CaLi: A Lattice-Based Model for License Classifications

Benjamin Moreau, Patricia Serrano-Alvarado, Emmanuel Desmontils

CaLi: A Lattice-Based Model for License Classifications

Benjamin Moreau
OpenDataSoft
Paris, France
Benjamin.Moreau@opendatasoft.com

Patricia Serrano-Alvarado
Nantes University, LS2N, CNRS, UMR6004
Nantes, France
Patricia.Serrano-Alvarado@ls2n.fr

Emmanuel Desmontils
Nantes University, LS2N, CNRS, UMR6004
Nantes, France
Emmanuel.Desmontils@ls2n.fr

ABSTRACT

Web applications facilitate combining resources (linked data, web services, source code, documents, etc.) to create new ones. For a resource producer, choosing the appropriate license for a combined resource involves choosing a license compliant with all the licenses of combined resources and controlling the reusability of the resulting resource through the compatibility of its license. The risk is either, to choose a license too restrictive making the resource difficult to reuse, or to choose a not enough restrictive license that will not sufficiently protect the resource. Finding the right trade-off between compliance and compatibility is a difficult process. An automatic classification over licenses protecting resources would facilitate this task. Our research question is: *given a license l_i, how to automatically position l_i over a set of licenses in terms of compatibility and compliance?* We propose CaLi, a lattice-based model to classify licenses. CaLi classifications are able to answer questions like, "what are the licenses with which l_i is compliant?" and "what are the licenses with which l_i is compatible?". We show the usability of a CaLi classification through a prototype of a license-based search engine for the Web of Data. Our work is a step forward to facilitate and encourage the reuse of license compliant resources in the Web.

KEYWORDS

Licenses, usage control, privacy, linked data, RDF, ODRL

1 INTRODUCTION

Web applications facilitate combining resources (linked data, web services, source code, documents, etc.) to create new ones. Resource producers should systematically associate licenses with resources before sharing or publishing them. Licenses specify precisely the conditions of reuse of resources, i.e., what actions are *permitted*, *obliged* and *prohibited* when using the resource.

For a resource producer, choosing the appropriate license for a combined resource and choosing the appropriate licensed resources for a combination is a difficult process. It involves choosing a license compliant with all the licenses of combined resources as well as controlling the reusability of the resulting resource through the compatibility of its license. The risk is either, to choose a license too restrictive making the resource difficult to reuse, or to choose a not enough restrictive license that will not sufficiently protect the resource.

We consider a simplified definition of compliance inspired by works like [5–7, 13]: *a license l_j is compliant with a license l_i if a resource licensed under l_i can be licensed under l_j without violating l_i.* If a license l_j is compliant with l_i then we consider that l_i is compatible with l_j and that resources licensed under l_i are reusable with resources licensed under l_j.

Compatibility of licenses can be verified through the actions permitted, obliged and prohibited. In general, when l_i is compatible with l_j, the obligations and prohibitions of l_i exist in l_j, and l_j has at most the permissions of l_i. Figure 1 shows an excerpt of three Creative Commons (CC)[1] licenses described in RDF using the ODRL vocabulary[2]. We can notice that:

- CC BY is compatible with CC BY, CC BY-NC and CC BY-NC-ND;
- CC BY-NC is compatible with CC BY-NC, CC BY-NC-ND; and
- CC BY-NC-ND is compatible with CC BY-NC-ND.

Imagine a search engine that can find linked data based on their licenses. Consider this search "*find all datasets concerning bikes that can be reused under the CC BY-NC license*". The answer must contain datasets with licenses compatible with CC BY-NC that (in our example) are less or as restrictive as it. Consider this other search that will find out the reusability of the combined resource "*find all datasets that can reuse a dataset protected by CC BY-NC*". The answer must contain datasets with licenses compliant with CC BY-NC that (in our example) are at least as restrictive as it is (CC BY-NC and CC BY-NC-ND). Answers are order form the least to the most restrictive (or vice versa).

We can imagine a similar search engine for services such as GitHub[3] or APISearch[4] that could find source code repositories or APIs protected by licenses that are compatible or compliant with a specific license.

