Integrating Open Data License Information into Data Pipelines

MASTER THESIS

Philip Rebbe

Submitted on 5 April 2024

Friedrich-Alexander-Universität Erlangen-Nürnberg
Faculty of Engineering, Department Computer Science
Professorship for Open Source Software

Supervisor:
Philip Heltweg, M. Sc.
Prof. Dr. Dirk Riehle, M.B.A.

Friedrich-Alexander-Universität
Faculty of Engineering

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others. The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Erlangen, 5 April 2024

License

Erlangen, 5 April 2024

Abstract

Today a lot of open data is available on the internet. To utilize this information to its full potential, it is often necessary to combine multiple datasets into new datasets. An important part of this is understanding and following the requirements of the licenses attached to them. This thesis presents a solution that can be used to integrate open data license information into a data pipeline. Following design-science, we construct a framework, that can be used to model and compare open data licenses. To achieve this, existing frameworks and solutions are compared and analyzed to find a solution fitting to open data licenses. The framework includes the functionality to check two licenses for compatibility, aggregate them into a composite license and give recommendations in case the composite license matches an existing license. The thesis also includes an implementation of the framework in the form of an extensible library and a demonstration, based on a website-prototype, on how the framework can be utilized to generate reports about different data-licenses. The framework is evaluated by comparing the results against existing frameworks from literature and testing if other licenses could be included in it as well.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

ASP Answer Set Programming

CC Creative Commons

CC-BY-4.0 Creative Commons Attribution 4.0 International

CC-BY-NC-4.0 Creative Commons Attribution Non-Commercial 4.0 International

CC-BY-ND-4.0 Creative Commons Attribution No-Derivatives 4.0 International

CC-BY-SA-4.0 Creative Commons Attribution Share-Alike 4.0 International

CSP Constraint-Satisfaction-Problem

ccREL Creative Commons Rights Expression Language

CSV Comma-Separated Values

DCAT-AP Data Catalogue Application Profile

GTFS General Transition Feed Specification

IODL 2.0 Italian Open Data License v2.0

JLA Joinup Licensing Assistant

JSON JavaScript Object Notation

JSON-LD JSON for Linked Data

OD Open Definition

ODbL Open Database License

ODC Open Data Commons

ODRL Open Digital Rights Language

REL Rights Expression Language

RDF Ressource Description Framework

SBOM Software Bill-of-Materials

SPDX Software Package Data Exchange

SQL Structured Query Language

UI user-interface

W3C World Wide Web Consortium

1 Introduction

1.1 General Motivation

Working with open data has become a standard practice in the IT-industry. Many companies or governments openly publish the information available to them, in an effort to become more transparent and to enable the industry to develop and provide new services and products. In response to this, big corporations like Google, small startups or even single individuals have gotten involved with and built entire businesses based on publicly available information (Brickley et al., 2019). In most cases, this entails building data pipelines to combine multiple datasets into one single data-source, that can be used to reveal new information or to provide more details to already existing information. To effectively do so not only requires a lot of knowledge on how to handle the published data, but also on the legal aspects attached to the data. This includes knowing the requirements and implications of different open data licenses and how to compare them to each other. This can be a challenge, since licenses are not standardized and can be quite hard to understand for someone who has no prior experience with them. It becomes even more difficult, when combining different datasets into a new dataset. In this case the engineers and researchers have to make sure that the combination of both datasets is allowed by the licenses and that the license of the resulting dataset is compatible with the previous licenses.

In this thesis, we propose a solution on how to integrate the information of open data licenses into a data pipeline. Using a design-science research approach, as described in (K. Peffers et al., 2007), we designed a framework that can automatically check licenses for compatibility and aggregate them into a synthetic license, that shows all the requirements both licenses entail. This aggregate is then used to find an existing license that can be recommended as a license for the new dataset. The thesis also includes an implementation and demonstration on how this framework can be used in a data pipeline.

1.2 Introduction into Data Pipelines

To be able to integrate license information into a data pipeline, we first have to understand how a data pipeline works. We will explain the basic concepts using the Jayvee-project[1] as an example. Since it has a good modular design, it is well suited to illustrate the different steps. Using this structure, we then explain how a pipeline works and how the license-information of the different datasets is connected to them.

In the Jayvee-project, pipelines are modeled as a combination of different blocks. Each block has a specific purpose and a predefined in- and output. Using these interfaces, multiple blocks can be connected to each other to create a data pipeline. There are three types of blocks:

- extractor-blocks
- transformator-blocks
- loader-blocks

Extractor-blocks are used to extract data from a data-source. Transformator-blocks model operations on the data like transforming them into a different data-format or removing information. Finally loader-blocks are used to transfer the result of these transformations into a data-sink. A sink defines a data-store like an Structured Query Language (SQL)-table, that is used to persist the results and potentially aggregate the results of different pipelines (JValue-Team, 2024).

To illustrate this concept a bit further we are going to reuse an example posted on the official Jayvee-website (see https://jvalue.github.io/jayvee/docs/user/examples/gtfs-rt). The example pulls three General Transition Feed Specification (GTFS)-files from the "French national access point to transport data" and pushes them into the same data-sink to create an aggregated dataset. Figure 1.1 illustrates the architecture of the pipeline.

Each feed is first run through an HttpExtractor-block to get the latest version of the data from the internet. Each dataset is then run through a GtfsRT- and TableInterpreter-block to bring the extracted data into a standardized format and finally exported using a SQLiteLoader to write the data into an SQLite-database. The resulting database contains three tables, each containing one of the pulled datasets.

In theory there is no limit to how many pipelines connect to a data-sink. This means, every sink can contain an unlimited number of datasets from different sources. It is also possible to support different source-formats like Excel-

[1] https://github.com/jvalue/jayvee

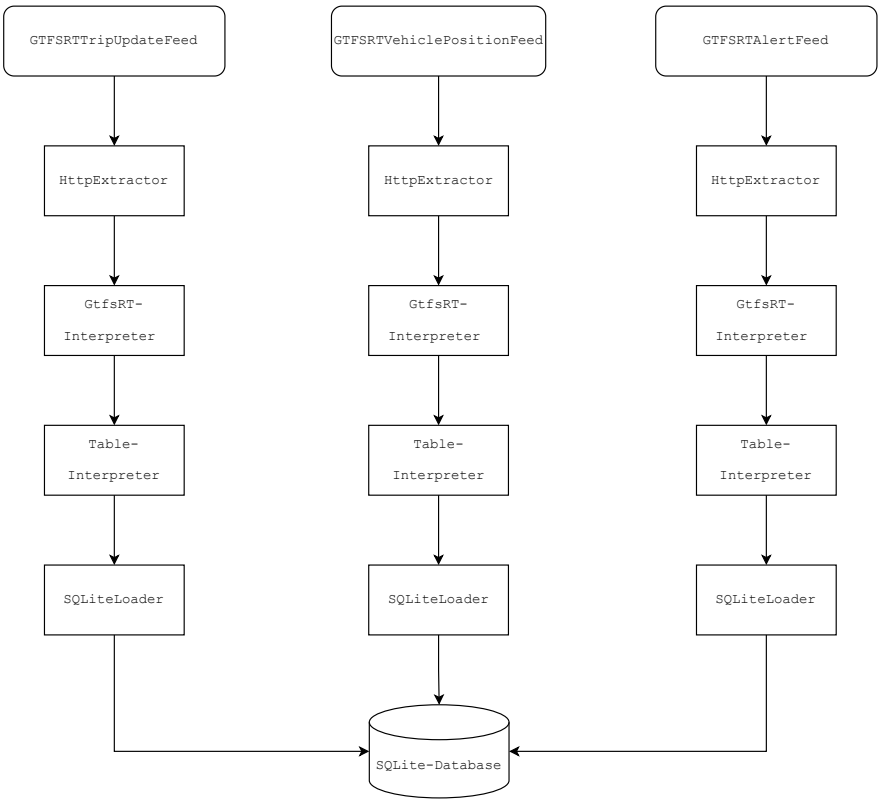


Figure 1.1: Illustration of the GTFS-RS example

spreadsheets or Comma-Separated Values (CSV) by using different transformations in each pipeline.

2 Problem Identification

The data pipeline, we described in the last section, only handled the aggregation of data. The process of handling the topic of licensing is left up to the user. This affects multiple steps of the process, starting with choosing the source-material and ending with the decisions of choosing a license for the created dataset. Each of these steps has some pitfalls that can lead to a wrong decision.

2.1 Difficulties when Choosing the Source Material

An important aspect of choosing the source-material is to understand the contents and requirements of the attached license. This includes finding the necessary license-information, understanding the terminology used in the licenses and how different legislations handle certain legal concepts. This step can be very difficult for someone with no prior experience of working with legal texts. On the other hand, it is also very important, since these requirements limit the use-cases in which you are allowed to use the source-material. As a study on open-source licenses (Almeida et al., 2017) has shown, this problem can still hold true for more experienced users, especially when working with multiple licenses at once.

2.1.1 Finding the License Information

When getting started with a new dataset, one of the first steps is to check the license that is attached to it. In the best case scenario, this information is directly linked to the dataset. This can be done by storing it as part of the meta-information of the dataset or displaying it on the website of the datahub or data-portal. But this is not always the case. There are some cases, where the license is only mentioned as part of the dataset. In other cases the dataset is licensed using the terms of usage of the data-hub or does not have a license at all. There can also be cases, in which a dataset or specific parts of them are licensed under multiple licenses (Ermilov & Pellegrini, 2015). This ambiguity can make finding the license information very confusing. Because of this automating this

process is very difficult.

2.1.2 Language Barriers

Another barrier to understanding a license can be the language it is written in. If a license is written in a language someone is not familiar with, he or she has no chance to know what is permitted or prohibited by a license. This problem can be illustrated by the Italian Open Data License v2.0 (IODL 2.0)[1]. The license is only available in Italian (without an official translation), meaning someone that only speaks English or German cannot know what is required by the license.

To the best of our knowledge, there is currently no tooling to reliably translate these texts. Normal automated translation tools (e.g. Google Translator) are not suited for the translation of legal texts, because they cannot guarantee that all legal concepts included in the license are translated correctly. Since they translate the text without any additional context information, they cannot make adjustments to their translations to cover missing information that would be necessary to correctly model some legal concepts in other languages or legal systems. There have been attempts like the "Law 10n"-project (Torres-Hostench & Salinas, 2015) that tried to solve this, but to our knowledge they are not ready yet. Currently the best solution (aside from consulting a lawyer), is for the license-provider to create an official translation. A good examples for this are the Creative Commons (CC)-licenses. There are currently 28 ports of the current version 4 available on their website[2], including translations to English, Spanish and German. Some national licenses like the "Datenlizenz Deutschland" are written in the corresponding native language, but provide at least an English translation to ensure the license is better suited for international use. But this practice has not become a standard yet, and in most cases there is only one version of a license, meaning there is still not a perfect solution to make the texts of a license available to everyone.

2.1.3 Understanding the Legal Aspects of Licenses

Another problem is the license-content itself, meaning understanding the legal terms and definitions used in the license-texts. These definitions are necessary to make a license applicable to different legislations, which is necessary to enable users to enforce it in multiple countries. They additionally leave some room for interpretation, so a license can be used for many different use-cases without having to explicitly declare every possible use-case. However they are not standardized, meaning different providers can use different definitions for the same condition, causing uncertainty about the consequences of these actions. A good

[1] https://www.dati.gov.it/content/italian-open-data-license-v20
[2] https://creativecommons.org/licenses/

example for this is the "Share-Alike"-condition used by different licenses. In general, the term "Share-Alike" describes a so-called "copyleft"-mechanism, which means that it enforces some restrictions on the licenses that can be used to license a derivative of the licensed dataset (Yi-Hsuan Lin et al., 2006). The license Creative Commons Attribution Share-Alike 4.0 International (CC-BY-SA-4.0)[3] defines "Share-Alike" in the following way:

> The Adapter's License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-SA Compatible License.

In this case the term "BY-SA Compatible License" is defined as:

> BY-SA Compatible License means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

To summary this, it means that you either have to reuse the current license or use a license that is verified by CC as being compatible with their license. In comparison to that the Open Database License (ODbL)[4] provided by Open Data Commons (ODC) defines "Share Alike" in the following way:

> a. Any Derivative Database that You Publicly Use must be only under the terms of:
>
> i. This License;
>
> ii. A later version of this License similar in spirit to this License; or
>
> iii. A compatible license.
>
> If You license the Derivative Database under one of the licenses mentioned in (iii), You must comply with the terms of that license.

It is similar to the CC-license, but it only mentions that the license has to be compatible with the ODbL. While this theoretically enables licensees to use licenses that could be compatible, it can also effectively limit them to the ODbL, since this is the only license that is officially mentioned. This shows, that while these difference are not very drastic, they still can lead to some uncertainty regarding the contents of a license.

"Share-Alike" is not the only condition that can have this effect. Another example of this is the "NonCommercial"-condition, that is part of some of the CC-licenses. It is defined as:

[3] https://creativecommons.org/licenses/by-sa/4.0/legalcode.en
[4] https://opendatacommons.org/licenses/odbl/1-0/

> NonCommercial means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange.

Here the problem is not the definition itself, but how a specific purpose applies to it. Most people will have a rough understanding of what non-commercial use requires. But if you have a more complicated use-case (e.g. the data is freely available, but some other independent component of the app requires a payment) it almost always requires legal advice, since the license-text alone does not provide information on its own.

There are also actions that can prevent the combination of two datasets. A simple example for this is the license "Creative Commons Attribution No-Derivatives 4.0 International (CC-BY-ND-4.0)". It allows users to share the licensed material, but only in an unmodified version. Since combining two datasets into a new dataset is classified as a modification of the dataset, this condition effectively prohibits the combination with another dataset. This action is independent of the conditions of other licenses, meaning it cannot be overruled by another license, since this would violate the rights of the original licensor. In these cases the user has to stop the aggregation and look if the dataset can be used in a different way.

2.2 Difficulties when Transforming and Persisting the Data

In the pipeline the data is first transformed and then written into the data-sink. At this point in time you have to take two things into consideration:

- What usage is allowed by the source-licenses?
- Under which license do you want to license the new dataset and are you allowed to do so?

Understanding what usages are allowed is mainly affected by someones understanding of the licence and their ability to transfers this knowledge to the current use-case. Since we already covered this in the previous sections, we are going to focus on the second question. The license for the aggregate has to cover all the requirements of the previous licenses, meaning the creator of the pipeline has to ensure all requirements are fulfilled and that there are no conflicts between the different licenses. This process is often even more difficult than simply understanding the licenses themself (Almeida et al., 2017).

2.2.1 Deciding on a License for the Aggregate

In some cases this problem can be solved by comparing the permissions and conditions of different licenses and how they compare to each other. For permissions like "Sharing the data" this can be a very simple decision. But in other cases, the conflicts between licenses are not limited to obvious differences and can come down to small details in the required attribution (Ministerium für Wirtschaft, Innovation, Digitalisierung und Energie des Landes Nordrhein-Westfalen & Beauftragte der Landesregierung für Informationstechnik (CIO) / Geschäftsstelle Open.NRW, 2019)[5]. Since these details can be easily overlooked, working with multiple licenses can be very challenging for inexperienced users and discourage them from working with certain datasets.

Additionally licenses can impose additional restrictions like the "Share-Alike"-condition, which we previously explained. In these cases just checking if the permissions and conditions of two license are compatible is not enough. This means, you also have to check if you can effectively relicense the data under one specific license, increasing the complexity of the process even more.

2.2.2 Documenting the Current Contents of the Pipeline

Finally there is the need to document the licenses connected to a project. To be able to quickly and effectively react to changes in licensing or the legalisation, it is important to always know which datasets and licenses are used in a project. This information can also be used to prove that all license-requirements are fulfilled and that the licensee is allowed to use the datasets for the intended purposes.

There are some things that can make this task more difficult. As mentioned before, in some cases the license-information for a datasets are harder to find. Another potential problem is that the amount of information can become unmaintainable over time. As research regarding open-source-licenses like (German et al., 2010) and (Wolter et al., 2023) has shown, the lists of licenses can grow exponentially over time, which can result in unmanageable license-constructs. Because there are no comparable studies regarding open data licenses, at least as far as we know, we cannot prove that this also applies to open data licenses. But since most datasets are continuously reused and combined, similarly to open-source software-packages, it is not unlikely that this will become a problem in the future. Both cases can make maintaining up-to-date information very difficult and easily can lead to missing some important license-change that breaks the license of your project.

[5] only available in German

2.3 State of the Art

2.3.1 License Descriptions

The idea to create alternate descriptions of licenses is not new. There are already a number of informal models, like the model used by GitHub for their ChooseALicense-platform[6]. The website provides short descriptions for open-source-licenses based on a limited set of properties. The idea behind this is to create simple to read descriptions that can help licensors and licensees to understand and compare licenses. But it lacks any tools to validate the license descriptions or compare two licenses which makes them unsuited for our framework.

A different approach to this are formal languages called Rights Expression Language (REL). These languages are defined as "a means of expressing use and access rights to assets"(Guth, 2003). This means they provide a vocabulary to describe the terms and conditions of a license and the syntax and semantics on how to use these descriptions. This can be used to create detailed license descriptions, but also enables users to validate these descriptions against the syntax and semantics of the language to ensure the descriptions are sound. Today there are mainly two RELs that are used in the context of license-information: the Open Digital Rights Language (ODRL) proposed by the World Wide Web Consortium (W3C) (W3C, 2018) and Creative Commons Rights Expression Language (ccREL) created by the CC-organization (Abelson et al., 2008). There also some more specialized versions like MPEG-21 for multimedia-information ('MPEG-21 Rights Expression Language', 2005), but they are rarely used. The biggest advantage of these languages is that descriptions created with them can guarantee validity. However they can be difficult to read and they lack any tools to automatically compare two or more licenses.

2.3.2 Automated License-Comparison and -Recommendation

There are already some solutions to automatically compare and handle license-recommendations. Some researchers have provided logic-based extensions to the mentioned RELs by transferring them into deontic or first-order logic. Examples of this are the logic described in Governatori, Lam, Rotolo, Villata and Gandon (2013), Governatori, Rotolo et al. (2013), Krötzsch and Speiser (2011) and Villata and Gandon (2012). They extend the vocabulary and semantics proposed by different RELs with rules that model the relationships between licenses (e.g. are permissions of one license prohibited by another license?). The goal for this approach is that this logic can be transferred to automated reasoners to check the compatibility between the licenses.

[6] https://choosealicense.com/

Ermilov and Pellegrini (2015) built on the research by Villata and Gandon (2012) and created a prototype for a License Compositor that can recommend licenses based on the incoming licenses (Ermilov & Pellegrini, 2015).

Gangadharan et al. (2007) propose a way to compose and compare service-licenses for web-services. It introduces an extension to the ODRL and a set of rules on how to compare and composite web-service-licenses. While its focus lies more on the side of web-service licenses, some of the concepts for the composition and comparison of licenses are kept very abstract, which means they could be transferred into other contexts (Gangadharan et al., 2007).

