available: https://hal.science/hal-04169706.

Technical reports for ANR DeKaloG project

- J. Andersen, S. Cazalens, and P. Lamarre, "Improving KG completeness evaluation considering information need as first-class citizen", LIRIS UMR 5205, INSA Lyon, Tech. Rep., Dec. 2022, p. 37. [Online]. Available: https://hal.science/hal-03986309.
- J. Andersen, S. Cazalens, and P. Lamarre, "On the way to measure KG transparency: formalizing transparency - Requirements and first models", LIRIS UMR 5205, INSA Lyon, Tech. Rep., Dec. 2021. [Online]. Available: https://hal.science/hal-03986511.

FOLIO ADMINISTRATIF

THESE DE L'INSA LYON, MEMBRE DE L'UNIVERSITE DE LYON

NOM : **ANDERSEN** DATE de SOUTENANCE : **5 juin 2024**
Prénoms : **Jennie**

TITRE : **De la transparence des graphes de connaissances à un cadre général pour la définition de mesures d'évaluation**

NATURE : **Doctorat** Numéro d'ordre : **2024ISAL0047**
École Doctorale : **École doctorale Informatique et Mathématiques de Lyon**
Spécialité : **Informatique**

RÉSUMÉ :
De nombreux graphes de connaissances (KG) sont disponibles sur le Web, et il peut être difficile de décider avec lequel travailler. Au-delà de la pertinence du domaine et du contenu, l'utilisation de standards, l'identification des créateurs... peuvent également influencer ce choix. En effet, la mise à disposition de toujours plus de données s'accompagne d'attentes supplémentaires en termes de qualité et de transparence.
Pour aider les utilisateurs à choisir un KG plutôt qu'un autre, nous voulons fournir une estimation de la transparence des KG. Les informations liées à la transparence sont essentielles pour renforcer la confiance dans les données et favoriser leur réutilisation. Cependant, il n'existe pas de définition consensuelle de la transparence. Pour mieux la comprendre, nous explorons tout d'abord cette notion et ses concepts associés (accessibilité, vérifiabilité...). Face à l'absence d'exigences précises concernant la transparence, nous nous concentrons ensuite sur un concept proche, et proposons une mesure de « l'accountability » des KG. Nous utilisons notre mesure pour évaluer des centaines de KGs disponibles via des SPARQL endpoints. Enfin, nous comparons notre mesure avec d'autres mesures pour les KG sur la qualité des données et les principes FAIR.
Ces comparaisons mettent en évidence des spécificités et des points communs pour ces multiples mesures. Aussi, choisir la mesure appropriée pour évaluer les KG dans le cadre d'une tâche donnée n'est pas aisé, d'autant plus qu'elles sont décrites de manières variées. Puisque beaucoup reposent sur une structure hiérarchique, nous proposons de définir une base formelle pour décrire les mesures dans un cadre commun. Nous souhaitons ainsi faciliter leur compréhension, leur réutilisation, leur comparaison et leur partage en définissant des opérateurs permettant de les manipuler, soit pour en créer de nouvelles, soit pour les comparer. Nous prolongeons ce cadre en proposant une application web.

MOTS-CLÉS : **Graphes RDF, Graphes de connaissances, Transparence, Accountability, Mesures.**

Laboratoire(s) de recherche : **Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)**
Directeur de thèse : **Philippe LAMARRE**

Président du Jury : **Frédérique LAFOREST**
Composition du Jury :
Mathieu D'AQUIN, Clément JONQUET, Fatiha SAÏS
Hala SKAF-MOLLI, Philippe LAMARRE, Sylvie CAZALENS