We argue that a model for license classifications would make possible the existence of such a license-based search engines. Our research question is: *given a license l_i, how to automatically position l_i over a set of licenses in terms of compatibility and compliance?* That is, given a set of licenses, what are the licenses that precede and follow l_i in terms of compatibility and compliance?

Our contribution is twofold: (1) inspired by access control models [4, 11], we propose CaLi (ClAssification of LIcenses), a lattice-based model for license classifications and (2) a prototype of a search engine based on a CaLi classification

[1] https://creativecommons.org/
[2] https://www.w3.org/TR/odrl-model/
[3] https://github.com/
[4] http://apis.io/

```
:cc-by4.0 a odrl:Policy ;
 rdfs:label "CC BY";
 ...
 odrl:permission
   [odrl:action
     cc:Distribution ,
     cc:Reproduction ,
     cc:CommercialUse ,
     cc:DerivativeWorks ;];
 odrl:duty
  [odrl:action
    cc:Notice ,
    cc:Attribution ;].
```

(a) CC BY

```
:cc-by-nc4.0 a odrl:Policy ;
 rdfs:label "CC BY-NC";
 ...
 odrl:permission
    [odrl:action
      cc:Distribution ,
      cc:Reproduction ,
      cc:DerivativeWorks ;];
 odrl:duty
    [odrl:action
      cc:Notice ,
      cc:Attribution ;]
 odrl:prohibition
   [odrl:action
     cc:CommercialUse].
```

(b) CC BY-NC

```
:cc-by-nc-nd4.0 a odrl:Policy
     ;
 rdfs:label "CC BY-NC-ND";
 ...
 odrl:permission
  [odrl:action
    cc:Distribution ,
    cc:Reproduction ;];
 odrl:duty
  [odrl:action
    cc:Notice ,
    cc:Attribution ;];
 odrl:prohibition
  [odrl:action
    cc:CommercialUse ,
    cc:DerivativeWorks].
```

(c) CC BY-NC-ND

Figure 1: Three Creative Commons licenses described in RDF.

that allows to find datasets of the Web of Data whose licenses are compatible or compliant with a target license.

Our work is a step forward in creating tools that facilitate the creation and reuse of license compliant resources. This kind of effort will encourage the publication and reuse of data in the Web of Data.

The remainder of the paper is as follows. Section 2 discuses related works, Section 3 introduces the CaLi model, Section 4 illustrates the usability of our model, Section 5 shows the implementation of a license-based search engine for linked data, and Section 6 concludes.

2 RELATED WORK

This section analyses several approaches and tools related to our research question.

2.1 Machine-readable licenses

Automatic license classification requires machine-readable licenses. License expression languages such as CC REL[5], ODRL, or L4LOD[6] enable fine-grained RDF description of licenses. Works like [10] and [1] use natural language processing to automatically generate RDF licenses from licenses described in natural language. Other works such as [9] or [2] propose a set of well-known licenses in RDF described in CC REL and ODRL. In this work we do not address these issues, we consider that there exist consistent licenses described in RDF.

[5]https://creativecommons.org/ns
[6]https://ns.inria.fr/l4lod/

2.2 Tools for license compliant resources

There exist some tools to facilitate the creation of license compliant resources. TLDRLegal[7], CC Choose[8] and ChooseALicense[9] help users to choose actions to form a license for their resources.

CC search[10] allows users to find images licensed under Creative Commons licenses that can be commercialized, modified, adapted, or built upon.

Web2rights proposes a tool to check compatibility among Creative Commons licenses[11].

Finally, Licentia[12], based on defeasible deontic logic to reason over the licenses, proposes a web service to find licenses compatible with a set of permissions, obligations and prohibitions chosen by the user.

From these tools, only Licentia uses machine-readable licenses[13] described in RDF using ODRL. Unfortunately, Licentia is not able to order licenses in terms of compatibility or compliance.

2.3 License compatibility and license combination

The easiest way to choose a license for a combined resource is to create a new one by combining all resource licenses to combine. Several works address the problem of license compatibility and license combination.

In Web services, [5] proposes a framework that analyzes compatibility of licenses to verify if two services are compatible and then generate the composite service license.