There are already some complete frameworks that build on this research. Cardellino et al. (2014) created the "LICENTIA"-tool [7]. It allows users to find a suiting license for a project or check if a license allows a specific use case. There is also the "LIVE License Checker" (LIVE: **Li**cense **Ve**rficiation) developed by Governatori et al. (2014), that can analyze license compatibility between datasets and vocabulary[8]. They are both based on a combination of the ODRL and ccREL and use the logic proposed by Governatori, Rotolo et al. (2013) and Villata and Gandon (2012) to check the compatibility between the licenses.

Moreau et al. (2019) propose the CaLi-framework (CaLi: **C**l**a**ssification of **Li**censes). It also uses the ODRL to model licenses. Its goal is to create an ordering of licenses by defining rules that model the restrictiveness (meaning how much you are allowed to do) of a licenses. By traversing over this ordering of different licenses, with the least restrictive license at the bottom and the most restrictive licenses at the top, it is then possible to check the compatibility between licenses. They then implemented it as a search-engine to search for datasets compatible with a specified license (Moreau et al., 2019).

Pellegrini et al. (2018) have created the DALICC-Framework (DALICC: **Da**ta **Li**cense **C**learance **C**enter) as a means to help people with composing and understanding licenses[9]. It also uses a combination of the ODRL and ccREL, but then extends it to create more detailed and comparable license descriptions (Pellegrini et al., 2018). It allows users to either compare a set of predefined license-descriptions or create their own license descriptions. The reasoning is done by applying a set of rules, defined in the semantic of Answer Set Programming (ASP)[10], to the license-descriptions. These rules (e.g. if one licenses prohibits an action it is not allowed to be part of the permission of another license) are then used to check if two licenses are compatible (Pellegrini et al., 2018).

Finally the European Commission provides a licensing assistant via its JOINUP

[7] http://licentia.inria.fr/
[8] https://www.eurecom.fr/~atemezin/licenseChecker/
[9] https://www.dalicc.net/
[10] https://github.com/dalicc/dalicc/blob/main/reasoner/app/programs/query.lp

platform called the "JLA" (Schmitz & Cacciaguerra Ranghieri, 2019)[11]. It is meant to help users find and compare licenses, by providing simplified license-information and allowing checks between them. In contrast to the previous examples it does not use a REL to model and compare its licenses. Instead it uses the following six categories to model a license:

- Permissions (what a license allows)
- Obligations (what a license requires)
- Prohibitions (what a license disallows)
- Compatibility (with what and how compatible is the license)
- Legal (how and by which legislation is the license backed)
- Support (are there communities or governments that back this license)

Each category consists of a list of predefined properties that can be set for a license (e.g. permits reproduction or prohibits holding someone liable). In addition to that each license-description is extended with meta-information like the license-name and the Software Package Data Exchange (SPDX)-identifier. Figure 2.1 shows an early concept of the resulting license-descriptions. Here the comparison of two licenses is not done by a automated reasoner, but instead done via a compatibility matrix that is curated by legal and data experts. Because of this the comparison does not return a binary result, but instead returns a short text that explains the result and hints at some of the important aspects (Schmitz & Cacciaguerra Ranghieri, 2019).

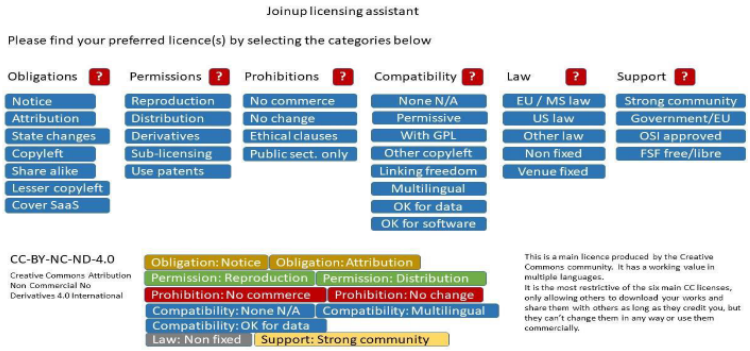


Figure 2.1: An early concept of the license-model implemented by the JLA (taken from: (Schmitz & Cacciaguerra Ranghieri, 2019))

[11] https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant

2.4 Drawbacks of the Existing Solutions

2.4.1 License Descriptions

As stated before there are currently two types of license-descriptions, we could use to model our licenses:

- An informal license-model like GitHub,
- A formal REL

Since the licenses-models are normally intended as a simple license-description they are specialized to match their specific context (e.g. the context of the GitHub-Model are open source-licenses). This means their vocabulary is limited to properties that fit this specific context. This makes reusing them very difficult, if they do not already match our context.

The RELs on the other hand are specifically built to be used to describe multiple different license-types. Because of this, their vocabulary usually includes a lot of attributes that have no connection to the topic of open data licenses, similar to the informal models. Additionally they do not include an option to effectively model the "Share-Alike"-condition. For example the ccREL defines the condition of "Share-Alike" as follows (Abelson et al., 2008):

> cc:ShareAlike - when redistributing derivative works of this work, using the same license.

This contradicts the definition that is used in the CC-licenses (see 2.1.3), since it does not model the case of similar licenses or different license-versions. All of this shows that, if we would like to reuse an existing model or REL, we would always have to adapt it to our context or extend it with additional definitions to be able to create good license-descriptions. In the case of reusing an existing REL, we would also have to condense the existing vocabulary to make sure the license-descriptions can only include information that is actually relevant when describing open data licenses.

So instead of reusing an existing model, it would be a better idea to create a new model that is built on top of these existing models and adds the missing capacities necessary to model open data licenses.

2.4.2 Automated License-Comparison and -Recommendation

There is also no perfect solution for the automated comparison of licenses. Gangadharan et al. (2007), Governatori, Lam, Rotolo, Villata and Gandon (2013), Governatori, Rotolo et al. (2013), Krötzsch and Speiser (2011) and Villata and Gandon (2012) cover different parts of the formal logic we need to reason over licenses, but lack

an implementation. This means, we always have to create an implementation if we want to integrate this into a pipeline. The existing frameworks, described in Cardellino et al. (2014), Governatori et al. (2014), Moreau et al. (2019) and Pellegrini et al. (2018), do provide an implementation for their logic. However none of them provide all the functionality we need at the moment. They either lack certain features (e.g. creating recommendations) or provide an implementation we cannot integrate into a data pipeline, which would require a port of the original implementation that fits the requirements of a tool like Jayvee.

Additionally all of these solutions use models that are intended to be able to handle multiple different license-types or are limited to only a short list of licenses. Because of this, we always have to extend their models to include all properties necessary to effectively model open data license or add missing licenses to their database. Aside from Krötzsch and Speiser (2011), they also do not have the capability to model the "Share-Alike" condition like we mentioned in the previous section, requiring additional modifications.

The two best solutions we identified are the DALICC- and CaLi-frameworks. The DALICC-framework misses a component to recommend a new license as a result of the framework. However it proposes a good architecture, extensive license-model and provides a REST-API[12] as a prototype, which could be connected to the pipeline via the internet. This would allow for easy license-compatibility-checks, but would need an additional extension to provide the recommendations we want.

The CaLi-framework does provide the logic to create a license-ordering, which can be used to check compatibility and create recommendations. But it only provides a limited license-model and a search-engine for datasets based on the defined logic. This is not suitable for our use-case.

Because of all these factors we decided to create a new framework specialized on open data licenses that combines the logic from the existing literature and frameworks. This framework also needs to include an implementation that can be included into an data pipeline.

[12] https://api.dalicc.net/docs

Table 2.1: Comparison of existing frameworks

Name	Prototype	Compatibility-Check	Recommendations?	ShareAlike?
Licentia	Website	Yes	No	No
LIVE	Javascript-Application	Yes	No	No
DALIIC	Website + API	Yes	No	No
CaLi	Search engine	Yes	Yes	No

3 Objective Definition

3.1 Requirements for a Solution

Based on these problems we identified two components, we need to be able to model and include open data licenses into data pipelines. First we need a model that can be used to represent multiple licenses in a standardized format, so they can be compared to each other. This model should also provide the option to include additional information and be capable to model special conditions like "Share-Alike".

The other component we need is an automated way to compare licenses. This component should be able to check for inconsistencies, create a license aggregate or create recommendations. As a final result it should also provide a human- and machine-readable report that can be attached to the result of a data pipeline.

3.2 Creating a Framework to Describe and Combine Licenses

As described in the previous chapter, many of the problems with open data licenses have to do with understanding the contents and combinations of different licenses. To help with these problems we created a framework that can automatically handle the comparison of open data licenses.

It gives providers and users of open data a common language to model licenses. To achieve this the framework includes definitions for the most common and important conditions and rights described in a license (e.g. on attribution or sharing data). These definitions are based on existing standards and legal terms, but also define new terms when necessary (e.g. there a multiple synonymous phrases). In both cases the framework contains an accurate description of each definition to help users understand them and/or resolve potential misunderstandings.

It is designed to allow an extension of these definitions to ensure missing terms or new legal concepts can be added when necessary. Providers and/or users will

then be able to map licenses to this framework, which will result in standardized license-descriptions that can be understood by everyone.

In addition to this model, the framework defines rules that can be used to perform compatibility checks between different licenses, aggregate them into a combined synthetic license and generate recommendations that show which existing licenses match this composite license. The compatibility checks are meant to detect potential risks when working with multiple licenses (e.g. which licenses are fundamentally incompatible?) and to help users understand them, while the other two functions should help users find a license they can use for their own work or communicate to data providers, why they cannot use datasets using certain licenses.

The definitions and rules will be provided as a set of definitions with detailed descriptions, explaining the expected inputs and the reasons for certain decisions we made. In addition to that, we also provide a formal notation of the functions as a Constraint-Satisfaction-Problem (CSP). This is done to give the framework a provable foundation, as well as provide another way for readers to understand its fundamentals.

This framework does not include a solution to extract the license-information from a dataset or attach them to it. This process is usually dependent on the implementation of a data pipeline. To ensure that our solution can be applied to multiple different architectures, we excluded this feature from our framework.

Finally it has to be explicitly said that the goal for this framework is not to provide binding legal advice. This is not possible without the knowledge, experience and additional context information available to a lawyer. The only goal for this framework is to provide a logic-based solution to compare and combine licenses.

3.3 Building the Tools to Integrate the Framework

To demonstrate our framework, we created a library that implements that provides the classes and functions necessary to use the framework in an application.

This tool enables users to create human-readable reports about the contents of their pipelines using the created description language or run the compatibility checks to find potential risks. To enable automation, the results of these tools are available in a machine-readable format like JavaScript Object Notation (JSON), CSV or Ressource Description Framework (RDF), that are common in the context of open data.

The tool is a standalone solution and does not require any external services to

run. This offline capability is necessary to ensure that the tool can be integrated into private and/or local data pipelines, that cannot have access to the internet.

The solution is open-sourced to invite others to contribute new ideas or make suggestions for improvements.

4 Solution Design

4.1 License Descriptions

As a base for our license-model, we use a derivative of the ODRL and ccREL. Both languages already provide a lot of the functionalities that we need and are broadly used in the existing literature. They both however also contain some functionalities that we do not need. Because of this, we extended them similar to the model used by the DALICC-framework (Pellegrini et al., 2018). This idea to combine both languages was first described by Cabrio et al. (2014), with the goal to create formal license descriptions, that can be used for further processing.

4.1.1 Creating the Basic License Descriptions

The ODRL is a REL, created as a policy expression language by the W3C to model "statements about usage of contents and services" (W3C, 2018). In the ODRL licenses are modeled as policies that consist of certain actions. The actions are categorized into three groups:

- **Permission**: A user is allowed to perform an action.
- **Prohibition**: A user is disallowed to perform an action.
- **Duty**: A user has to perform an action to conform to the policy.

Each permissions, prohibitions and duties can contain some constraints to model dependencies between actions, e.g. if you have to perform certain duties to enable permissions. This allows to create relationships between actions, that can later be used to compare and check these licenses. To better illustrate this concept figure 4.1 shows how the permission to distribute a dataset licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) would be modeled using the ODRL Information Model 2.2 provided by the W3C Permissions and Obligations Expression Working Group (2018). The dotted lines signal the blocks defined by the information model, while the solid lines indicate an action defined by the ODRL.

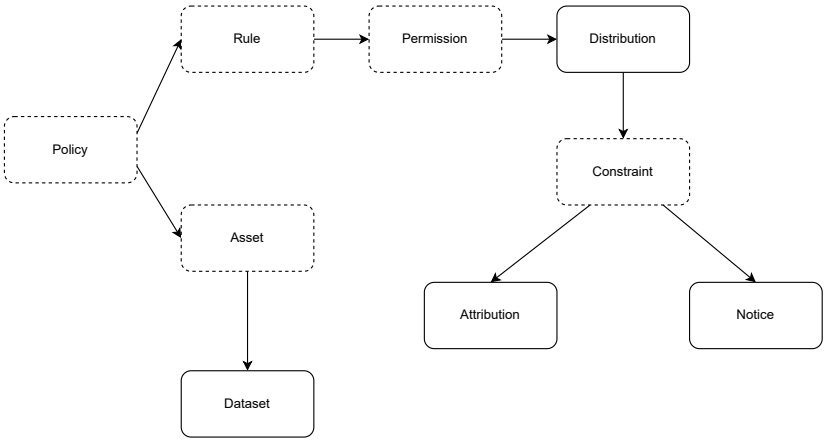


Figure 4.1: The distribute-permission of the license CC-BY-4.0, modeled using the ODRL information model

ccREL is another REL that was created by the CC-organisation to model their licenses. It also models a license as a collection of actions that are classified as **permits**, **prohibits**, **requires**. This is similar to the ODRL, but the actions are more specific to the CC-licenses. It also adds some terms to adds some meta-information, e.g. to model the deprecation of a license (Abelson et al., 2008).

The license-descriptions for our framework uses the three categories defined in the ODRL to model licenses, meaning each license consists of three distinct sets of permitted, prohibited and required actions. This categorization was chosen, because it is very flexible and can be extended very well with new actions. This was shown in Pellegrini et al. (2018). Additionally there is a lot of research available regarding the topics of license compatibility checking and composition that uses the vocabulary of the ODRL and ccREL. Since it will be easier to integrate and adapt this research into our framework, when we use the same base, we decided to use this language as the base for our framework.

However in contrast to the ODRL, our model does not include a way to define a specific constraint from one action to another action. This is instead solved by ensuring that the requirements for every actions are directly included into the set of required actions(= duties). This was done for two reasons: First it makes comparing and aggregating licenses easier. With this structure it is possible to compare the sets of different licenses, without having to also check the constraints of each action. This would not be a problem when comparing two licenses, but could become computationally difficult when combining more licenses. Secondly this ensures that we always consider every possible restriction when comparing two licenses. Since we do not have any context-information on how the contents of a dataset are used, we cannot know which permissions, prohibitions and duties apply to the current use-case. This means we must assume

the most-restrictive case to prevent classifying two incompatible licenses as compatible (=false-positive). Because of this, the added restrictiveness is actually helpful. Of course this can lead to some cases where two licenses are licensed as incompatible, even if they are technically compatible in the current use case (=false-negative). However this can be mitigated by adjusting the definitions of the available actions or including hints in the final report.

4.1.2 Defining the License Actions

To curate a list of allowed actions, we analyzed existing open data licenses. We used diversity sampling for this task, as it was described by Jansen (2010). The reason for this was that this method will result in a good representation for many licences, while keeping the number of actions relatively small. This is ideal, since a bigger model is more difficult to understand and makes comparing and aggregating two items more complex.

In total we looked at 19 different open data licenses. This list was curated from the most well-known open data licensing frameworks (CC and ODC) and other organisations like the Linux Foundation, since these license are used very often and are usually written to be globally applicable. We then extended this list with open data licenses backed by different European countries (e.g. UK, France, Germany). This was done to ensure that our model can describe and compare how different countries handle some of the legal aspects of open data licenses. Because of language barriers we could not include all licenses that we wanted, like the Italian Open Data License[1], that was only available in Italian. A final list of analyzed licenses can be seen in table 4.1.

During the analysis we split the licenses into multiple groups. These groups were then analyzed one after another with the goal of reaching theoretical saturation, meaning no new insights could be found by adding more licenses (Bowen, 2008). We started the process with just the CC-licenses. This was done since they are the most-common licenses, are easy to compare and the CC-organisation provides a lot of information about the meaning of each condition[2]. From this we added more licenses in groups of three until we reached saturation. In total we did five runs. Figure 4.2 shows how many changes were necessary per run.

[1] https://www.dati.gov.it/content/italian-open-data-license-v20
[2] https://wiki.creativecommons.org/wiki/License_Versions

Table 4.1: List of analyzed licenses

Name	Is Public Domain?	Is OD-Compliant?
CC-BY-4.0	No	Yes
CC-BY-SA-4.0	No	Yes
CC-BY-ND-4.0	No	No
CC-BY-NC-4.0	No	No
CC-BY-NC-SA-4.0	No	No
CC-BY-NC-ND-4.0	No	No
CC0	Yes	Yes
Open Data Commons-By-1.0	No	Yes
Open Data Commons-PDDL-1.0	Yes	No
Open Data Commons-ODbL-1.0	No	Yes
DL-DE-BY-2.0	No	No
DL-DE-Zero-2.0	No	No
Open Government License	No	Yes
Singapore Open Data License	No	No
Licence Ouverte	No	No
Norwegian Licence for Open Government Data (NLOD) 2.0	No	No
Open Use of Data Agreement	No	Yes
Community Data License Agreement - Permissive - Version 2.0	No	No
Community Data License Agreement - Sharing - Version 1.0	No	No

As you can see the number of gradually decreased during each run, until in the end there were no more changes necessary. During this run, we identified 26 different actions that could be used to describe the different licenses. They are categorized into two main groups:

- Permissions and Prohibitions
- Duties

The first group includes the actions to model what a licensee is allowed or not allowed to do. The second group includes actions that can be set as a requirement by a license. We made this decision, because it makes comparing two license simpler. If all actions could be defined in any of the three sets, we would always have to compare all three groups before making a decision. By limiting them two one of two groups, we can decrease the number of necessary checks. We also ensured that the two groups are distinct, meaning it is impossible to create a license, that either permits/prohibits and requires a certain at the same time.

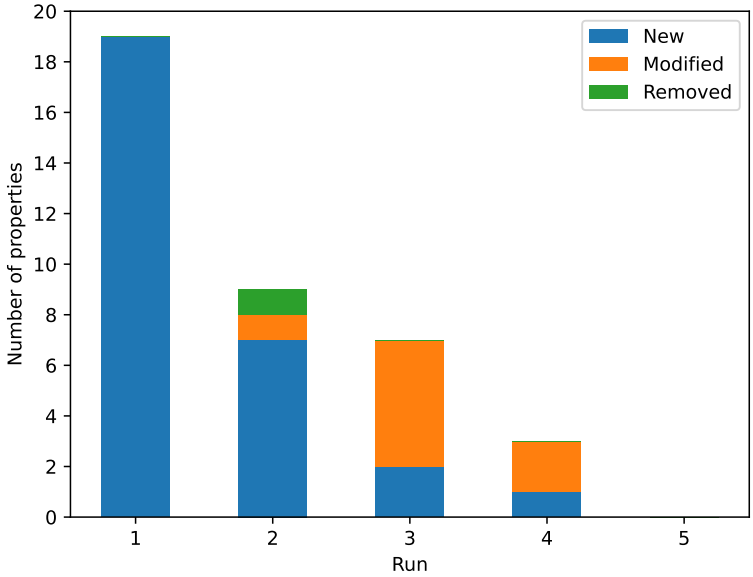


Figure 4.2: Number of changes per run

The final list of identified actions (along with a more detailed explanation of each property) is shown in the appendix (for Permissions/Prohibitions see Appendix A, for duties see Appendix B).

After identifying the necessary actions, we then adjusted them to closer resemble the vocabulary of the ODRL. The ODRL provides some definitions for common actions like e.g. sharing, modifying and distribution. It also contains some definitions from the ccREL in order to make the two languages more compatible. In cases where an action matched a definition from the ODRL-vocabulary[3], we renamed our actions to match the name from the vocabulary. If our definition matched two or more actions defined in the ODRL, we split the action into the different standard definitions to ensure compliance with the ODRL. The goal with this is that this compliance should help license-creators understand our model and potentially reuse their existing license-descriptions in our framework. If one our definitions matched multiple definitions in the ODRL, we tried to split the action into smaller definitions, so we could reuse the existing definitions. We then created new definitions for the actions, that we could not match to the ODRL. In case an identified action does not match any definitions in the ODRL or ccREL, we created our own action definition.

This process is similar to the one used by the DALICC-framework to streamline license-descriptions (Pellegrini et al., 2018). The reason, why we did not reuse the

[3]https://www.w3.org/TR/odrl-vocab/

DALICC-vocabulary[4], was that it and our list of actions were too different. Our list was not completely covered by the DALICC-vocabulary. Their vocabulary also includes many actions that we do not need. Because of this, if we wanted to reuse their vocabulary, we still would have to create our own definitions for these actions to extend their model, while keeping all the unused definitions to ensure backwards-compatibility. Since this would unnecessarily increase the size our framework, we decided to only create our own definitions, based on the ODRL and ccREL. This ensured that our definitions exactly fitted our intended purposes and also allowed us to fit the actions to our logic when necessary.

Overall this process resulted in a list of 28 possible actions that could be part of a license. Only 6 of these actions were taken from the combined ODRL- and ccREL-vocabulary. The reason for this was that some of their definitions summarized some actions, we previously defined as separate steps. Since we did not want to lose this flexibility, we then decide not to use the existing definitions. This resulted in 22 new definitions. Each action is independent from all other actions, meaning that a license-description always has to include all the permitted, prohibited and required actions. The full list of actions, including the definitions and the origins of the definitions, can be found in Appendix C.

4.1.3 Including Meta-Information

In addition to the permissions, prohibitions and duties, we also need to store some meta-information about each license in our model. This includes details like identifiers for a license, a link to the original source or if it complies with the Open Definition (OD)[5].

First we need some way to model the different identifiers for each license. This information is necessary to reliably identify the licenses. They are either provided by the license-creators (e.g. the full name of the license), but in many cases there are also standardized identifiers from a third-party that are used to reference them. Because of this we decided to not only include the full name of a license, but also include two third-party identifiers to show how they can be integrated into our model. The two third-party identifiers, we chose, are:

- The SPDX-identifiers[6] as an example for a global standard.
- The identifiers used in the German adaptation of the Data Catalogue Application Profile (DCAT-AP)-specification[7] as an example for a standard with limited range.

[4]https://docs.dalicc.net/
[5]https://opendefinition.org/od/2.1/en/
[6]https://spdx.org/licenses/
[7]https://www.dcat-ap.de/def/licenses/

We treat each identifier as one property of the meta-information. If a license does not have one of these licenses, this property is left empty. This way we can find a license-description by checking if the identifier provided by the original dataset matches any of the values from one of the properties. If no variable matches the identifier, we can assume that we do not know the license. It can also be used in the final report to provide additional information to the user. An alternative to this solution would be to model the identifiers as a list, containing all the known identifiers. This solution would reduce the number of properties we would need to model for the meta-information, but it would also result in the loss of any context information (e.g. which third-party-provider maintains a specific value?). Since we want to be able to provide this kind of information when searching for a license or when generating our final reports, this solution is not suitable for our model.

Aside from the identifiers, there is also a need to model other meta-information. Examples for this are classifiers, like if a license is considered as a public-domain-license or if it is compliant with the OD or a link to the original version of the license. In the same way as before, these informations are modeled as independent properties, each with a predefined set of possible values, either taken from a specification or derived from available information.

At the moment, this information is not needed for the reasoning we are going to explain later. We nonetheless included this information into our model to show how additional meta-information can be added to the model, which in turn could be used to create an extension to our logic.

4.1.4 Model Share-Alike

To complete our model we need a way to describe the "Share-Alike"-condition. As we previously described "Share-Alike" is a condition that restricts which licenses can be used to relicense a dataset (Yi-Hsuan Lin et al., 2006) and is defined as:

- only the current license can be used to license a derivative dataset, or
- you can use one of a multiple different licenses that are compatible with the current one.

These different definitions can also observed in the available research that deals with the topic of modeling "Share-Alike". Krötzsch and Speiser (2011) mention the restrictive definitions in the CC-licenses, but try to model it more freely by using a definition that classifies a license as similar if it uses the same content (meaning permissions, prohibitions and duties) (Krötzsch & Speiser, 2011). On the other hand Cabrio et. al define "Share-Alike" as a duty that limits the derivative to the same license, as it is defined in the ODbL(Cabrio et al., 2014).

In our case, we model the "Share Alike"-condition as an action that can be set as a

duty, similar to the way it was proposed by Cabrio et al. (2014), in combination with a list containing the compatible licenses. This was necessary to give us the ability to model multiple compatible licenses for a single license, without having to use a content-based approach like Krötzsch and Speiser (2011). We initially thought about integrating their content-based approach into our model, but could not use it, because this could lead to license-recommendations that are not officially supported. Since this would violate the terms of the CC-licenses, we deemed this approach not suited for our use-case. Instead we used the list-based approach that Krötzsch and Speiser (2011) proposed as an alternative solution to their content-based approach. They discarded the idea, since, in their opinion, it would not fit the idea of "Share-Alike" that was initially proposed for the CC-licenses. For our use-case however, it fit perfectly, since this meant we could limit the recommended licenses to either one license or a list of multiple licenses.

4.2 Architecture of the Framework

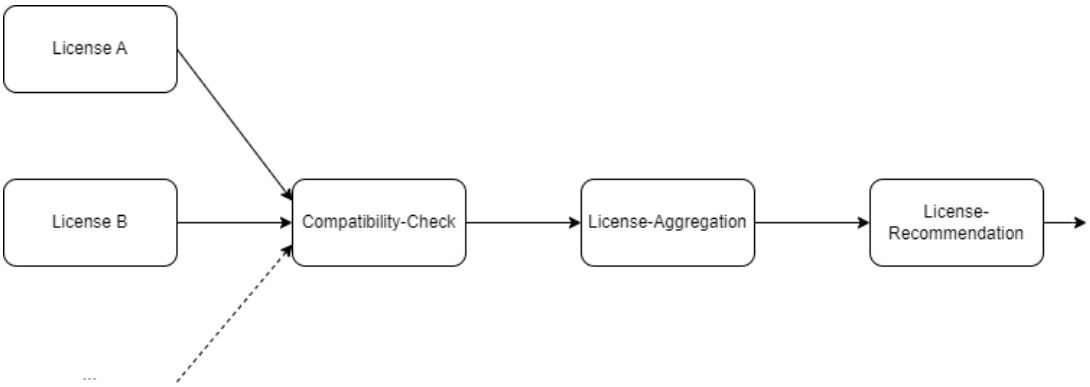


Figure 4.3: Overview over the framework components

As mentioned during the objectives, the framework consists of a model to describe licenses and the logic to automatically compare and aggregate them. When designing the pipeline for our framework, we made the following assumptions:

- Each license represents a separate dataset.
- A license is represented by an identifier, its permissions, prohibitions and duties and a list of compatible licenses.

We made these assumptions based on the previously described use-case. Each pipeline provides the license for its corresponding dataset, which in turn is represented by our internal representation. The framework then operates on this internal representation and returns a result for all licenses.

The logic itself is separated into three separate components for the compatibility-checks, license-aggregation and license-recommendation. This modular design is inspired by the architecture of the DALICC-framework (Pellegrini et al., 2018). The advantage of this is that the components could be used stand-alone or connected to each other to create a pipeline that handles the complete process. This pipeline is describe in figure 4.3. The output of this pipeline would then include the result of the compatibility-check, the aggregated license and the license-recommendations.

4.3 License-Compatibility-Checking

Before composing two licenses, it is mandatory to check if the licenses in question are compatible with each other. In our case, two licenses are "compatible with each other" if they can be composed into a synthetic license without violating a prohibition or duty of one of the original licenses.

We identified two methods to ensure that we can create a composite license:

- One license is less or equally restrictive as the other license, or
- Both licenses have some common permissions and allow switching the combined derivative to another license.

The first case can be checked by comparing the permissions, prohibitions and duties of the two licenses. The three sets are distinct for each license (R0). A license A can be classified as less restrictive than a license B, if it fulfills the following rules:

- The permissions of license A are a superset of or equal to the permissions of license B and are not prohibited by license B (R1).
- The prohibitions of license A are a subset of or equal to the prohibitions of a license B and not permitted by license B (R2).
- The duties of license A are a subset of or equal to the duties of a license B (R3).
- License A allows to change the license and License B conforms to "Share-Alike" (R4).

This rule-set is derived from Moreau et al. (2019). If it holds, we have a guarantee that the two licenses can be composed, since the composite of both licenses will be equal to the more restrictive license. We added the last rule, to ensure that we are allowed to change the licensing from the less-restrictive license A to the more restrictive license B. Without this, even if one license is more restrictive license than the other, a licensee is not allowed to change the license.

To formalize this check, we can model it as a CSP. This means we can describe the problem as a set of variables and constraints. Each variable contains a value from a predefined domain. The problem can be solved if all variables can be set to a value, while fulfilling the defined constraints (Russell & Norvig, 2022).

In our case a license is represented by six variables modeling an identifier for the license, its permissions, prohibitions and duties, the list of compatible licenses (="Share-Alikes") and a variable containing a five-tuple representing the valid licenses.

$$\begin{aligned} V = \{&I_a, I_b \text{ (Identifiers)} \\ &Pe_a, Pe_b \text{ (Permissions)}, \\ &Pr_a, Pr_b \text{ (Prohibitions)}, \\ &R_a, R_b \text{ (Duties)}, \\ &S_a, S_b \text{ (Share-Alikes)}, \\ &L_a, L_b \text{ (Licenses)}\} \end{aligned} \tag{4.1}$$

The domains for the variables are defined as a set of integers for the identifiers, three sets containing the sets of possible combinations of actions (permissions, prohibitions and duties), a set representing the sets of compatible licenses ("Share-Alike") and a set defining the valid combinations of these sets.

$$\begin{aligned} D_{I_a}, D_{I_b} =& \{0, 1, 2, \ \dots \ , 18\} \\ D_{Pe_a}, D_{Pe_b} =& \{\{"Sharing", "DerivativeWorks", \ \dots \ \}, \ \dots \ \}, \\ D_{Pr_a}, D_{Pr_b} =& \{\{"ClaimWarranty", "ClaimLiability", \ \dots \ \}, \ \dots \ \}, \\ D_{R_a}, D_{R_b} =& \{\{"Attribution", "Inform", \ \dots \ \}, \ \dots \ \} \\ D_{S_a}, D_{S_b} =& \{\{\}, \{1\}, \{4\}, \ \dots \ \} \\ D_{L_a}, D_{L_b} =& \{\langle 0, \{"Sharing", "DerivativeWorks", \ \dots \ \}, \\ & \{"ClaimWarranty", "ClaimLiability", \ \dots \ \}, \\ & \{"Attribution", "Inform", \ \dots \ \}, \\ & \{\}\rangle, \ \dots \ \} \end{aligned} \tag{4.2}$$

The rules we introduced earlier can then be modeled as constraints restricting the space of possible solution. If we find a solution that fulfills these constraints, we can assume that license A is less compatible than license B.

$$
\begin{aligned}
\text{R0a: } & (Pe_a \cap Pr_a = \{\}) \wedge (Pe_a \cap R_a = \{\}) \wedge (Pr_a \cap R_a = \{\}) \\
\text{R0b: } & (Pe_b \cap Pr_b = \{\}) \wedge (Pe_b \cap R_b = \{\}) \wedge (Pr_b \cap R_b = \{\}) \\
\text{R1: } & Pe_a \supseteq Pe_b \\
\text{R2: } & Pr_a \subseteq Pr_b \\
\text{R3: } & R_a \subseteq R_b \\
\text{R4a: } & \text{"}ChangeLicense\text{"} \in Pe_a \\
\text{R4b: } & \text{"}ShareAlike\text{"} \notin R_a \vee \\
& (\text{"}ShareAlike\text{"} \in R_a \wedge I_b \in S_a), \\
\text{License A: } & L_a = \langle l_0, l_1, l_2, l_3, l_4 \rangle \text{ with} \\
& l_0 = I_a, l_1 = Pe_a, l_2 = Pr_a, l_3 = R_a \text{ and } l_4 = S_a \\
\text{License B: } & L_b = \langle l_0, l_1, l_2, l_3, l_4 \rangle \text{ with} \\
& l_0 = I_b, l_1 = Pe_b, l_2 = Pr_b, l_3 = R_b \text{ and } l_4 = S_b
\end{aligned}
\tag{4.3}
$$

The other check is necessary to check compatibility between licenses that are not classified as less-restrictive. This can be the case if both licenses have some permissions, prohibitions or duties in common without each other, without one being clearly less restrictive than the other. If this is the case, the resulting composite does not match any of the previous two licenses. But there could potentially be a third license that does match the necessary requirements. In this case we still want to make sure that the composite can be a valid license, but we have to use a different rule-set from before:

- Both licenses have at least one common permission that is not prohibited by the other license (R5).
- Both licenses allow to change the license (R6).

The purpose of these rules is to ensure that we do not combine licenses that lead to "empty" licenses without permissions. This license would not be usable. Since this also will not result in one of the two original license, both licenses need to at least allow a change to another license for the derivative.

Again we can model this as a CSP. We are going to reuse the variables and domains from the previous CSP and only define a new set of constraints:

$$
\begin{aligned}
\text{R0a: } & (Pe_a \cap Pr_a = \{\}) \wedge (Pe_a \cap R_a = \{\}) \wedge (Pr_a \cap R_a = \{\}) \\
\text{R0b: } & (Pe_b \cap Pr_b = \{\}) \wedge (Pe_b \cap R_b = \{\}) \wedge (Pr_b \cap R_b = \{\}) \\
\text{R5: } & Pe_a \cap Pe_b \neq \{\} \\
\text{R6a: } & \text{"}ChangeLicense\text{"} \in Pe_a \\
\text{R6b: } & \text{"}ChangeLicense\text{"} \in Pe_b \\
\text{License A: } & L_a = \langle l_0, l_1, l_2, l_3, l_4 \rangle \text{ with} \\
& l_0 = I_a, l_1 = Pe_a, l_2 = Pr_a, l_3 = R_a \text{ and } l_4 = S_a \\
\text{License B: } & L_b = \langle l_0, l_1, l_2, l_3, l_4 \rangle \text{ with} \\
& l_0 = I_b, l_1 = Pe_b, l_2 = Pr_b, l_3 = R_b \text{ and } l_4 = S_b
\end{aligned}
\tag{4.4}
$$

In both cases the licenses have to allow creating a derivative of the original dataset. This is necessary, because a license in our framework represents a single dataset. Since combining two datasets counts as creating a derivative, we have to classify the licenses as incompatible (R7). To model this requirements, we add the following constraints to the previous problems:

$$
\begin{aligned}
\text{R7a: } & \text{"}DerivativeWorks\text{"} \in Pe_a \\
\text{R7b: } & \text{"}DerivativeWorks\text{"} \in Pe_b
\end{aligned}
\tag{4.5}
$$

To help illustrate the two CSPs, the appendix includes two examples showing a solution using the CC-BY-4.0 and CC-BY-SA-4.0 (see Appendix D and E). Figure 4.4 illustrates the whole process of checking the compatibility between licenses. In total a license has to be either more- or less-restrictive than the other license or show that it there is a chance to create a valid composite license (following the previous rule set). If both of these checks fail, the licenses are not compatible with each other and cannot be composed into a single valid license. If it holds true, we can continue with the license-composition and -recommendations.

4.4 License-Aggregation

To create synthetic licenses we need a way to combine two or more licenses into a single dataset. For this we use a solution that is proposed in Gangadharan et al. (2007), Governatori, Lam, Rotolo, Villata and Gandon (2013), Governatori, Rotolo et al. (2013) and Villata and Gandon (2012). Gangadharan et al. (2007) checks and composes licenses by utilizing AND- and OR-composition (Gangadharan et al., 2007). Similar to this Villata and Gandon (2012) presented the following three solutions that could be used to combine licenses:

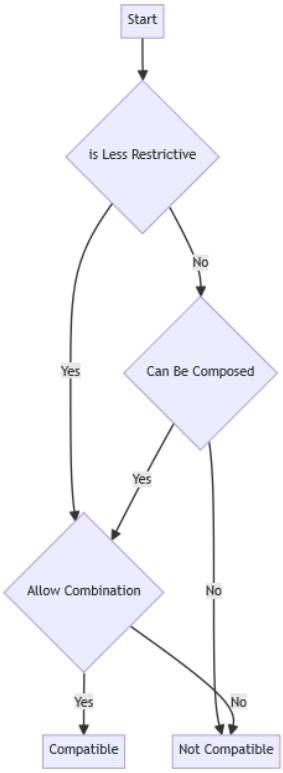


Figure 4.4: Flow-Chart for the process of checking the compatibility between two licenses

- OR-composition: If one license contains an obligation, it becomes part of the aggregated license.
- AND-composition: An obligation only becomes part of an aggregate if it is included in all licenses.
- Constraining-value: The most restraining obligation of two clauses is transferred to the composition.

Governatori, Lam, Rotolo, Villata and Gandon (2013) and Governatori, Rotolo et al. (2013) proposes an extension for this logic, which only focuses on the first two heuristics. To combine the permissions of two or more licenses, the AND-composition is used. This automatically ensures only permissions that are included in all licenses are contained in the synthetic license. The prohibitions and duties are composed using the OR-composition. The reason for this is, that in case an action is not allowed by one licenses, this prohibition has to transfer to the composite. The same holds true for the duties. If something is required by one license, it has to be required by the composite. Otherwise it would result in a conflict with the original license. This ensures that the composite license is at least as restrictive as the original licenses (Governatori, Rotolo et al., 2013).

We are not composing the list of compatible licenses, since these are only representative for the current license.

Our framework is not the first to use this approach. Similar to our framework, both the Licentia- and the LIVE-framework use the first two compositions for their compatibility-checks. They propose to use the AND-composition for the composition of license-permissions and to check the validity of the combination of license-permission and -prohibitions. The OR-composition is then used to compose the license-duties. This results in a final set of obligations that is considered the composition of two licenses. (Cardellino et al., 2014; Governatori et al., 2014).