[8] addresses the problem of license preservation during the combination of digital resources (e.g., music, data, picture,

[7]https://tldrlegal.com/
[8]https://creativecommons.org/choose/
[9]https://choosealicense.com/
[10]https://ccsearch.creativecommons.org/
[11]http://www.web2rights.com/creativecommons/
[12]http://licentia.inria.fr/
[13]purl.org/NET/rdflicense

etc.) in a collaborative environment. Licenses of combined resources are combined into a new one.

In the Web of Data, [13] proposes a framework to check compatibility among CC REL licenses. If licenses are compatible, a new license compliant with combined ones is generated. [6] formally defines the combination of licenses using deontic logic.

[12] proposes PrODUCE, an approach to combine usage policies taking into account the usage context. These works focus on combining operators for automatic license combination but do not propose to position a license over a set of licenses.

Such operators will be very valuable in the design of an automatic classification.

2.4 License classification

Concerning the problem of license classification to facilitate the selection of a license, [3] uses Formal Concept Analysis (FCA) to generate a lattice of actions. Once pruned and annotated, this lattice can be used to classify licenses in terms of features. This classification reduces the selection of a license to an average of three to five questions. However, this work does not address the problem of license combination and license compatibility. Moreover, FCA is not suitable to generate complex compatibility relations among licenses. FCA defines a derivation operator on objects that returns a set of attributes shared by the objects. We consider that the set of actions in common of two licenses is not enough to infer compatibility. If applied to our introductory license compatibility example, FCA can only work with permissions but not with obligations and prohibitions. That is because l_i is compatible with l_j if l_i permissions are a superset of l_j permissions, but regarding obligations and prohibitions, l_i is compatible with l_j if they are a subset of those of l_j.

In the context of Free Open Source Software (FOSS), [7] proposes an approach, based on a directed acyclic graph, to detect license violations in existing software packages. It considers that license l_i is compatible with l_j if the graph contains a path from l_i to l_j. However, as such a graph is build from a manual interpretation of each license, its generalization and automation is not possible.

2.5 Lattice-based access control

In the domain of access control, [4] proposes a lattice model of secure information flow. This model classifies security classes with associated resources. Like in the compatibility graph of [7], security class sc_i is compatible with sc_j if the lattice contains a path from sc_i to sc_j. Thus, this path represents the authorized flow of resources (e.g., resource r_i protected with sc_i can flow to a resource protected by sc_j without violating sc_i.). The lattice can be generated automatically through a pairwise combination of all security classes if sc_i combined with sc_k gives sc_j where sc_i and sc_k are both compatible with sc_j. [11] describes several models based on this approach but none focuses on classifying licenses.

None of these works answers our research question. They do not allow to automatically position a license over a set of licenses in terms of both compatibility and compliance.

In our work we propose a lattice-based model inspired by [4]. Existing combining operators like [5, 6, 12, 13] make feasible the automatic generation of a lattice. Our model allows users to easily find and compare licenses as well as licensed resources. This model is independent on any license combination approach, license description language and licensed resources so that it can be used in a wide variety of domains.

3 CALI: A LATTICE-BASED LICENSE MODEL

CaLi is a model that allows to express license classifications with a lattice structure. The model considers a finite set of licenses, a combining operator, a compatibility relation over licenses, resources and their association with licenses, and a constraint set. The combining operator applied to the set of licenses produces a lattice that expresses the compliance and compatibility among licenses.

Next section introduces the CaLi model, then we describe its lattice structure and we finish with a simple example of a CaLi-based classification.

3.1 Model description

The license classification model is introduced by Definition 3.1.

Definition 3.1. CaLi $= \langle L, \bigoplus, \rightarrow, R, \mapsto, C \rangle$.

$L = \{l_1, ..., l_n\}$ is a set of licenses.

$\bigoplus$ is a license combining operator.

$\rightarrow$ is the *compatible with* relation defined on pairs of licenses.

$R = \{r_1, ..., r_m\}$ is a set of resources.

$\mapsto$ is the *protected by* relation between R and L.

$C = \langle C_L, C_\rightarrow \rangle$ is a finite set of constraints to express *viability*. C_L is a set of constraints over L and $C_\rightarrow$ is a set of constraints over $\rightarrow$ in $L \times L$.

A license is considered as a set of actions. Actions used in L can be taken from $A = \{a_1, ..., a_n\}$, a set of actions that can be applied on resources.