To illustrate this process, we are going to give an example using the two licenses CC-BY-4.0 and Creative Commons Attribution Non-Commercial 4.0 International (CC-BY-NC-4.0). The permissions, prohibitions and duties are each represented as a variable containing the set of corresponding actions. They are then composed following the explained rules to create three new sets representing the composed permissions, prohibitions and duties. A visualization of this process can be found in the appendix (see Appendix G)

1. $Pe_c = Pe_{cc-by} \cap Pe_{cc-by-nc}$
$= \{"Sharing", "DerivativeWorks", "CommercialUse",$
$"Reproduction", "Distribution", "FreeAccess",$
$"UninhibitedAccess", "ChangeLicense"\} \cap$
$\{"Sharing", "DerivativeWorks", Reproduction",$
$"Distribution", "FreeAccess", "UninhibitedAccess", "ChangeLicense"\}$
$= \{"Sharing", "DerivativeWorks", "Reproduction", "Distribution",$
$"FreeAccess", "UninhibitedAccess", "ChangeLicense"\}$

2. $Pe_c = Pr_{cc-by} \cup Pr_{cc-by-nc}$
$= \{"ClaimWarranty", "ClaimLiability", "Endorse",$
$"ClaimMoralRights", "ClaimTrademark", "ClaimPersonalityRights",$
$"ClaimPatentRights", "ClaimCopyrightRights", "Relicense"\} \cup$
$\{"CommercialUse", "ClaimWarranty", "ClaimLiability",$
$"Endorse", "ClaimMoralRights", "ClaimTrademark",$
$"ClaimPersonalityRights", "ClaimPatentRights",$
$"ClaimCopyrightRights", "Relicense"\}$
$= \{"CommercialUse", "ClaimWarranty", "ClaimLiability",$
$"Endorse", "ClaimMoralRights", "ClaimTrademark",$
$"ClaimPersonalityRights", "ClaimPatentRights",$
$"ClaimCopyrightRights", "Relicense"\}$

3. $R_c = R_{cc-by} \cup R_{cc-by-nc}$
$= \{"Attribution", "Inform", "Notice", "LinkLicense", "LinkDataSet"\}$
$\cup \{"Attribution", "Inform", "Notice", "LinkLicense", "LinkDataSet"\}$
$= \{"Attribution", "Inform", "Notice", "LinkLicense", "LinkDataSet"\}$

(4.6)

The advantage of using this composition in combination with our proposed license-model is that it automatically returns a valid composite license. Because of the usage of AND- and OR- composition, the composite cannot include any duplicates for permissions and prohibitions, since the permissions, prohibitions and duties of a license are distinct. This means the permissions are only included, if they are permitted by all licenses. If another license either does not mention a permission or does prohibit it, the conflicting permissions are automatically removed from the composite. On the other hand, a prohibition is always included,

even if it is only mentioned once, meaning no information is lost. The duties are automatically distinct and do not require sanitation, since they cannot include any actions from the other two sets.

Other papers or frameworks solve this by introducing additional constraints on their composite to ensure their validity. Villata and Gandon (2012) for example solves this conflict, by introducing relationships between the three categories and between the actions themselves. These relationships are then enforced and used during license-composition to create valid license-descriptions, automatically removing invalid actions from the finished license.

4.5 License Recommendation

The final part of our framework is to provide a reliable solution to create recommendations, which licenses could be used to license the created derivative. We do this by comparing the created license-aggregates to the list of existing licenses. Here we use a set of rules, that is similar to the compatibility-checks. A license is recommended if it is equally or more restrictive than the generated license-aggregate.

To check this, we again use a rule-set based on the principles described in Moreau et al. (2019).

- The permissions of the composite-license are a superset or equal to the permissions of a license (R8).
- The prohibitions of the composite-license are a subset or equal to the prohibitions of a license and not permitted by the license (R9).
- The duties of the composite-license are a subset or equal to the duties of a license (R10).
- The original licenses contained in the composite allow the change to the recommended license (R11).

The first three rules are the same rules we used to check the license-compatibility. The reason for the decision to limit the recommendations to equal or more restrictive licenses is to ensure that we only recommend licenses, that definitely fulfill all the necessary prohibitions and duties.

We also considered including less-restrictive licenses into our recommendations and show what parts of the composite-license are missing from the license. This could be useful for researchers to find out what prevents them from reusing a certain license or if they could license the datasets by limiting its use-case to specific purposes. But since we currently do not include context-information into our process, we have no way to weigh the importance of different actions. This

means, we cannot measure how much less-restrictive a license is, if it e.g. only misses one prohibition or duty. Since this could potentially lead to invalid recommendations, we decided not include these recommendations, but it is certainly something that could be part of future works.

To check the final rule, we have to validate if one or more of the original licenses, specify a restriction for the derivative-licenses. This can be done by checking the duties of the original licenses and if necessary compare if the target-license is included in the list of allowed licenses.

Again, we can model this as a CSP. For this we need the same variables we used for the compatibility-checks, but also have to add five additional variables. One variable represents the licenses included in the composite and three variables model the permissions, prohibitions and duties of the composite. Finally we need a variable that can represent the composite license.

$$
\begin{aligned}
V = \{ & I_0, ..., I_n, I_r \text{ (Identifiers)}, \\
& Pe_0, ..., Pe_n, Pe_r \text{ (Permissions)}, \\
& Pr_0, ..., Pr_n, Pr_r \text{ (Prohibitions)}, \\
& R_0, ..., D_n, D_r \text{ (Duties)}, \\
& S_0, ..., S_n, S_r \text{ (Share-Alikes)}, \\
& L_0, ..., L_n, L_r \text{ (Licenses)}, \\
& Inc \text{ (Included license)}, \\
& CPe \text{ (Composite-Permissions)}, \\
& CPr \text{ (Composite-Prohibitions)}, \\
& CR \text{ (Composite-Duties)}, \\
& C \text{ (Composite-License)}\} \\
& \text{with n} \in [0, ..., 18]
\end{aligned}
\tag{4.7}
$$

The reason, why we need to add three new variables to model the license-actions, is that they have a different domain from the existing domains for the permissions, prohibitions and duties. Their domain models the sets of actions that can be created by the composition, which are a combination of the sets defined in D_{Pe}, D_{Pr} and D_R.

$$
\begin{aligned}
D_{I_n}, D_{I_r} =& \{0, 1, 2, ..., 18\} \text{ with n} \in [0, \ ... \ , 18] \\
D_{Pe_n}; D_{Pe_r} =& \{\{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \ ... \ \}, \ ... \ \}, \\
& \text{with n} \in [0, ..., 18] \\
D_{Pr_n}, D_{Pr_r} =& \{\{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \ ... \ \}, \ ... \ \}, \\
& \text{with n} \in [0, \ ... \ , 18] \\
D_{R_n}, D_{R_r} =& \{\{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \ ... \ \}, \ ... \ \} \\
& \text{with n} \in [0, ..., 18] \\
D_{S_n}, D_{S_r} =& \{\{\}, \{1\}, \{4\}, \ ... \ \} \\
& \text{with n} \in [0, \ ... \ , 18] \\
D_{L_n}, D_{L_r} =& \{\langle 0, \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \ ... \ \}, \\
& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \ ... \ \}, \\
& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \ ... \ \}, \{\}\rangle, \ ... \ \} \\
& \text{with n} \in [0, \ ... \ , 18] \\
D_{Inc} =& \{(0, 0), (0, 1), \ ... \ \}, \\
D_{CPe} =& \{\{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \ ... \ \}, \ ... \ \}, \\
D_{CPr} =& \{\{\}, \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \ ... \ \}, \ ... \ \}, \\
D_{CR} =& \{\{\}, \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \ ... \ \}, \ ... \ \} \\
D_C =& \{\langle (0, 0), \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \ ... \ \}, \\
& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \ ... \ \}, \\
& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \ ... \ \}, \{\}\rangle, \ ... \\
& \}
\end{aligned}
\tag{4.8}
$$

The rules we described earlier can then be modeled using the following constraints:

$$
\begin{aligned}
\text{R0a: } & (CPe \cap CPr = \{\}) \wedge (CPe \cap CR = \{\}) \wedge (CPr \cap CR = \{\}) \\
\text{R0b: } & (Pe_r \cap Pr_r = \{\}) \wedge (Pe_r \cap R_r = \{\}) \wedge (Pr_r \cap R_r = \{\}) \\
\text{R8: } & CPe \supseteq Pe_r \\
\text{R9: } & CPr \subseteq Pr_r \\
\text{R10: } & CR \subseteq R_r \\
\text{R11: } & \forall x \in Inc, L_x = \langle l_0, l_1, l_2, l_3, l_4 \rangle \\
& \text{with } l_0 = x \wedge \text{"}ChangeLicense\text{"} \in l_1 \wedge \\
& (\text{"}ShareAlike\text{"} \notin l_3 \vee (\text{"}ShareAlike\text{"} \in l_3 \cap I_r \in l_4) \\
\text{Recommendation: } & L_r = \langle l_0, l_1, l_2, l_3, l_4 \rangle \text{ with} \\
& l_0 = I_r, l_1 = Pe_r, l_2 = Pr_r, l_3 = R_r \text{ and } l_4 = S_r \\
\text{Composite: } & C = \langle c_0, c_1, c_2, c_3 \rangle \text{ with} \\
& c_0 = Inc, c_1 = CPe, c_2 = CPr \text{ and } c_3 = CR
\end{aligned}
\tag{4.9}
$$

If we find a solution to this problem, we can recommend the license that is modeled in L_r. If we do not find a solution, there is no license we can recommend and we have to end the pipeline with no license-recommendation. Again to help understand this CSP, we provide an example using the CC-BY-4.0 and CC-BY-SA-4.0 in the appendix (see Appendix F). In this example, the composite license consists of a combination of both licenses and it is checked if either of the two licenses are suitable as a recommendation.

5 Implementation

To make the framework applicable to be used in a data pipeline, we also created a software-implementation for it. This includes a license-database that stores the licenses and a core-library, that implements the previously described framework. The code of the library is released as an open-source-project licensed under the license "Apache-2.0" (Philip Rebbe, 2024). It includes an automatically generated Software Bill-of-Materials (SBOM), referencing the used open-source-libraries following the "SPDX specification 2.3"[1]. This was done using the GitHub-API[2]. A short summary of the used npm-packages is also included in Appendix I.

5.1 The License Database

The first part is the license-database. In it we need to store the license-descriptions for our analyzed licenses in addition to detailed information about the possible actions.

We chose to store the license-descriptions and actions using multiple JSON-files. Each license and action is describe in a single JSON-file. We chose JSON as a file format, since it is both human- and machine-readable, is very resource-efficient and can be integrated into many different programming languages very well. It also fits our data-model very well, as it allows us to model the permissions, prohibitions and duties very well.

During development we also considered using an SQLite-database[3] to store the information, but ultimately decided against it. Using SQLite would make distributing the library and database, e.g. as a npm-package, more difficult. We would either have to include the entire database into the package, which will dramatically increase the size of the package or regularly provide update scripts, that then have to be separately run on the users local database. A relational

[1] https://spdx.github.io/spdx-spec/v2.3/

[2] https://docs.github.com/en/code-security/supply-chain-security/understanding-your-software-supply-chain/exporting-a-software-bill-of-materials-for-your-repository

[3] https://www.sqlite.org/

```
{
    "metaInformation": {
        "id": 0,
        "name": "Creative Commons Attribution License
        ↪  4.0",
        "spdxName": "CC-BY-4.0",
        "dcatName": "cc-by/4.0",
        "sourceLink": "https://creativecommons.org/
        ↪  licenses/by/4.0/legalcode.de",
        "description": null
    },
    "permissions": [ 0, 1, 2, 3, 4, 15, 16, 27 ],
    "prohibitions": [ 12, 13, 14, 17, 19, 21, 22, 23, 28
    ↪  ],
    "duties": [ 6, 7, 8, 9, 10 ],
    "shareAlikes": []
}
```

Listing 1: Overview over the framework components

database also does not match our data-model very well, since each license can have different amounts of permissions, prohibitions and duties.

We also considered to use other formats like JSON for Linked Data (JSON-LD)[4] or RDF[5]. However we found them to be too complex for our use-case, as we do not need to model any constraints in our descriptions. It also makes adding new licenses more difficult, since both formats have much stricter syntax-rules than a simple JSON-file.

In total there are 19 separate files containing the license-descriptions and 28 files containing the action-description. Each license-file includes all the meta-information, permissions, prohibitions, duties and "Share-Alike"-information, as described in the framework. The action-descriptions contains the name, a more detailed version of the name and the definition for each action.

In the framework, the licenses and actions are linked together by using an identifier. Since using a string as an identifier always carries the risk of missing a reference because of a typing error, we again added a unique numeric identifier to each file to make references between them easier. This was done by adding a new property to the meta-information of the licenses, since it semantically fits there, and by extending the action-description with another parameter. The identifiers for licenses and actions are independent from each other and always start at 0.

[4]https://json-ld.org/
[5]https://www.w3.org/RDF/

It is then incremented by 1 for each new license/action.

Listings 4 and 2 show an example for contents of the license- and action-files.

```
{
    "id": 0,
    "name": "Sharing",
    "displayName": "Share the original data",
    "description": "Sharing the original data without
    ↪  modification"
}
```

Listing 2: JSON-representations of an action

The files are stored in one directory called "data" and then split into the two subdirectories "actions" and "licenses". Each files is named after the assigned id and the license or action it represents. This way a user can either search for the name of a license or the identifier. Putting the identifier first also results in an ordering for the files, making manual searches for a specific file a lot easier.

```
data
├── actions
│   ├── 0_sharing.json
│   ├── 1_derivativeWorks.json
│   └── ...
└── licenses
    ├── 00_CC-BY-4-0.json
    ├── 01_CC-BY-SA-4-0.json
    └── ...
```

Figure 5.1: An overview of the structure of each license

5.2 The Core Library

The other half of the reference implementation is the core library. It is written using Typescript[6] and implements the logic of our proposed framework. We chose Typescript, since it is a simple and popular programming language that can be easily extended when necessary.

We considered using logic programming languages like Prolog (specifically SWI-Prolog[7]) or Logica[8], but decided against it. They have the advantage that they

[6] https://www.typescriptlang.org/
[7] https://www.swi-prolog.org/
[8] https://logica.dev/

are optimized to be used to implement logic functions like the ones we explained in the previous chapter. Their syntax is also similar to the notation we previously used to represent our rules, making the transfer of bigger statements very easy.

However they are not as well-known as a all-purpose programming language like Typescript, making our reference implementation less approachable for someone who has never worked with them. Additionally our logic functions are not very complex and have a limited scope, meaning we do not need an optimized programming language to recreate them. Since this library is also supposed to be a reference-implementation, others can use to learn about our framework, we decided to use the more approachable programming language. In addition to that, the Jayvee-project is also implemented using Typescript, making a later integration of the framework a lot easier.

The core library is implemented using a layered architecture and consists of the following components:

- A data-access layer that connects the library to the license-database.
- A business-layer that implements the framework.
- An interface-layer that makes the business-logic available to a consumer.

The business-layer is split into the different core-components of the license-framework to keep a separation of concern and keep the code-base manageable. Figure 5.2 shows the overall architecture, including the dependencies between the different components. The appendix also includes a complete class-diagram showing all implemented classes (see Appendix H).

5.2.1 Data-Access

The data-access-layer provides the connection from the core-library to the previously described license-database. It only provides the function `getLicenses`, that returns the stored licenses to the caller. To do so, the different JSON-files, describing the actions and licenses, are read using the default Typescript file-system-library[9]. It then deserializes all of them into a list of objects, that can then be used to run the compatibility-checks and aggregation logic on them. The types and classes for this are provided by the library.

Each license is represented as an object of type `License` with properties describing the different parts of our license model. An overview of the base entities is shown in figure 6.4.

At this moment the files are read once during initialization and only a copy of the stored values is returned to the caller. This is done since we do not expect the files

[9]https://nodejs.org/api/fs.html

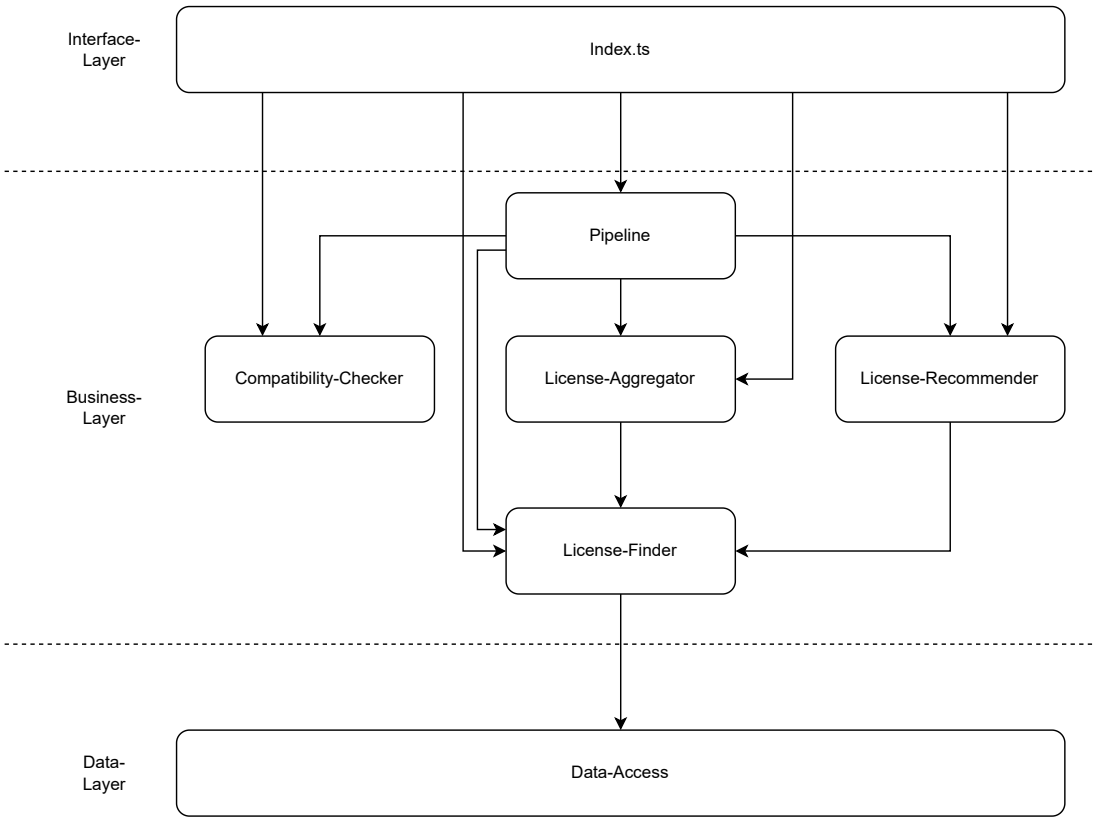


Figure 5.2: Overview over the architecture of the core library

to change during runtime. It would also make the reference-implementation a lot more complex, since we would need to implement a mechanism, that monitors the files for changes. This simplicity is also the reason, why all operations are implemented in serial, meaning the files are read one after another. Parallelizing this process is not necessary for the current size of our database and would only complicate the process.