$\bigoplus$ is an associative and commutative binary operator that specifies for any pair of licenses, the combined license that is compliant with both licenses. For licenses l_i and l_k, if $l_i \bigoplus l_k = l_j$ then l_j is compliant with l_i and l_k.

For licenses l_i and l_j, we write $l_i \rightarrow l_j$ iff l_i is compatible with l_j (or l_j is compliant with l_i). $l_i \rightarrow l_j$ implies $l_i \bigoplus l_j = l_j$, i.e., l_j overcomes l_i.

Concerning R, we write $r_i \mapsto l_i$ when the resource r_i is protected by l_i. For $r_i \mapsto l_i$, if l_i is compatible with l_j then r_i can be protected by l_j.

In this work, we introduce the concepts of viability of a license and viability of a $\rightarrow$ relation.

- *A viable license is a license that allows the licensed resource to be used.* A license l_i is viable iff it respects

all C_L constraints. We define a *constraint on licenses* as an application $\omega_L : L \rightarrow Boolean$ which associates each license of L with either True or False.

- *A viable compatible with relation, $l_i \rightarrow l_j$, is a relation that allows a resource licensed under l_i to be licensed under l_j.* A $\rightarrow$ relation between two licenses is viable iff it respects all $C_{\rightarrow}$ constraints. We define a *constraint on $\rightarrow$ relations* as an application $\omega_{\rightarrow} : L \times L \rightarrow Boolean$ which associates each $\rightarrow$ with either True or False.

3.2 Lattice structure $\langle L, \rightarrow, \bigoplus \rangle$

The set of licenses L, the *compatible with* relation $\rightarrow$ and the combining operator $\bigoplus$ form a lattice with the following properties:

(1) $\langle L, \rightarrow \rangle$ is a partially ordered set.
(2) L is finite.
(3) L has an infimum (or greatest lower bound) I such that $I \rightarrow l_i \ \forall l_i \in L$.
(4) $\bigoplus$ is a totally defined least upper bound operator on L.

Demonstrations.

Property (1) is demonstrated by showing that the relation $\rightarrow$ is reflexive, transitive and antisymmetric.

$\forall l_i, l_j, l_k \in L$:

(a) $l_i \rightarrow l_i$ (reflexive), i.e., a license is compatible with itself.
(b) $l_i \rightarrow l_j$ and $l_j \rightarrow l_k \Rightarrow l_i \rightarrow l_k$ (transitive), i.e., if l_i is compatible with l_j and l_j is compatible with l_k then l_i is compatible with l_k.
(c) $l_i \rightarrow l_j$ and $l_j \rightarrow l_i \Rightarrow l_i = l_j$ (antisymmetric), i.e., if two licenses are compatible with each other, then they are the same license.

Property (2) is demonstrated by the fact that the set of actions A is finite. If a license is a set of actions and the license combining operator $\bigoplus$ is a least upper bound operator, then L is finite.

Property (3) can be made without loss of generality considering as infimum a license I with zero, one or more actions.

Property (4) is demonstrated by showing that every pair of licenses is combined and the result is a combined license (i.e., $\bigoplus$ is totally defined) and that the license combining operator is a least upper bound, that is, $\forall l_i, l_j, l_k \in L$:

(a) $l_i \rightarrow l_i \bigoplus l_j$ and $l_j \rightarrow l_i \bigoplus l_j$.
(b) $l_i \rightarrow l_k$ and $l_j \rightarrow l_k \Rightarrow l_i \bigoplus l_j \rightarrow l_k$.

For *(a)* without loss of generality we can consider that every pair of licenses is combined and the combining operator is commutative and associative. In *(b)*, for any two licenses l_i and l_j the least upper bound is $l_i \bigoplus l_j$. This property implies the existence of a supremum S (or least upper bound) that is a $l_1 \bigoplus l_2 \bigoplus ... \bigoplus l_n$ of n licenses.

Our model with the lattice structure allow to answer our research question *given a license l_i, how to automatically position l_i over a set of licenses in terms of compatibility and compliance?* For a license l_i, the licenses which precede it in the lattice are compatible with it, and the licenses which follow it in the lattice are compliant with this one. This is equivalent to say that l_i is compliant with the licenses which precede it in the lattice and it is compatible with the licenses which follow it in the lattice.