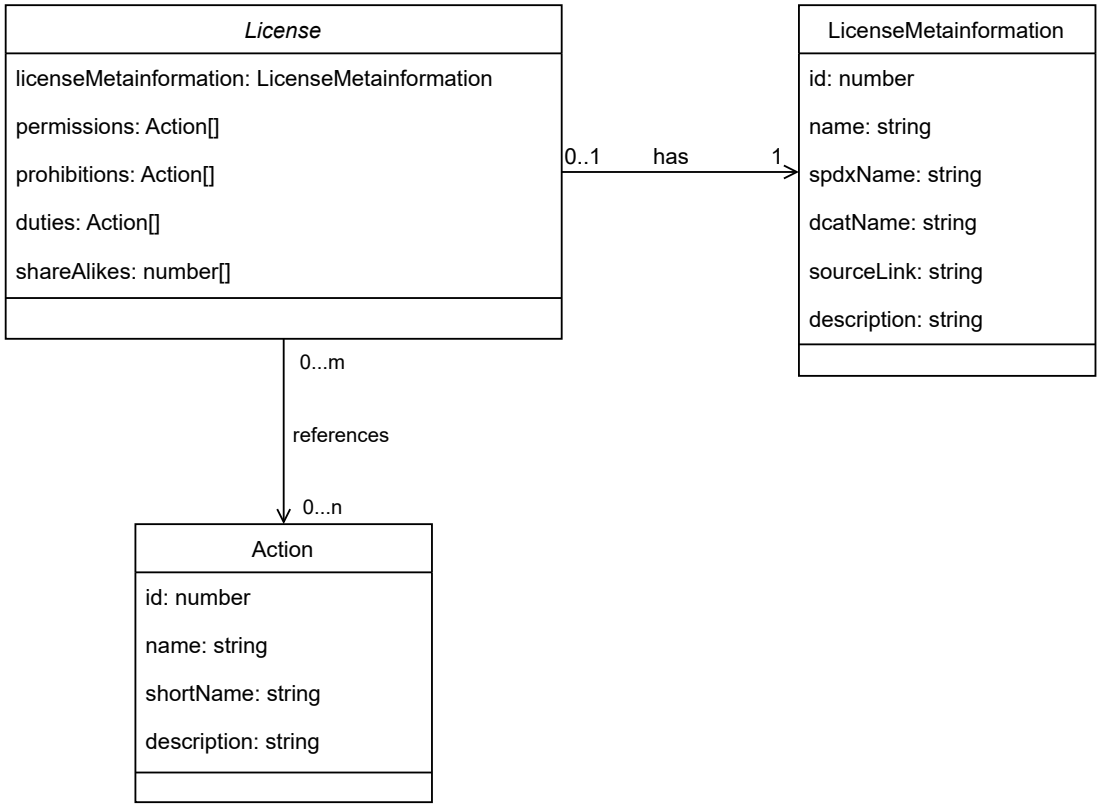


Figure 5.3: Class-Diagram for the base entities

5.2.2 Business-Logic

As mentioned before we split the business logic into five classes to keep them separated by concern and keep the code observable:

- License-Search: `LicenseFinder`
- License-Compatibility-Checks: `Checker`
- License-Aggregation: `Aggregator`
- License-Recommendations: `Recommender`
- Pipeline-Simulation: `Pipeline`

The class `LicenseFinder` is the only connection from the business- to the data-access-layer. It contains three methods that can be used to either return all licenses or search for a specific-license by name or by id. If the algorithm does not find a license, it returns an object of type `undefined` signaling to the caller that no matching license was found.

The class `Checker` implements the logic needed to compare the compatibility

between two licenses. It implements the logic described in section 4.3 and provides the two functions `areCompatible()` and `runCompatibilityChecks()`. The first function contains the logic to compare two licenses, while the latter executes this function over all possible license-combinations. The results of the compatibility-check are returned as an object of type `CompatibilityCheckResult` containing the names of the compared licenses, the results of the single sub-checks and the final result.

The class `Aggregator` implements the aggregation-logic described in section 4.4. It has the three functions `createCompositeLicense()`, `extendCompositeLicense()` and `runFullAggregation()`. The function `createCompositeLicense()` creates an aggregate of two licenses. This aggregate can then be extended by adding another license by calling `extendCompositeLicense()`. The third function `runFullAggregation()` is similar to `runCompatibilityChecks()` and runs the first function over all possible combinations of two licenses to generate a composite licenses for each combination.

Finally the class `Recommender` implements the recommendation-component of the framework (see 4.5). Using the function `findSimilarLicenses()` a caller can automatically search for a license that matches a provided composite-license. It returns a list of licenses that are similar or more restrictive than the provided composite-license. The results also includes a hint, how the composite license differs from the original license.

The class `Pipeline` combines the previously described classes and recreates the pipeline we described when designing our framework. After initializing an object of this class, a caller can add multiple license-names to an internal array. This array is then used to automatically create an composite license, using the logic implemented in `Aggregator`. It also provides two functions to to run a full compatibility-check for the stored licenses (`checkLicenses()`) and generate a list of recommendations for the created aggregate(`getRecommendations()`). Finally it provides a function to create a human- and machine-readable representation of the results of the pipeline as a JSON-string, so users can download it and attach the result to their dataset.

The search is implemented using a brute-force search, meaning the function iterates over the array of existing licenses until a license with a matching parameter is found. A more complex search-algorithm is not needed, since we at most have to check 19 items. Because of this small sample-size, the search stays computationally unproblematic, even if we use a brute force approach.

The comparisons, aggregation and recommendation are done by directly comparing the actions of a license against the actions of the other license. To ensure we do not miss an action, this usually means looping over the actions of both licenses and checking if the specified conditions are met. Of course this also means,

```
getLicense(name: string) : License | undefined {
    return this.db.licenses.find((license) =>
     ↪  license.metaInformation.name == name);
}
```

Listing 3: The function getLicense as an example for search-functions

that, in the worst-case, we have to loop over all actions of every licenses multiple times. One possible improvement for this would have been to reduce the number of loops by combining some of the checks into a single loop. However this introduces dependencies between the different functions, which can be problematic during debugging or when making future improvements. It would also make the code less readable, since the additional functions would add additional conditions to the loops. Since we only have a very limited scope of actions and licenses at the moment, we went with the simpler solution for our reference implementation. If the number of licenses and potential actions grows in the future, this part of the implementation is likely in need of some improvements.

5.2.3 Interface-Layer

The final part of our library is the interface-layer. This layer is used to make the previously describe logic available to a caller of the library. This is done by exporting the previously described types and classes through the index-file of the solution. The index-file functions as proxy for the library. This means a caller of the library can import all classes via one reference, instead of having to reference every class on its own.

```
function join(actions1: Action[], actions2: Action[]):
↪  Action[] {
    let results: Action[] = [];

    if ((actions1.length - actions2.length) >= 0) {
        for (let i = 0; i < actions2.length; i++) {
            let action2 = actions2[i];

            let index = actions1.findIndex((action1) =>
            ↪  areEqual(action1, action2));

            if (index >= 0) {
                results.push(action2);
            }
        }
    } else {
        for (let j = 0; j < actions1.length; j++) {
            let action1 = actions1[j];

            let index = actions2.findIndex((action2) =>
            ↪  areEqual(action2, action1));

            if (index >= 0) {
                results.push(action1);
            }
        }
    }

    return results;
}
```

Listing 4: The implementation of a join-function for the license-aggregation

6 Demonstration

To demonstrate our previously described artifact, we integrated the core library into a website-prototype, that can be used to view the license-descriptions, check the compliance of licenses and create reports by simulating the inputs we expect from a data pipeline.

There were multiple reasons, why we went with a website as our demonstrator. This solution enabled us to test the framework in multiple different scenarios, by building different user-interfaces to mock them. This was much faster than creating multiple different data pipelines. The interfaces also have the added benefits, that they can be used to either evaluate new ideas or demonstrate specific parts of the framework to other researchers. Since it is set up as a stand-alone application, it does not require much preparation to run and has no external dependencies, meaning the setup-process is relatively easy. It also has the potential to be extended into an online-tutorial or -documentation at a later point in time, in case we want to promote our framework in the future.

As an alternative to the website-prototype, we considered to demonstrate the framework by integrating the reference-implementation into the Jayvee-project. This would have given us a real-world test-environment for our framework. However to do so, a lot of additional changes to the project would have been necessary. Since the amount of changes would have exceeded the timeline of this thesis, we decided to go with the website-prototype.

The demonstrator is again implemented using Typescript and uses the Remix-framework[1] as a web-framework. We chose this setup to ensure that our reference-implementation seamlessly integrates with the demonstrator. All used npm-packages are referenced in Appendix I.

[1] https://remix.run/

The demonstrator is split into four parts:

- A license-overview displaying the stored license-information.
- A compatibility-matrix showing the compatibility between different licenses.
- A view showing the results for the aggregates between different licenses.
- An interactive simulation for a data pipeline.

Each part is either used to visualize a specific part of the framework or to demonstrate how the framework could be used in a specific scenario. We will not go into much detail on the implementation of each page, but instead focus on the goals and motivations for each of them. The code is again published in the same open-source-repository as the library (Philip Rebbe, 2024).

6.1 License-Overview

Figure 6.1: Screenshot for the license overview

The license-overview shows the different license-descriptions. This view is an introduction to our license-model and is meant to help users learn and understand it. When a user first clicks on the corresponding tab, the page shows a list of all available licenses. The user can then click on a link, which redirects him to another view showing the details of this specific license. This includes the stored meta-informations and a list of all permission, prohibitions and duties for this license.

6.2 Compatibility-Matrix

The main focus of the compatibility-matrix is the logic to check the compatibility of licenses. When the page is loaded, a full-scale compatibility-check is run via the logic implemented in the the core library. The result is then displayed

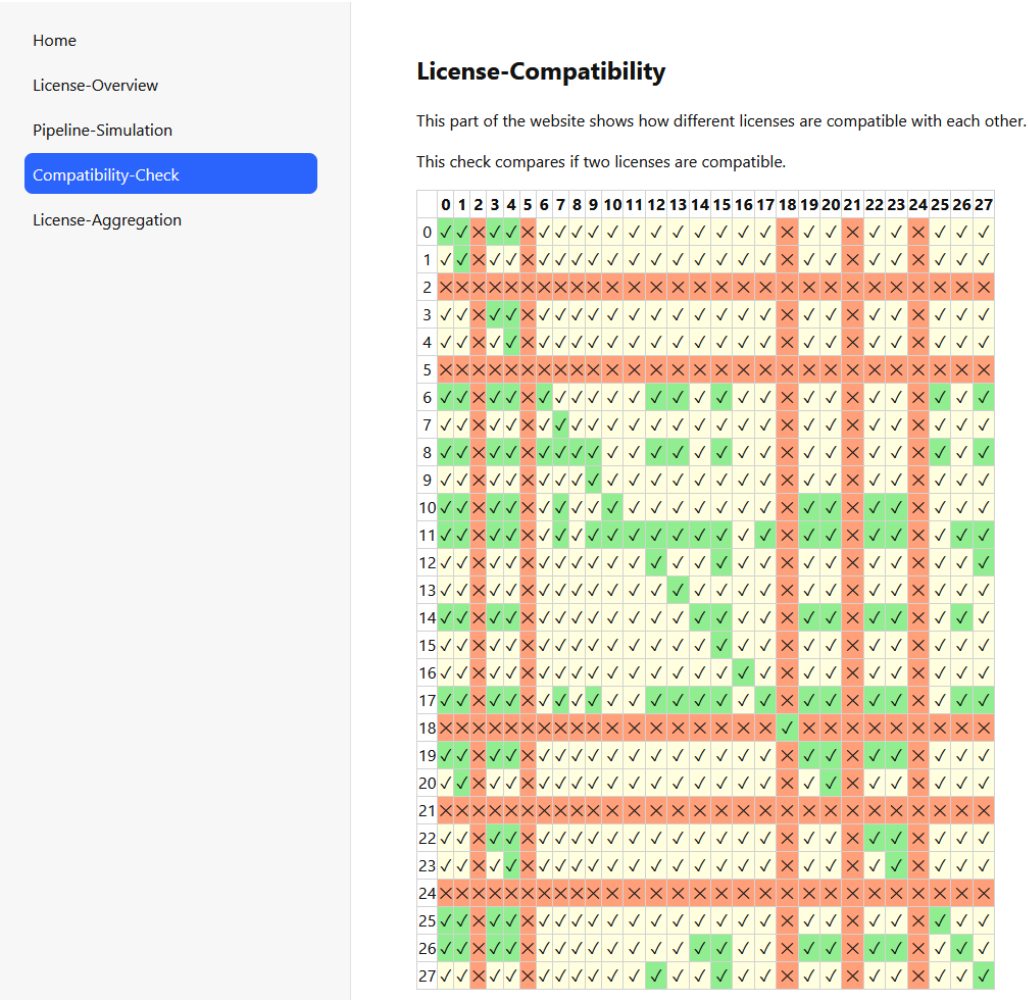


Figure 6.2: Screenshot for the compatibility-matrix

in a matrix showing the compatibility between all possible license combinations. The main goal for this view is to give an overview, which licenses are currently classified as compatible by our library, meaning that we can create a valid aggregate for this license. This was useful to find flaws in the current version of the framework, get ideas for new improvements or to validate our framework against other frameworks from literature.

6.3 License-Aggregation-Overview

This view displays a table containing the aggregation-results for all potential license-pairs. Like the previous page, the results are generated by core-library on page-load and then displayed in a table. It was created to be able to check if the logic of the framework was working and find remaining inconsistencies.

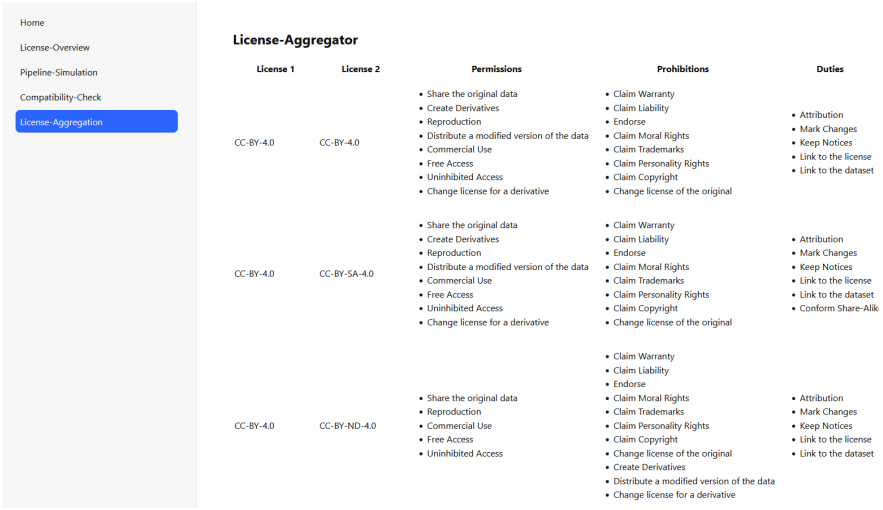


Figure 6.3: Screenshot for the license-aggregation overview

6.4 Pipeline-Simulation

This view recreates the use-case we described in the introduction (see figure 1.1). To recap, in the Jayvee-project one or more data pipelines can flow into the same sink, resulting in a new dataset containing data from all the previous datasets. This means that the new dataset and its license must be compatible with all the previous licenses.

The page provides users with the option to choose one or more licenses from a dropdown-list. This dropdown simulates the inputs from the different data pipelines into our framework. A user can add one license at the time to the pipeline representing the addition of a new dataset to the sink. The licenses are then automatically run through the pipeline of our framework, implemented using the `Pipeline`-class from the library. The pipeline then automatically checks the licenses for compatibility, aggregates them into a composite license and a license-recommendation is generated. The results of each step are then displayed below the dropdown-list to enable a manual evaluation of the results. Finally the JSON-representation of the pipeline-result is displayed at the end of the page to show the output from our pipeline.

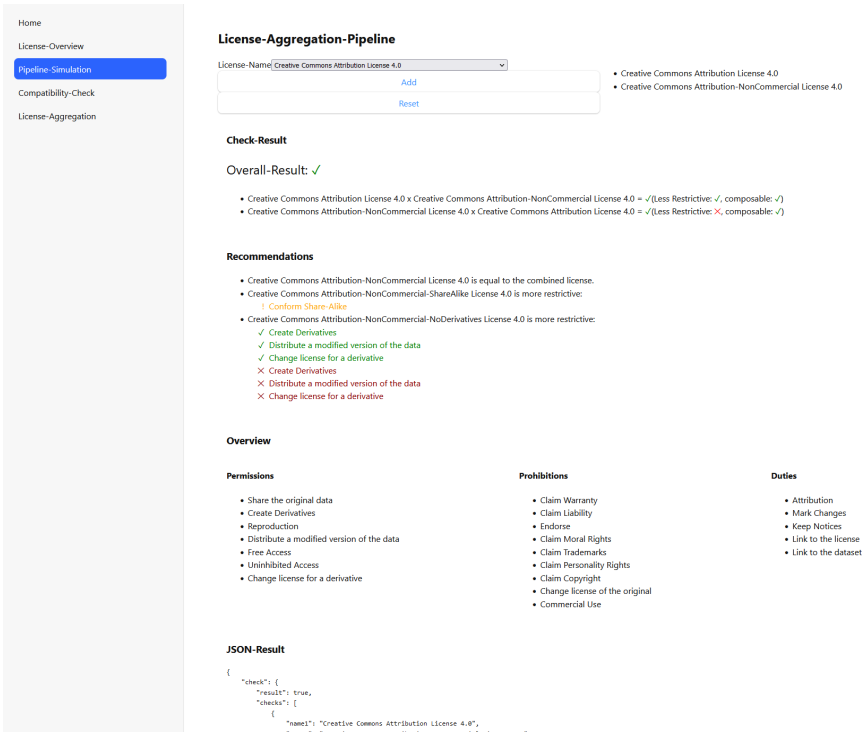


Figure 6.4: Screenshot for the pipeline-simulation

7 Evaluation

As the final step of the design-science research methodology, we evaluated our artifact. The goal for this step is to see if the created "artifact supports a solution to the artifact" (K. Peffers et al., 2007). This included testing the validity of our framework and testing how our framework functioned in comparison to other frameworks.

7.1 Evaluation of our Objectives

7.1.1 Objectives for the Framework

Creating a Model to Describe Licenses

To verify if the defined actions could be used to implement licenses that were not part of the original dataset, we tried to model older or other versions of the same licenses to see if they would be marked as compatible or if we needed to include additional actions. This included the earlier versions of the "Open Government License", "Licence Ouverte", the CC-licenses and the CC-public domain mark. All of these licenses could be modeled with the defined actions (see table 7.1). The new license-descriptions either were copies of the original licenses or were less restrictive versions than the newer iterations.

Comparing and Recommending Licenses

To ensure the validity of our framework, we compared the results from our framework with the results in existing literature. This included existing compatibility-matrices like the one provided by CC[1] or existing reports like the one describe by the Ministerium für Wirtschaft, Innovation, Digitalisierung und Energie des Landes Nordrhein-Westfalen and Beauftragte der Landesregierung für Informationstechnik (CIO) / Geschäftsstelle Open.NRW (2019) that detail how some licenses should be classified. In case a license explicitly stated that it should be

[1] https://wiki.creativecommons.org/wiki/Wiki/cc_license_compatibility

Table 7.1: List of older licenses in comparison with newer versions

Added License	Comparable License	Compatible?
CC-BY-3.0	CC-BY-4.0	Yes
CC-BY-SA-3.0	CC-BY-SA-4.0	Yes
CC-BY-ND-3.0	CC-BY-ND-4.0	No (No derivatives)
CC-BY-NC-3.0	CC-BY-NC-4.0	Yes
CC-BY-NC-SA-3.0	CC-BY-NC-SA-4.0	Yes
CC-BY-NC-ND-3.0	CC-BY-NC-ND-4.0	No (No derivatives)
Public Domain Mark	CC0	Yes
Open Government License v2	Open Government License v3	Yes
Licence Ouverte v1	Licence Ouverte v2	Yes

compatible with a specific license, we also validated if these licenses were classified as compatible. Examples for this are the "Open Government License" or the "Norwegian Licence for Open Government Data (NLOD) 2.0" which explicitly state their compatibility with the CC-licenses. We tested this by entering the specific licenses into the demonstrator and comparing if the compatibility-checks and recommendations matched the expected output.

As is shown in 7.2, there are some licenses that were classified as not compatible by the new framework, even though they should be compatible according to the literature. The reason for this is that we included the disclaimers for other rights like personal-rights into our model. This was done to ensure that these prohibitions would be included in the composite license, but also resulted in a pessimistic interpretation of the license-compatibility. A possible solution to change this would be to combine them into a single action or exclude them when comparing two licenses. Since both solutions seem promising, this should be evaluated in future works.

7.1.2 Objectives for the Library

Comparison with DALICC-Framework

To further test if our framework implemented the adopted logic correctly, we also conducted a test against one of the other frameworks. For the comparison, we used results from the DALICC-Api[2] and from the demonstrator. The DALICC-Api provides an endpoint called "compatibilitycheck" that takes a POST-request containing the licenses it should compare and returns a list of all found conflicts between the licenses (see listing 5). It compares two licenses and checks if the second license contains a condition that makes it incompatible with the

[2]https://api.dalicc.net/docs

first license. To replicate this, we entered the licenses into our demonstrator and checked if the first license would be marked as a valid recommendation for the resulting composite. Because both frameworks implement different sets of licenses, we limited the test to the licenses that both frameworks implemented.

We had two goals for this test: The first goal was to see how our framework classified the licenses in comparison to the DALICC-framework. The second goal was to see if all conflicts, registered by the DALICC-framework, were recognized by our framework. Out of the 121 license-combinations we looked at, 45 were marked as conflicting by the DALICC-framework. In comparison to this, our framework registered 95 conflicts, including the 45 conflicts found by the DALICC-framework. This verified that our framework found all conflicts registered by the DALICC-framework. But it also showed that our framework is a lot more pessimistic when comparing licenses. This is caused by the disclaimer actions, as we already mentioned, but also because we added the additional rules regarding "Share-Alike" and not allowing a derivative license, in case one license did not allow it. This resulted in us automatically eliminating license that were classified as compatible by the DALICC-framework. Out of the 50 additional conflicts that were registered, 18 cases were prohibited by the "NoDerivative"-clause, 23 cases were rejected because of the added actions and 9 cases were removed because of the new Share-Alike check.

Collecting Feedback from Jayvee-Team

Finally the artifact and demonstrator were presented to the JValue-Team during one of their developer-meetings in February of 2024 to collect feedback from a group of experts in the context of data pipelines. This was done to gather feedback regarding the current state of the library and on how the demonstrator could be improved to better represent the expected use-case.

The feedback for the demonstrator was overall positive, with some suggestions on how to improve the user-interface (UI). This included suggestions like adding some visual hints or explanations for certain actions that were displayed during each step. However in regards to the library the developers expressed the idea to include license-recommendations based on the created license-aggregates. The reason for this was that while they saw the composite license as a good indicator on how to proceed, when relicensing the new dataset, they would prefer if the system was able to give a recommendation which licenses could be used to fulfill the requirements of this composite license. Since we already had a similar idea on how to integrate this feature into our framework, we then took another iteration of the design-science process to include this feature into our artifact, resulting in the version presented in this thesis.

7.2 Limitations

There are still some limitations regarding the framework and library. One limitation, the pessimistic interpretation of certain licenses, was already mentioned during the evaluation of the objectives.

Another limitation of this framework is the missing functionality to extract the license-information from the meta-information of a dataset. As mentioned during the objective-definition, we decided to remove this part from this iteration of the framework and therefore the core-library to allow us to focus on the logic for the license-operations. Because of this users of the library currently either have to implement the extraction of meta-information themself and then pass this information to the library or manually enter the information to get a result from the library. To complete the framework and core-library into a solution that can handle all operations regarding license information this function has to be added in the future. It should also add additional export-formats to support the requirements of different data-platforms.

The framework is also limited by the assumption that each dataset is represented by a single license. Datasets can be licensed under multiple licenses at the same time. Currently this can be problematic in cases were one of the licenses does not allow derivatives, since the framework would interpret two licenses as the combination of two datasets and therefore reject them. This could be solved in the future by extending the framework to allow passing a combinations of both license to the framework or aggregating them before passing them through the pipeline.

Finally the framework and library were not validated by a legal expert and are therefore not able to give legally binding advice. Since our goal was to find a logic-based approach to the problem of license-comparison, this does not invalidate the proposed solution, but should be mentioned again at this point to prevent any misunderstandings. Both the framework and library were created based on the knowledge found in the cited literature and the experiences of the author and provide a suggestion to the user. It is not a replacement for the advice of a lawyer and must not be used as such, in case someone needs any legal advice.

Table 7.2: List of validations according to the literature

License	Compatible with according to the literature	Matched Result
CC-BY-4.0	CC-BY-4.0	Yes
CC-BY-4.0	CC-BY-SA-4.0	Yes
CC-BY-4.0	CC-BY-NC-4.0	Yes
CC-BY-4.0	CC-BY-NC-SA 4.0	Yes
CC-BY-SA-4.0	CC-BY-4.0	Yes
CC-BY-SA-4.0	CC-BY-SA-4.0	Yes
CC-BY-NC-4.0	CC-BY-4.0	Yes
CC-BY-NC-4.0	CC-BY-NC-4.0	Yes
CC-BY-NC-4.0	CC-BY-NC-SA-4.0	Yes
CC-BY-NC-SA-4.0	CC-BY-NC-SA-4.0	Yes
Data licence Germany-Zero-Version 2.0	CC-BY-4.0	Yes
Data licence Germany-Zero-Version 2.0	ODC-By	No
Open Government License v3	CC-BY-4.0	No
Open Government License v3	ODC-By	No
Licence Ouverte	CC-BY-4.0	Yes
Licence Ouverte	ODC-By	No
Licence Ouverte	Open Government Licence	No
NLOD 2.0	CC-BY-4.0	No
NLOD 2.0	Open Government License v2	No
NLOD 2.0	Open Government License v3	No

```
{
    "conflicting_statements": {
        "direct": {
            "0": {
                "statement_1": [
                    "https://dalicc.net/licenselibrary/
                    ↪ CC-BY-NC-4.0",
                    "http://www.w3.org/ns/odrl/2/permission",
                    "http://www.w3.org/ns/odrl/2/derive"
                ],
                "statement_2": [
                    "https://dalicc.net/licenselibrary/
                    ↪ CC-BY-ND-4.0",
                    "http://www.w3.org/ns/odrl/2/prohibition",
                    "http://www.w3.org/ns/odrl/2/derive"
                ],
                "reason": "Direct permission-prohibition
                ↪ conflict."
            },
            "1": {
                "statement_1": [
                    "https://dalicc.net/licenselibrary/
                    ↪ OdcPublicDomainDedicationAndLicence",
                    "http://www.w3.org/ns/odrl/2/permission",
                    "https://dalicc.net/ns#ChangeLicense"
                ],
                "statement_2": [
                    "https://dalicc.net/licenselibrary/
                    ↪ CC-BY-ND-4.0",
                    "http://www.w3.org/ns/odrl/2/prohibition",
                    "https://dalicc.net/ns#ChangeLicense"
                ],
                "reason": "Direct permission-prohibition
                ↪ conflict."
            },
            ...
        },
        "derived": {}
    }
}
```


Listing 5: Extract from the response of the DALICC-API

8 Conclusions

The goal of this thesis was to create a solution for integrating open data licenses into data pipelines. For this we looked at how open data licenses affect data pipelines and what kind of problems occur when working with them.

To solve these problems, we created a framework that can model and compare open data licenses. To do so, we looked and compared existing frameworks and other literature that are concerned with the topic of license-compatibility to find a solution that would fit the context of data pipelines. The created framework includes the functionality to check two licenses for compatibility, aggregate them into a composite license and give recommendations in case the composite license matches an existing license.

We then created a Typescript-based reference-implementation for this framework and integrated it into a web-based demonstrator to test how the framework could be used in a real data pipeline to create reports or handle license-comparisons.

The evaluation was done by comparing the results against existing literature and other frameworks from literature. We also tested if other licenses could be included in it as well and collected feedback from the JValue-team on how to improve the framework even further.

For future works, we would recommend to integrate this framework into the Jayvee-project to gather more real-world examples and extend the core-library to support some of the missing functionalities, like the ability to automatically extract license-information from a datasource. It should also be analyzed by a group of legal experts to find out, what kind of improvements would be necessary to gain an official certification for the framework and library. Another interesting option could be to use the insights gathered during this thesis to create a ML-based tool that is trained by crawling and analyzing datasets contained in different datahubs.

Appendices

A List of Identified Actions - Part 1

Id	Name	Definition
0	Share	Use and sharing the original data
1	Modify	Create, use and share a derivative
2	Commercial Use	Use the derivative commercially
3	New License	Define a new license for the dataset
4	Warranty	You can disclaim the Warranty
5	Liability	You can disclaim the Liability
6	MoralRights	You can disclaim the moral rights that are attached to the content of the dataset
7	Trademarks	You can disclaim any trademarks used in the data
8	DataProtection	You can disclaim the data protection rights attached ot the content of the dataset
9	PersonalityRights	You can disclaim the personality rights that are attached to the content of the dataset
10	PatentRights	You can disclaim the personality rights that are attached to the content of the dataset
11	CopyrightRights	You can disclaim a copyright that are attached to the content of the dataset
12	DesignRights	You can disclaim the design rights that are attached to the content of the dataset
13	PersonalData	You can prevent usage of personal data that is part of the content of the dataset
14	ThirdPartyRights	You can disclaim third party rights that are attached to the content of the dataset
15	Endorsement	You can use the dataset for endorsement
16	Free Access	You can freely access the dataset
17	UninhibitedAccess	You can access without inhibitions

B List of Identified Actions - Part 2

Id	Name	Definition
0	Share-Alike	Requires Share-Alike
1	Attribution by Name	Name the original creator
2	Describe Changes	Describe Changes made to the data
3	Copyright-Notice	Maintain all copyright-notices
4	Link to Source	Include a link to the original source
5	Link to License	Include a link to the license
6	Limit License Version	Limit the allowed version license to a specific version

C List of Normed Actions

Id	Name	Origin	Definition
0	Share	-	Sharing the original data
1	Reproduction	cc	Making multiple copies
2	DerivativeWorks	-	Create derivative works
3	Distribution	cc	Distribution, public display, and publicly performance
4	CommercialUse	cc	Exercising rights for commercial purposes
5	ShareAlike	-	If you want to relicense the data you have to use the same or a similar license
6	Attribution	cc	Credit be given to copyright holder and/or author
7	Inform	odrl	To inform that an action has been performed on or in relation to the Asset.
8	Notice	cc	copyright and license notices be kept intact
9	LinkLicense	-	You have to include the text or a link to the original license
10	LinkDataset	-	You have to provide a link to the original dataset
11	LimitLicenseVersion	-	You have to use a specific version of this license or one compatible with it, to relicense the dataset
12	ClaimWarranty	-	You can claim a warranty for the provided information
13	ClaimLiability	-	The provider can be made liable for damages
14	Endorse	-	You can use the use of the data to endorse your project
15	FreeAccess	-	Accessing the data is free of charge
16	UninhibitedAccess	-	Accessing the data is not prevented by unneccessary means
17	ClaimMoralRights	-	You can claim the moral rights that are attached to the content of the dataset
18	ClaimTrademarks	-	You can claim any trademarks used in the data
19	ClaimDataProtection	-	You can claim the data protection rights attached ot the content of the dataset

Id	Name	Origin	Definition
20	ClaimPersonalityRights	-	You can claim the personality rights that are attached to the content of the dataset
21	ClaimPatentRights	-	You can claim the personality rights that are attached to the content of the dataset
22	ClaimCopyrightRights	-	You can claim a copyright that are attached to the content of the dataset
23	ClaimDesignRights	-	You can claim the design rights that are attached to the content of the dataset
24	ClaimPersonalData	-	You can use personal data that is part of the content of the dataset
25	ClaimThirdPartyRights	-	You can claim third party rights that are attached to the content of the dataset
27	Relicense	-	The license of the unchanged dataset can be changed to another license

D Example - Less-Restrictiveness

$$
\begin{aligned}
I_a =& 0 \\
Pe_a =& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}Reproduction\text{"}, \\
& \text{"}Distribution\text{"}, \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \\
& \text{"}ChangeLicense\text{"}\} \\
Pr_a =& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \\
& \text{"}ClaimMoralRights\text{"}, \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \\
& \text{"}ClaimPatentRights\text{"}, \text{"}ClaimCopyrightRights\text{"}, \\
& \text{"}Relicense\text{"}\} \\
R_a =& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \text{"}LinkDataSet\text{"}\} \\
S_a =& \{\}
\end{aligned}
$$

$$
\begin{aligned}
L_a =& \langle 0, \\
& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}Reproduction\text{"}, \text{"}Distribution\text{"}, \\
& \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \text{"}ChangeLicense\text{"}\}, \\
& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \text{"}ClaimMoralRights\text{"}, \\
& \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \text{"}ClaimPatentRights\text{"}, \\
& \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\}, \\
& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \text{"}LinkDataSet\text{"}\}, \\
& \{\}\rangle
\end{aligned}
$$

$$
\begin{aligned}
I_b =& 1 \\
Pe_b =& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}Reproduction\text{"}, \\
& \text{"}Distribution\text{"}, \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \\
& \text{"}ChangeLicense\text{"}\} \\
Pr_b =& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \\
& \text{"}ClaimMoralRights\text{"}, \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \\
& \text{"}ClaimPatentRights\text{"}, \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\} \\
R_b =& \{\text{"}ShareAlike\text{"}, \text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, LinkLicense\text{"}, \\
& \text{"}LinkDataSet\text{"}\} \\
S_b =& \{1\}
\end{aligned}
$$

$$
\begin{aligned}
L_b =& \langle 1, \\
& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}Reproduction\text{"}, \text{"}Distribution\text{"}, \\
& \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \text{"}ChangeLicense\text{"}\}, \\
& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \text{"}ClaimMoralRights\text{"}, \\
& \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \text{"}ClaimPatentRights\text{"}, \\
& \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\}, \\
& \{\text{"}ShareAlike\text{"}, \text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \\
& \text{"}LinkDataSet\text{"}\}, \\
& \{1\}\rangle
\end{aligned}
$$

R0a: $(Pe_a \cap Pr_a = \{\}) \wedge (Pe_a \cap R_a = \{\}) \wedge (Pr_a \cap R_a = \{\})(true)$

R0b: $(Pe_b \cap Pr_b = \{\}) \wedge (Pe_b \cap R_b = \{\}) \wedge (Pr_b \cap R_b = \{\})(true)$

-> R0 fulfilled

R1: $Pe_a \supseteq Pe_b(true)$

-> R1 fulfilled

R2: $Pr_a \subseteq Pr_b(true)$

-> R2 fulfilled

R3: $R_a \subseteq R_b(true)$

-> R3 fulfilled

R4a: "$ChangeLicense$" $\in Pe_a(true)$

R4b: "$ShareAlike$" $\notin R_a \vee$ ("$ShareAlike$" $\in R_a \wedge I_b \in S_a)(true)$

-> R4 fulfilled

License A: $L_a = \langle l_0, l_1, l_2, l_3, l_4 \rangle$ with $l_0 = I_a, l_1 = Pe_a, l_2 = Pr_a, l_3 = R_a$ and $l_4 = S_a(true)$

-> License A fulfilled

License B: $L_b = \langle l_0, l_1, l_2, l_3, l_4 \rangle$ with $l_0 = I_b, l_1 = Pe_b, l_2 = Pr_b, l_3 = R_b$ and $l_4 = S_b(true)$

-> License B fulfilled

R7a: "$DerivativeWorks$" $\in Pe_a(true)$

R7b. "$DerivativeWorks$" $\in Pe_b(true)$

-> R7 fulfilled

-> license A is less restrictive than license B

E Example - Composition-Check

$$
\begin{aligned}
I_a =& 0 \\
Pe_a =& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}CommercialUse\text{"}, \text{"}Reproduction\text{"}, \\
& \text{"}Distribution\text{"}, \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \text{"}ChangeLicense\text{"}\} \\
Pr_a =& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \text{"}ClaimMoralRights\text{"}, \\
& \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \text{"}ClaimPatentRights\text{"}, \\
& \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\} \\
R_a =& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \text{"}LinkDataSet\text{"}\} \\
S_a =& \{\}
\end{aligned}
$$

$$
\begin{aligned}
L_a =& \langle 0, \\
& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}CommercialUse\text{"}, \text{"}Reproduction\text{"}, \\
& \text{"}Distribution\text{"}, \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \text{"}ChangeLicense\text{"}\}, \\
& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \text{"}ClaimMoralRights\text{"}, \\
& \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \text{"}ClaimPatentRights\text{"}, \\
& \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\}, \\
& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \text{"}LinkDataSet\text{"}\}, \\
& \{\}\rangle
\end{aligned}
$$

$I_b = 1$
$Pe_b = \{$"$Sharing$","$DerivativeWorks$","$CommercialUse$","$Reproduction$",
"$Distribution$","$FreeAccess$","$UninhibitedAccess$","$ChangeLicense$"$\}$
$Pr_b = \{$"$ClaimWarranty$","$ClaimLiability$","$Endorse$","$ClaimMoralRights$",
"$ClaimTrademark$","$ClaimPersonalityRights$","$ClaimPatentRights$",
"$ClaimCopyrightRights$","$Relicense$"$\}$
$R_b = \{$"$ShareAlike$","$Attribution$","$Inform$","$Notice$","$LinkLicense$",
"$LinkDataSet$"$\}$
$S_b = \{1\}$

$L_b = \langle 1,$
$\{$"$Sharing$","$DerivativeWorks$","$CommercialUse$","$Reproduction$",
"$Distribution$","$FreeAccess$","$UninhibitedAccess$","$ChangeLicense$"$\}$,
$\{$"$ClaimWarranty$","$ClaimLiability$","$Endorse$","$ClaimMoralRights$",
"$ClaimTrademark$","$ClaimPersonalityRights$","$ClaimPatentRights$",
"$ClaimCopyrightRights$","$Relicense$"$\}$,
$\{$"$ShareAlike$","$Attribution$","$Inform$","$Notice$","$LinkLicense$",
"$LinkDataSet$"$\}$,
$\{1\}\rangle$

R0a: $(Pe_a \cap Pr_a = \{\}) \wedge (Pe_a \cap R_a = \{\}) \wedge (Pr_a \cap R_a = \{\})(true)$

R0b: $(Pe_b \cap Pr_b = \{\}) \wedge (Pe_b \cap R_b = \{\}) \wedge (Pr_b \cap R_b = \{\})(true)$

-> R0 fulfilled

R5: $Pe_a \cap Pe_b \neq \{\}(true)$

-> R5 fulfilled

R6a: "$ChangeLicense$" $\in Pe_a(true)$

R6b: "$ChangeLicense$" $\in Pe_b(true)$

-> R6 fulfilled

License A: $L_a = \langle l_0, l_1, l_2, l_3, l_4 \rangle$ with $l_0 = I_a, l_1 = Pe_a, l_2 = Pr_a, l_3 = R_a$ and $l_4 = S_a(true)$

-> License A fulfilled

License B: $L_b = \langle l_0, l_1, l_2, l_3, l_4 \rangle$ with $l_0 = I_b, l_1 = Pe_b, l_2 = Pr_b, l_3 = R_b$ and $l_4 = S_b(true)$

-> License B fulfilled

R7a: "$DerivativeWorks$" $\in Pe_a(true)$

R7b: "$DerivativeWorks$" $\in Pe_b(true)$

-> R7 fulfilled

-> license A and license B can be composed!