When the constraint set C is not empty, $C \neq \emptyset$, l_i is compatible with l_j if there exists a path from l_i to l_j where all *compatible with* relations and all licenses are viable.

3.3 Example 1

Consider a classification where *licenses are simplified to a set of prohibitions.* The set A of actions is *{read, modify, distribution}*. The license combining operator consists in the union of prohibitions. Resources are datasets in R. This $CaLi$ classification is described as follows.

$L = powerset(A)$ *where each l_i is a set of prohibited actions Pr_{l_i} taken from A =*{read, modify, distribution}.
$\bigoplus$ *is the license combining operator where $l_i \bigoplus l_j \equiv Pr_{l_i} \cup Pr_{l_j}$.*
$\rightarrow$ *is the compatible with relation stating that $l_i \rightarrow l_j$ iff $Pr_{l_i} \subseteq Pr_{l_j}$.*
$R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$
$\mapsto$ *is the* protected by *relation such that $r_1, r_2 \mapsto l_0; r_3 \mapsto l_2; r_4 \mapsto l_3; r_5 \mapsto l_4; r_6 \mapsto l_6; r_7 \mapsto l_7$.*
$C = \langle C_L, C_{\rightarrow} \rangle$ *where $C_L = \{\omega_{L_1}\}$ and $C_{\rightarrow} = \emptyset$ (all $\rightarrow$ relations are viable).*

In this example, the constraint that characterizes viable licenses is, *if the $modify$ action is prohibited then the $read$ action must be too*:

$$\omega_{L_1}(l_i) = \begin{cases} False & \text{if } read \in Pr_{l_i} \text{ and } modify \notin Pr_{l_i}; \\ True & \text{otherwise.} \end{cases}$$

Figure 2 shows a visual representation of the lattice of this example. In the lattice, $I = \emptyset$ and $S = A$. The combining operator generates the powerset of the first level of the lattice. The first level of the lattice is composed by the minimal set of licenses that allows CaLi to generate all other licenses. Due to the combining operator, the minimal set for this example is composed of licenses having only one action $\in A$, i.e., $|level_1| = |A|$. The total number of licenses in L is $2^{|level_1|} = 2^{|A|}$. Non-viable licenses, l_1 and l_5, are part of the lattice but they cannot protect resources.

To the question, *"find resources whose licenses are compatible with l_4"*, this lattice allows to answer r_1, r_2, r_3 and r_5 which are protected by l_0 and l_2 which precede l_4 in the classification as well as by l_4 itself. That means that resources protected by l_0, l_2 and l_4 can be reusable with resources protected by l_4.

Similarly, to the question, *"find resources whose licenses are compliant with l_4"*, this lattice allows to answer r_5 and r_7, which respectively are protected by license l_7 as well as by l_4 itself. That means that resources protected by l_4 are reusable with resources protected by l_4 and l_7.

Thus, analyzing compatibility and compliance allow to evaluate the reusability of resources.

Licenses above l_4 are more restrictive than l_4 and licenses below l_4 are less restrictive than l_4. The least restrictive compliant license for l_4 is the direct following one l_7. The most

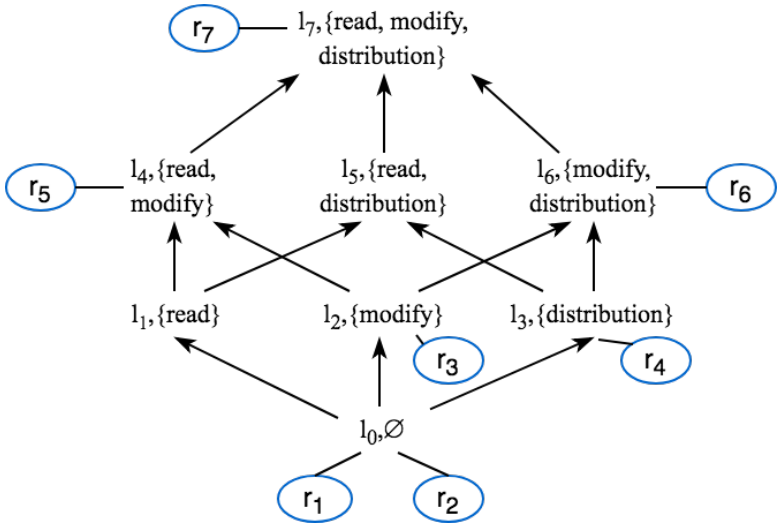

Figure 2: Visual representation of the CaLi classification of Example 1.

restrictive compatible licenses for l_4 are the direct preceding ones l_1 and l_2.

This example of classification illustrates our model with a very simple combining operator. It shows how our model answers the two motivating questions of this work. Next section introduces a CaLi classification of licenses used to publish creative works on the Web with a realistic combining operator.

4 LICENSE CLASSIFICATION FOR ACTIONS OF CREATIVE COMMONS

Creative Commons (CC) proposes easy-to-understand licenses that are widely used when publishing resources on the web (creative content, linked data[14], etc.). Licenses are composed of at most 7 actions distributed in permissions, obligations and prohibitions: *cc:Distribution, cc:Reproduction, cc:DerivativeWorks, cc:CommercialUse, cc:Notice, cc:Attribution, cc:ShareAlike*. CC REL is the language to express Creative Commons licenses in RDF.

CC proposes only 7 licenses but with the CaLi model we search to produce a complete classification of all possible licenses using these 7 actions.

4.1 Description of a CC classification based on CaLi

In this CC classification, we consider that each license has a set of permissions, obligations and prohibitions. Licenses are *consolidated*, that is, each term of the set of actions A must appear in each license. The idea is that each action be permitted, obliged or prohibited in each license.

The license combining operator makes the intersection of permissions, the union of obligations and the union of prohibitions of concerned licenses.

[14]https://ns.inria.fr/l4lod/v2/l4lod_v2.html

$L = \{\langle P, O, Pr\rangle\}$; *$P, O, Pr$ are sets of permissions, obligations and prohibitions containing actions $\in A$; the union of these sets is A.*
A ={cc:Distribution, cc:Reproduction, cc:DerivativeWorks, cc:CommercialUse, cc:Notice, cc:Attribution, cc:ShareAlike}.
$\bigoplus$ *is the license combining operator, $l_i \bigoplus l_j \equiv \langle P_{l_i} \cap P_{l_j}, O_{l_i} \cup O_{l_j}, Pr_{l_i} \cup Pr_{l_j}\rangle$ where P_{l_i} is the permission set of l_i, O_{l_i} is the obligation set of l_i, and Pr_{l_i} is the prohibition set of l_i.*
$\rightarrow$ *is the compatible relation stating that $l_i \rightarrow l_j$ iff $P_{l_i} \supseteq P_{l_j}$ and $O_{l_i} \subseteq O_{l_j}$ and $Pr_{l_i} \subseteq Pr_{l_j}$.*
$C = \langle C_L, C_\rightarrow\rangle$ *where $C_L = \{\omega_{L_1}\}$ and $C_\rightarrow = \{\omega_{\rightarrow_1}, \omega_{\rightarrow_2}\}$.*

Concerning the set of constraints that characterizes viable licenses and viable $\rightarrow$ relations:

$$\omega_{L_1}(l_i) = \begin{cases} False & \text{if } (P_{l_i} \cap O_{l_i}) \cup (P_{l_i} \cap Pr_{l_i}) \cup (O_{l_i} \cap Pr_{l_i}) \neq \emptyset; \\ True & \text{otherwise.} \end{cases}$$

That is, sets of permissions, obligations and prohibitions are mutually disjoint.

$$\omega_{\rightarrow_1}(l_i, l_j) = \begin{cases} False & \text{if cc:ShareAlike} \in O_{l_i} \text{ and } l_i \neq l_j; \\ True & \text{otherwise.} \end{cases}$$

That is, cc:ShareAlike term obligates the distribution of derivative works only under the same license.

$$\omega_{\rightarrow_2}(l_i, l_j) = \begin{cases} False & \text{if cc:DerivativeWorks} \in Pr_{l_i}; \\ True & \text{otherwise.} \end{cases}$$

That is, if cc:DerivativeWorks is prohibited in l_i then resources protected by l_i should not be protected by l_j.

Other constraints could be defined, but for the purposes of this classification we consider that these three constraints are enough.

$I = \langle l_l, A, \emptyset, \emptyset\rangle$ is the license compatible with all other licenses of the lattice. I is obtained by creating a license that permits all actions from A.