F Example - Recommendation

Variable-Definitions - Composite License:

$$
\begin{aligned}
Inc =& (0,1)\\
CPe =& \{"Sharing","DerivativeWorks","CommercialUse","Reproduction",\\
& "Distribution","FreeAccess","UninhibitedAccess","ChangeLicense"\}\\
CPr =& \{"ClaimWarranty","ClaimLiability","Endorse","ClaimMoralRights",\\
& "ClaimTrademark","ClaimPersonalityRights","ClaimPatentRights",\\
& "ClaimCopyrightRights","Relicense"\}\\
CR =& \{"ShareAlike","Attribution","Inform","Notice","LinkLicense",\\
& "LinkDataSet"\}
\end{aligned}
$$

$$
\begin{aligned}
C =& \langle (0,1),\\
& \{"Sharing","DerivativeWorks","CommercialUse","Reproduction",\\
& "Distribution","FreeAccess","UninhibitedAccess","ChangeLicense"\},\\
& \{"ClaimWarranty","ClaimLiability","Endorse","ClaimMoralRights",\\
& "ClaimTrademark","ClaimPersonalityRights","ClaimPatentRights",\\
& "ClaimCopyrightRights","Relicense"\},\\
& \{"ShareAlike","Attribution","Inform","Notice","LinkLicense",\\
& "LinkDataSet"\},\\
& \{1\}\rangle
\end{aligned}
$$

Variable-Definitions - CC-BY-4.0:

$$
\begin{aligned}
I_0 =& 0 \\
Pe_0 =& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}CommercialUse\text{"}, \text{"}Reproduction\text{"}, \\
& \text{"}Distribution\text{"}, \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \text{"}ChangeLicense\text{"}\} \\
Pr_0 =& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \text{"}ClaimMoralRights\text{"}, \\
& \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \text{"}ClaimPatentRights\text{"}, \\
& \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\} \\
R_0 =& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \text{"}LinkDataSet\text{"}\} \\
S_0 =& \{\}
\end{aligned}
$$

$$
\begin{aligned}
L_0 =& \langle 0, \\
& \{\text{"}Sharing\text{"}, \text{"}DerivativeWorks\text{"}, \text{"}CommercialUse\text{"}, \text{"}Reproduction\text{"}, \\
& \text{"}Distribution\text{"}, \text{"}FreeAccess\text{"}, \text{"}UninhibitedAccess\text{"}, \text{"}ChangeLicense\text{"}\}, \\
& \{\text{"}ClaimWarranty\text{"}, \text{"}ClaimLiability\text{"}, \text{"}Endorse\text{"}, \text{"}ClaimMoralRights\text{"}, \\
& \text{"}ClaimTrademark\text{"}, \text{"}ClaimPersonalityRights\text{"}, \text{"}ClaimPatentRights\text{"}, \\
& \text{"}ClaimCopyrightRights\text{"}, \text{"}Relicense\text{"}\}, \\
& \{\text{"}Attribution\text{"}, \text{"}Inform\text{"}, \text{"}Notice\text{"}, \text{"}LinkLicense\text{"}, \text{"}LinkDataSet\text{"}\}, \\
& \{\}\rangle
\end{aligned}
$$

Variable-Definitions - CC-BY-SA-4.0:

$I_1 = 1$

$Pe_1 = \{$"$Sharing$","$DerivativeWorks$","$CommercialUse$","$Reproduction$",
"$Distribution$","$FreeAccess$","$UninhibitedAccess$","$ChangeLicense$"$\}$

$Pr_1 = \{$"$ClaimWarranty$","$ClaimLiability$","$Endorse$","$ClaimMoralRights$",
"$ClaimTrademark$","$ClaimPersonalityRights$","$ClaimPatentRights$",
"$ClaimCopyrightRights$","$Relicense$"$\}$

$R_1 = \{$"$ShareAlike$","$Attribution$","$Inform$","$Notice$","$LinkLicense$",
"$LinkDataSet$"$\}$

$S_1 = \{1\}$

$L_1 = \langle 1,$
$\{$"$Sharing$","$DerivativeWorks$","$CommercialUse$","$Reproduction$",
"$Distribution$","$FreeAccess$","$UninhibitedAccess$","$ChangeLicense$"$\}$,
$\{$"$ClaimWarranty$","$ClaimLiability$","$Endorse$","$ClaimMoralRights$",
"$ClaimTrademark$","$ClaimPersonalityRights$","$ClaimPatentRights$",
"$ClaimCopyrightRights$","$Relicense$"$\}$,
$\{$"$ShareAlike$","$Attribution$","$Inform$","$Notice$","$LinkLicense$",
"$LinkDataSet$"$\}$,
$\{1\}\rangle$

Constraint-Check - CC-BY-4.0 ($I_r = 0$):

R0a: $(CPe \cap CPr = \{\}) \wedge (CPe \cap CR = \{\})$
$\wedge (CPr \cap CR = \{\})(true)$

R0b: $(Pe_r \cap Pr_r = \{\}) \wedge (Pe_r \cap R_r = \{\}) \wedge (Pr_r \cap R_r = \{\})(true)$
-> R0 fulfilled

R8: $CPe \supseteq Pe_r(true)$
-> R8 fulfilled

R9: $CPr \subseteq Pr_r(true)$
-> R9 fulfilled

R10: $CR \subseteq R_r(false)$
-> R10 not fulfilled

R11: $\forall x \in Inc, L_x = \langle l_0, l_1, l_2, l_3, l_4 \rangle$
with $l_0 = x \wedge$ "$ChangeLicense$" $\in l_1 \wedge$
("$ShareAlike$" $\notin l_3 \vee$ ("$ShareAlike$" $\in l_3 \wedge I_r \in l_4)(false)$
-> R11 fulfilled

Recommendation: $L_r = \langle l_0, l_1, l_2, l_3, l_4 \rangle$ with
$l_0 = I_r, l_1 = Pe_r, l_2 = Pr_r, l_3 = R_r$ and $l_4 = S_r(true)$
-> Recommendation fulfilled

Composite: $C = \langle c_0, c_1, c_2, c_3 \rangle$ with
$c_0 = Inc, c_1 = CPe, c_2 = CPr$ and $c_3 = CR(true)$
-> Composite fulfilled

-> license 0 cannot be recommended!

Constraint-Check - CC-BY-SA-4.0 ($I_r = 1$):

R0a: $(CPe \cap CPr = \{\}) \wedge (CPe \cap CR = \{\})$
$\wedge (CPr \cap CR = \{\})(true)$

R0b: $(Pe_r \cap Pr_r = \{\}) \wedge (Pe_r \cap R_r = \{\}) \wedge (Pr_r \cap R_r = \{\})(true)$
-> R0 fulfilled

R8: $CPe \supseteq Pe_r(true)$
-> R8 fulfilled

R9: $CPr \subseteq Pr_r \wedge (CPr \cap Pe_r = \{\})(true)$
-> R9 fulfilled

R10: $CR \subseteq R_r(true)$
-> R10 fulfilled

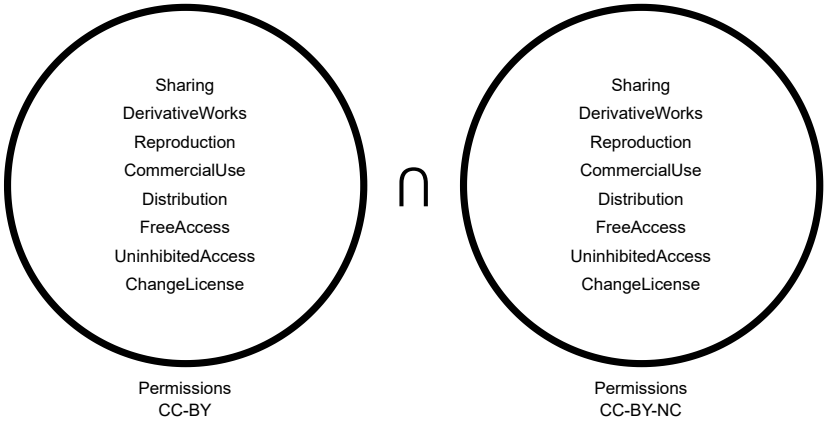
R11: $\forall x \in Inc, L_x = \langle l_0, l_1, l_2, l_3, l_4 \rangle$
with $l_0 = x \wedge$ "$ChangeLicense$" $\in l_1 \wedge$
("$ShareAlike$" $\notin l_3 \vee$ ("$ShareAlike$" $\in l_3 \wedge I_r \in l_4)(true)$
-> R11 fulfilled

Recommendation: $L_r = \langle l_0, l_1, l_2, l_3, l_4 \rangle$ with
$l_0 = I_r, l_1 = Pe_r, l_2 = Pr_r, l_3 = R_r$ and $l_4 = S_r(true)$
-> Recommendation fulfilled

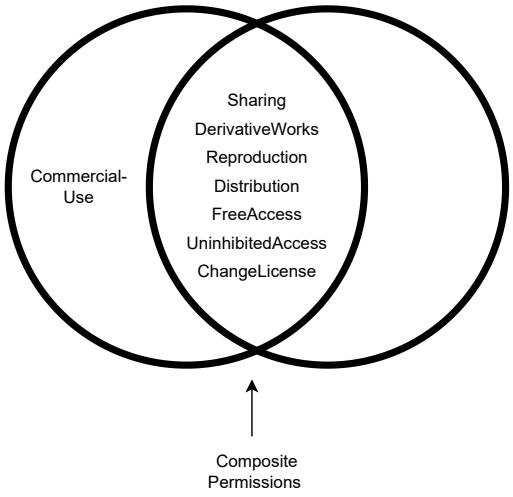
Composite: $C = \langle c_0, c_1, c_2, c_3 \rangle$ with
$c_0 = Inc, c_1 = CPe, c_2 = CPr$ and $c_3 = CR(true)$
-> Composite fulfilled

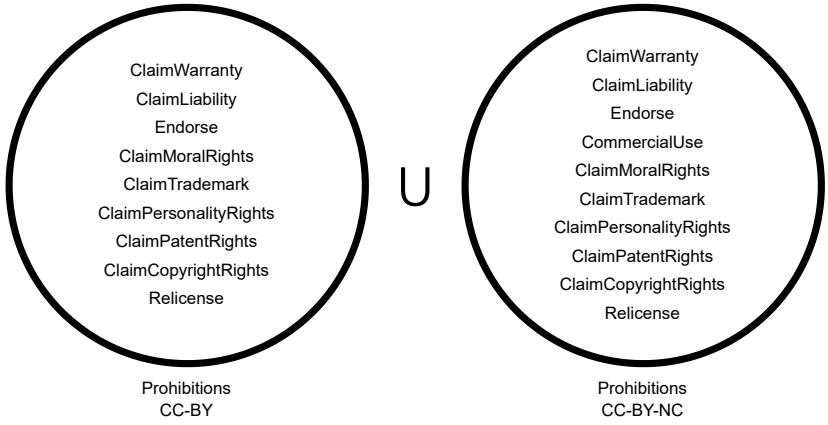
-> license 1 can be recommended!

G Visualization - AND-/OR-composition

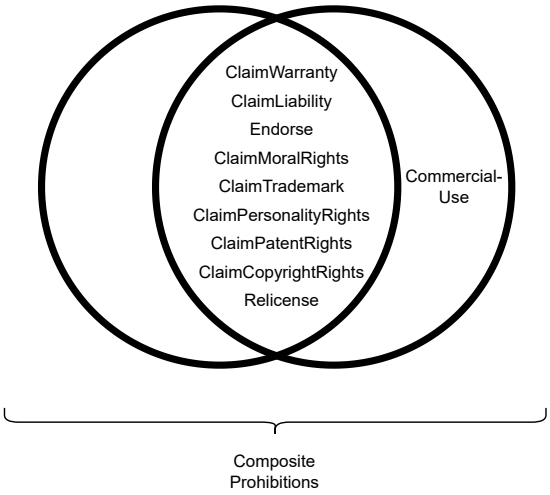


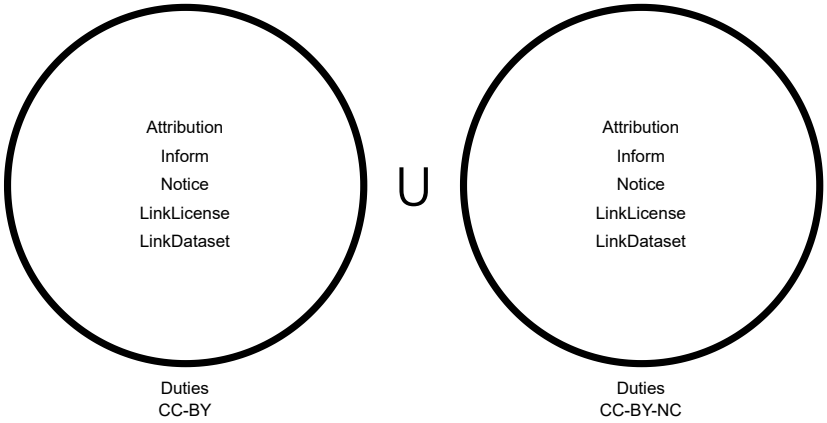
=



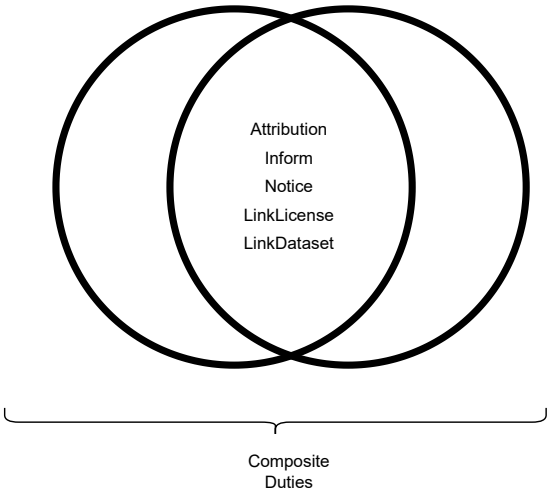

ClaimWarranty
ClaimLiability
Endorse
ClaimMoralRights
ClaimTrademark
ClaimPersonalityRights
ClaimPatentRights
ClaimCopyrightRights
Relicense
Prohibitions
CC-BY
U
ClaimWarranty
ClaimLiability
Endorse
CommercialUse
ClaimMoralRights
ClaimTrademark
ClaimPersonalityRights
ClaimPatentRights
ClaimCopyrightRights
Relicense
Prohibitions
CC-BY-NC

=

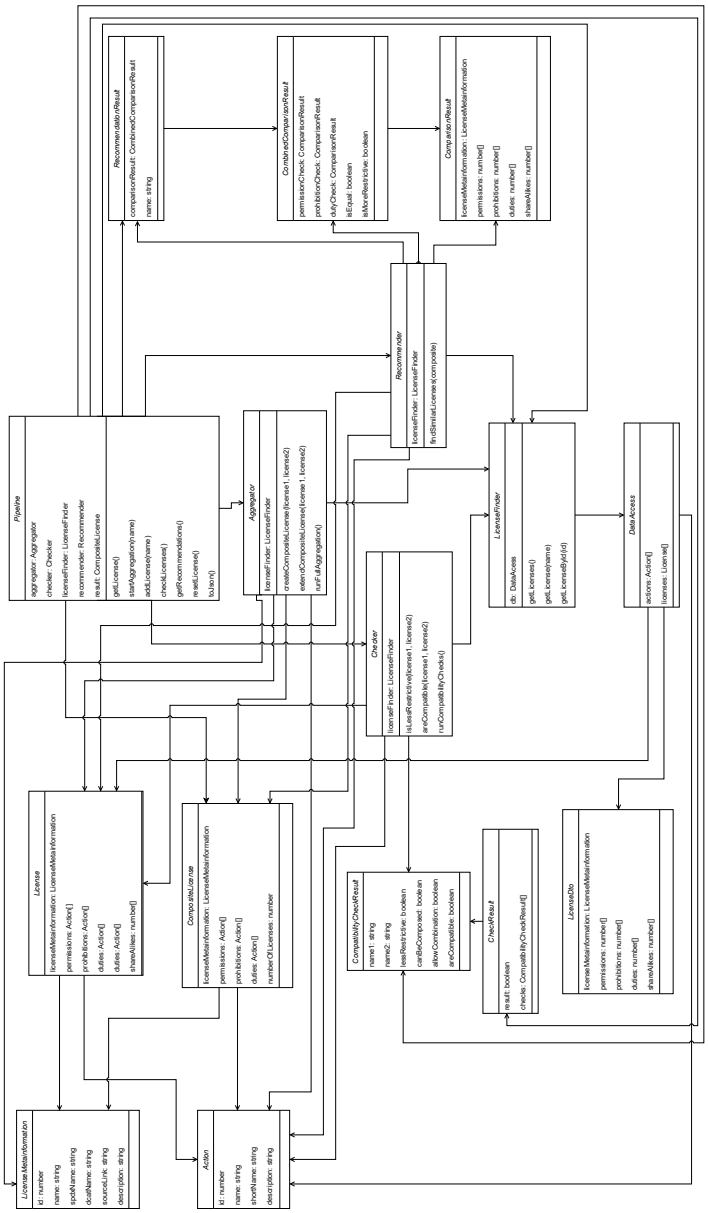

ClaimWarranty
ClaimLiability
Endorse
ClaimMoralRights
ClaimTrademark
ClaimPersonalityRights
ClaimPatentRights
ClaimCopyrightRights
Relicense
Commercial-
Use
Composite
Prohibitions


Attribution
Inform
Notice
LinkLicense
LinkDataset
U
Attribution
Inform
Notice
LinkLicense
LinkDataset
Duties
CC-BY
Duties
CC-BY-NC

=


Attribution
Inform
Notice
LinkLicense
LinkDataset
Composite
Duties

H Complete Class-Diagram



I Software Bill of Materials

This section covers the npm-packages that were used in the core-library and the web-demonstrator. These were extracted from the package.json-files used in Philip Rebbe (2024). The repository also includes a SBOM, showing the complete list of referenced packages.

I.1 Core-Library

```
{
    ...
    "devDependencies": {
        "@jest/globals": "^29.7.0",
        "@remix-run/dev": "^2.5.1",
        "@types/jest": "^29.5.12",
        "@types/node": "^20.11.0",
        "jest": "^29.7.0",
        "ts-jest": "^29.1.2",
        "ts-node": "^10.9.2",
        "typescript": "^5.3.3"
    },
    "dependencies": {
        "@remix-run/node": "^2.5.1",
        "@remix-run/react": "^2.5.1",
        "@remix-run/serve": "^2.5.1",
        "isbot": "^4.4.0",
        "react": "^18.2.0",
        "react-dom": "^18.2.0"
    }
}
```

I.2 Demonstrator

```
{
    ...
    "dependencies": {
        "@remix-run/node": "^2.5.1",
        "@remix-run/react": "^2.5.1",
        "@remix-run/serve": "^2.5.1",
        "isbot": "^4.1.0",
        "match-sorter": "^6.3.1",
        "react": "^18.2.0",
        "react-dom": "^18.2.0",
        "sort-by": "^0.0.2",
        "tiny-invariant": "^1.3.1"
    },
    "devDependencies": {
        "@remix-run/dev": "^2.3.1",
        "@types/react": "^18.2.20",
        "@types/react-dom": "^18.2.7",
        "@typescript-eslint/eslint-plugin": "^6.13.0",
        "@typescript-eslint/parser": "^6.13.0",
        "eslint": "^8.47.0",
        "eslint-import-resolver-typescript": "^3.6.1",
        "eslint-plugin-import": "^2.29.1",
        "eslint-plugin-jsx-a11y": "^6.8.0",
        "eslint-plugin-react": "^7.33.2",
        "eslint-plugin-react-hooks": "^4.6.0",
        "typescript": "^5.1.6"
    },
    "engines": {
        "node": ">=18.0.0"
    }
}
```

J Comparison with DALICC-Framework

License 1	License 2	DALICC	Recommended	Reason
CCO	CCO	No conflict	No	
CCO	CC-BY-4.0	Conflict	Yes	
CCO	CC-BY-SA-4.0	Conflict	Yes	
CCO	CC-BY-NC-4.0	Conflict	Yes	
CCO	CC-BY-ND-4.0	Conflict	Yes	
CCO	CC-BC-NC-SA-4.0	Conflict	Yes	
CCO	CC-BY-NC-ND-4.0	Conflict	Yes	
CCO	ODC-BY	Conflict	Yes	
CCO	ODC-PDDL	No conflict	No	
CCO	ODC-ODbL	Conflict	Yes	
CCO	Open Government License v3	No conflict	Yes	Actions
CC-BY-4.0	CCO	No conflict	No	
CC-BY-4.0	CC-BY-4.0	No conflict	No	
CC-BY-4.0	CC-BY-SA-4.0	No conflict	Yes	Share-Alike
CC-BY-4.0	CC-BY-NC-4.0	Conflict	Yes	
CC-BY-4.0	CC-BY-ND-4.0	Conflict	Yes	
CC-BY-4.0	CC-BC-NC-SA-4.0	Conflict	Yes	
CC-BY-4.0	CC-BY-NC-ND-4.0	Conflict	Yes	
CC-BY-4.0	ODC-BY	No conflict	Yes	Share-Alike

License 1	License 2	DALICC	Recommended	Reason
CC-BY-4.0	ODC-PDDL	No conflict	No	
CC-BY-4.0	ODC-ODbL	Conflict	Yes	
CC-BY-4.0	Open Government License v3	No conflict	Yes	Actions
CC-BY-SA-4.0	CCO	No conflict	No	
CC-BY-SA-4.0	CC-BY-4.0	No conflict	No	
CC-BY-SA-4.0	CC-BY-SA-4.0	No conflict	No	
CC-BY-SA-4.0	CC-BY-NC-4.0	Conflict	Yes	
CC-BY-SA-4.0	CC-BY-ND-4.0	Conflict	Yes	
CC-BY-SA-4.0	CC-BC-NC-SA-4.0	Conflict	Yes	
CC-BY-SA-4.0	CC-BY-NC-ND-4.0	Conflict	Yes	
CC-BY-SA-4.0	ODC-BY	No conflict	Yes	Share-Alike
CC-BY-SA-4.0	ODC-PDDL	No conflict	No	
CC-BY-SA-4.0	ODC-ODbL	No conflict	Yes	Share-Alike
CC-BY-SA-4.0	Open Government License v3	No conflict	Yes	Actions
CC-BY-NC-4.0	CCO	No conflict	No	
CC-BY-NC-4.0	CC-BY-4.0	No conflict	No	
CC-BY-NC-4.0	CC-BY-SA-4.0	No conflict	Yes	Share-Alike
CC-BY-NC-4.0	CC-BY-NC-4.0	No conflict	No	
CC-BY-NC-4.0	CC-BY-ND-4.0	Conflict	Yes	

License 1	License 2	DALICC	Recommended	Reason
CC-BY-NC-4.0	CC-BC-NC-SA-4.0	No conflict	Yes	Share-Alike
CC-BY-NC-4.0	CC-BY-NC-ND-4.0	Conflict	Yes	
CC-BY-NC-4.0	ODC-BY	No conflict	Yes	Share-Alike
CC-BY-NC-4.0	ODC-PDDL	No conflict	No	
CC-BY-NC-4.0	ODC-ODbL	Conflict	Yes	
CC-BY-NC-4.0	Open Government License v3	No conflict	Yes	Actions
CC-BY-ND-4.0	CCO	No conflict	Yes	Derivative
CC-BY-ND-4.0	CC-BY-4.0	No conflict	Yes	Derivative
CC-BY-ND-4.0	CC-BY-SA-4.0	No conflict	Yes	Derivative
CC-BY-ND-4.0	CC-BY-NC-4.0	Conflict	Yes	
CC-BY-ND-4.0	CC-BY-ND-4.0	No conflict	Yes	Derivative
CC-BY-ND-4.0	CC-BC-NC-SA-4.0	No conflict	Yes	Derivative
CC-BY-ND-4.0	CC-BY-NC-ND-4.0	Conflict	Yes	
CC-BY-ND-4.0	ODC-BY	No conflict	Yes	Derivative
CC-BY-ND-4.0	ODC-PDDL	No conflict	Yes	Derivative
CC-BY-ND-4.0	ODC-ODbL	Conflict	Yes	
CC-BY-ND-4.0	Open Government License v3	No conflict	Yes	Derivative
CC-BY-NC-SA-4.0	CCO	No conflict	No	
CC-BY-NC-SA-4.0	CC-BY-4.0	No conflict	No	

License 1	License 2	DALICC	Recommended	Reason
CC-BY-NC-SA-4.0	CC-BY-SA-4.0	No conflict	Yes	Share-Alike
CC-BY-NC-SA-4.0	CC-BY-NC-4.0	No conflict	No	
CC-BY-NC-SA-4.0	CC-BY-ND-4.0	Conflict	Yes	
CC-BY-NC-SA-4.0	CC-BC-NC-SA-4.0	No conflict	No	
CC-BY-NC-SA-4.0	CC-BY-NC-ND-4.0	Conflict	Yes	
CC-BY-NC-SA-4.0	ODC-BY	No conflict	Yes	Share-Alike
CC-BY-NC-SA-4.0	ODC-PDDL	No conflict	No	
CC-BY-NC-SA-4.0	ODC-ODbL	No conflict	Yes	Actions
CC-BY-NC-SA-4.0	Open Government License v3	No conflict	Yes	Actions
CC-BY-NC-ND-4.0	CCO	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	CC-BY-4.0	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	CC-BY-SA-4.0	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	CC-BY-NC-4.0	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	CC-BY-ND-4.0	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	CC-BC-NC-SA-4.0	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	CC-BY-NC-ND-4.0	No conflict	Yes	Derivative

License 1	License 2	DALICC	Recommended	Reason
CC-BY-NC-ND-4.0	ODC-BY	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	ODC-PDDL	No conflict	Yes	Derivative
CC-BY-NC-ND-4.0	ODC-ODbL	Conflict	Yes	
CC-BY-NC-ND-4.0	Open Government License v3	No conflict	Yes	Derivative
ODC-BY	CCO	No conflict	Yes	Actions
ODC-BY	CC-BY-4.0	No conflict	Yes	Actions
ODC-BY	CC-BY-SA-4.0	No conflict	Yes	Actions
ODC-BY	CC-BY-NC-4.0	Conflict	Yes	
ODC-BY	CC-BY-ND-4.0	Conflict	Yes	
ODC-BY	CC-BC-NC-SA-4.0	Conflict	Yes	
ODC-BY	CC-BY-NC-ND-4.0	Conflict	Yes	
ODC-BY	ODC-BY	No conflict	No	
ODC-BY	ODC-PDDL	No conflict	No	
ODC-BY	ODC-ODbL	No conflict	Yes	Actions
ODC-BY	Open Government License v3	No conflict	Yes	Actions
ODC-PDDL	CCO	No conflict	Yes	Actions
ODC-PDDL	CC-BY-4.0	Conflict	Yes	
ODC-PDDL	CC-BY-SA-4.0	Conflict	Yes	
ODC-PDDL	CC-BY-NC-4.0	Conflict	Yes	

License 1	License 2	DALICC	Recommended	Reason
ODC-PDDL	CC-BY-ND-4.0	Conflict	Yes	
ODC-PDDL	CC-BC-NC-SA-4.0	Conflict	Yes	
ODC-PDDL	CC-BY-NC-ND-4.0	Conflict	Yes	
ODC-PDDL	ODC-BY	Conflict	Yes	
ODC-PDDL	ODC-PDDL	No conflict	No	
ODC-PDDL	ODC-ODbL	Conflict	Yes	
ODC-PDDL	Open Government License v3	No conflict	Yes	Actions
ODC-ODbL	CCO	No conflict	Yes	Actions
ODC-ODbL	CC-BY-4.0	No conflict	Yes	Actions
ODC-ODbL	CC-BY-SA-4.0	No conflict	Yes	Actions
ODC-ODbL	CC-BY-NC-4.0	No conflict	Yes	Actions
ODC-ODbL	CC-BY-ND-4.0	Conflict	Yes	
ODC-ODbL	CC-BC-NC-SA-4.0	No conflict	Yes	Actions
ODC-ODbL	CC-BY-NC-ND-4.0	Conflict	Yes	
ODC-ODbL	ODC-BY	No conflict	Yes	Actions
ODC-ODbL	ODC-PDDL	No conflict	No	
ODC-ODbL	ODC-ODbL	No conflict	No	
ODC-ODbL	Open Government License v3	No conflict	Yes	Actions

License 1	License 2	DALICC	Recommended	Reason
Open Government License v3	CCO	No conflict	No	
Open Government License v3	CC-BY-4.0	No conflict	Yes	Actions
Open Government License v3	CC-BY-SA-4.0	No conflict	Yes	Actions
Open Government License v3	CC-BY-NC-4.0	Conflict	Yes	
Open Government License v3	CC-BY-ND-4.0	Conflict	Yes	
Open Government License v3	CC-BC-NC-SA-4.0	Conflict	Yes	
Open Government License v3	CC-BY-NC-ND-4.0	Conflict	Yes	
Open Government License v3	ODC-BY	No conflict	Yes	Actions
Open Government License v3	ODC-PDDL	No conflict	No	
Open Government License v3	ODC-ODbL	Conflict	Yes	
Open Government License v3	Open Government License v3	No conflict	No	

References

Abelson, H., Adida, B., Linksvayer, M., & Yergler, N. (2008). Ccrel: The creative commons rights expression language. https://wiki.creativecommons.org/images/d/d6/Ccrel-1.0.pdf

Almeida, D. A., Murphy, G. C., Wilson, G., & Hoye, M. (2017). Do software developers understand open source licenses? *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, 1–11. https://doi.org/10.1109/ICPC.2017.7

Bowen, G. A. (2008). Naturalistic inquiry and the saturation concept: A research note. *Qualitative Research*, *8*(1), 137–152. https://doi.org/10.1177/1468794107085301

Brickley, D., Burgess, M., & Noy, N. (2019). Google dataset search: Building a search engine for datasets in an open web ecosystem. *The World Wide Web Conference*. https://doi.org/10.1145/3308558.3313685

Cabrio, E., Palmero Aprosio, A., & Villata, S. (2014). These are your rights. In V. Presutti (Ed.), *The semantic web: Trends and challenges* (pp. 255–269). Springer. https://doi.org/10.1007/978-3-319-07443-6_18

Cardellino, C., Villata, S., Gandon, F., Governatori, G., Lam, H.-P., & Rotolo, A. (2014). *Licentia: A tool for supporting users in data licensing on the web of data masterthesis*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=6c9d9e353d7332c1024a9626f01514a3f736f70e

Ermilov, I., & Pellegrini, T. (2015). Data licensing on the cloud. *Proceedings of the 11th International Conference on Semantic Systems*. https://doi.org/10.1145/2814864.2814878

Gangadharan, G. R., Weiss, M., D'Andrea, V., & Iannella, R. (2007). Service license composition and compatibility analysis. In B. J. Krämer, K.-J. Lin & P. Narasimhan (Eds.), *Service-oriented computing - icsoc 2007* (pp. 257–269). Springer. https://doi.org/10.1007/978-3-540-74974-5_21

German, D. M., Di Penta, M., & Davies, J. (2010). Understanding and auditing the licensing of open source software distributions. *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, 84–93. https://doi.org/10.1109/ICPC.2010.48

Governatori, G., Lam, H.-P., Rotolo, A., Villata, S., Atemezing, G., & Gandon, F. (2014). Live: A tool for checking licenses compatibility between vocabularies and data. http://ceur-ws.org/vol-1272/paper_62.pdf

Governatori, G., Lam, H.-P., Rotolo, A., Villata, S., & Gandon, F. (2013). Heuristics for licenses composition. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ddeabb1908cc952c1b2f2dc59a72bf9d8f6364a2

Governatori, G., Rotolo, A., Villata, S., & Gandon, F. (2013). One license to compose them all. In H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. Noy, C. Welty & K. Janowicz (Eds.), *The semantic web - iswc 2013* (pp. 151–166). Springer. https://doi.org/10.1007/978-3-642-41335-3_10

Guth, S. (2003). Rights expression languages. In E. Becker (Ed.), *Digital rights management* (pp. 101–112, Vol. 2770). Springer. https://doi.org/10.1007/10941270_8

Jansen, H. (2010). The logic of qualitative survey research and its position in the field of social research methods: Forum qualitative sozialforschung / forum: Qualitative social research, vol 11, no 2 (2010): Visualising migration and social division: Insights from social sciences and the visual arts. https://doi.org/10.17169/fqs-11.2.1450

JValue-Team. (2024). Jayvee - core concepts. https://jvalue.github.io/jayvee/docs/user/core-concepts

K. Peffers, T. Tuunanen, M.A. Rothenberger & S. Chatterjee. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. https://www.researchgate.net/publication/284503626_A_design_science_research_methodology_for_information_systems_research

Krötzsch, M., & Speiser, S. (2011). Sharealike your data: Self-referential usage policies for the semantic web. In L. Aroyo (Ed.), *The semantic web - iswc 2011* (pp. 354–369). Springer. https://doi.org/10.1007/978-3-642-25073-6_23

Ministerium für Wirtschaft, Innovation, Digitalisierung und Energie des Landes Nordrhein-Westfalen & Beauftragte der Landesregierung für Informationstechnik (CIO) / Geschäftsstelle Open.NRW (Eds.). (2019). Datenlizenzen für open government data - rechtliches kurzgutachten: Handreichung zu den nutzungsrechteregelungen gebräuchlicher open data lizenzen und empfehlungen für ihren einsatz. https://open.nrw/system/files/media/document/file/opennrw_rechtl_gutachten_datenlizenzen_lowres_web.pdf

Moreau, B., Serrano-Alvarado, P., Perrin, M., & Desmontils, E. (2019). Modelling the compatibility of licenses. In P. Hitzler, M. Fernández, K. Janowicz, A. Zaveri, A. J. Gray, V. Lopez, A. Haller & K. Hammar (Eds.), *The semantic web* (pp. 255–269). Springer. https://doi.org/10.1007/978-3-030-21348-0_17

Mpeg-21 rights expression language. (2005). https://mpeg.chiariglione.org/standards/mpeg-21/rights-expression-language

Pellegrini, T., Mireles, V., Steyskal, S., Panasiuk, O., Fensel, A., & Kirrane, S. (2018). Automated rights clearance using semantic web technologies: The dalicc framework. In T. Hoppe, B. Humm & A. Reibold (Eds.), *Semantic applications* (pp. 203–218). Springer Vieweg. https://doi.org/10.1007/978-3-662-55433-3_14

Philip Rebbe. (2024). Prebbe/license-comparer: V0.0.4-thesis. https://doi.org/10.5281/ZENODO.10926133

Russell, S. J., & Norvig, P. (2022). *Artificial intelligence: A modern approach* (Fourth edition, global edition). Pearson.

Schmitz, P.-E., & Cacciaguerra Ranghieri, G. (2019). Joinup licensing assistant – white paper: Project of a legal operability tool implemented as a "solution" in the framework of joinup.eu. https://joinup.ec.europa.eu/sites/default/files/document/2023-02/Joinup%20Licensing%20Assistant%20-%20White%20Paper_v1.01_1.pdf

Torres-Hostench, O., & Salinas, C. B. (2015). Technology and e-resources for legal translators: The law 10n project. In *Conducting research in translation technologies* (pp. 285–306).

Villata, S., & Gandon, F. (2012). Licenses compatibility and composition in the web of data. https://inria.hal.science/hal-01171125/document

W3C. (2018). Odrl vocabulary & expression 2.2. https://www.w3.org/TR/2018/REC-odrl-vocab-20180215/

W3C Permissions and Obligations Expression Working Group. (2018). Odrl information model 2.2: W3c recommendation 15 february 2018 (W3C, Ed.). https://www.w3.org/TR/odrl-model/

Wolter, T., Barcomb, A., Riehle, D., & Harutyunyan, N. (2023). Open source license inconsistencies on github. *ACM Transactions on Software Engineering and Methodology*, *32*(5), 1–23. https://doi.org/10.1145/3571852

Yi-Hsuan Lin, Tung-Mei Ko, Tyng-Ruey Chuang & Kwei-Jay Lin. (2006). Open source licenses and the creative commons framework: License selection and comparison. *Journal of Information Science and Engineering*, *22*(1), 1–17. https://www.researchgate.net/profile/kwei-jay-lin/publication/220587883_open_source_licenses_and_the_creative_commons_framework_license_selection_and_comparison