$S = \langle l_s, \emptyset, A, A \rangle$ is the license compliant with all other licenses of the lattice. S is obtained by combining all licenses with $\bigoplus$.

The size of the lattice for this classification (i.e., the number of licenses) is $2^{2|A|}$ where $2|A|$ is the size of the first level. The first level of the lattice is composed of the minimal set of licenses that allows $\bigoplus$ to generate all possible licenses, that is, all licenses having $|A| - 1$ actions in permissions and at most one action in obligations or prohibitions: $|P_{l_i}| = |A| - 1$ and $|O_{l_i}| + |Pr_{l_i}| = 1$. That gives $2\,|A| = 14$ licenses in our example. Thus, this classification is composed of 16384 licenses. The set of constraints identifies 2134 viable licenses.[15]

4.2 Analysis of the CC classification based on CaLi compared to the Creative Commons family

Our CC classification that uses the seven actions of Creative Commons is consistent with the one obtained from the web2rights tool. The number of licenses produced by CaLi is huge compared to the number of Creative Commons licenses but our goal is not classify their seven licenses. Our goal is to have a formal model that automatically classifies all possible licenses from a set of actions. To control the number of viable licenses, depending on the context, the set of constraints can be expanded. For instance, all compatibility rules identified in [13] can be included as C_L constraints.

Table 1 shows some licenses of the first level of this lattice.

Next levels of the lattice are produced by the combining operator. Table 2 shows some licenses of the Creative Commons family. Notice that CC BY is produced by $l_1 \bigoplus l_2$ and CC BY-NC by $l_1 \bigoplus l_2 \bigoplus l_3$.

Table 3 shows some licenses that are not part of the Creative Commons family. First license is like CC BY-NC but without the obligation to give credit to the copyright holder and/or author of the work. The second license prohibits to make multiple copies of the resource. The third license obligates the reuse of the resource by making exact copies of the original source. The last license is the third license with the prohibition to make a commercial use of the resource.

The combining operator may produce non-viable licenses or non-viable compatibility relations. Table 4 shows one non-viable license of the lattice that does not respect C_1.

Figure 3 shows a small part of the lattice of this CaLi classification, the one that concerns the compatibility relations of Creative Commons licenses. This graph shows only licenses and $\rightarrow$ relations that are viable. Thanks to C_2, the compatibility relation between CC BY-SA and CC BY-NC-SA is identified as non-viable and thanks to C_3, the compatibility relation between CC BY-ND and CC BY-NC-ND is identified as non-viable. We recall that the compatibility relations of this graph are conform to the ones obtained from the web2rights tool.

[15]This CaLi classification can be generated from https://github.com/benjimor/CaLi/tree/combinatory_method

5 A LICENSE-BASED LINKED DATA SEARCH ENGINE

We experiment the usefulness of a CaLi classification with the implementation of a license-based search engine for the Web of Data.

5.1 A CaLi-based classification for licenses of the Web of Data

This search engine is based on a CaLi classification that is similar to the CC classification of last section with the following particularities: (i) A contains the set of 72 actions considered by ODRL[16], (ii) licenses are not required to be consolidated, and (iii) a ω_{L_2} constraint is added to C_L.

Unlike Creative Commons actions, ODRL actions are organized using inclusion. For example, action *use* is included by *CommercialUse* or *play* is included by *display*. To preserve this dependency we include the constraint ω_{L_2} in C_L such that, if an action a_i is prohibited all the actions included in a_i must not be permitted nor obligated. For example, a license that prohibits *Use* is not viable if it permits or obligates *Commercial Use*.

$$\omega_{L_2}(l_i) = \begin{cases} False & \text{if } \exists a_i \in Pr_{l_i} \text{ and } \exists a_j \text{ included by } a_i \\ & \text{and } (a_j \in P_{l_i} \text{ or } a_j \in O_{l_i}); \\ True & \text{otherwise.} \end{cases}$$

The generation of a lattice has a complexity of $O(2^n)$. In our case, $n = 2|A|$ and the number of nodes in the lattice is 2^{144}. The number of nodes grows exponentially with the number of terms resulting in a combinatorial explosion. Generating the complete lattice is not suitable for a real scenario with a large number of terms. In next section, we show how we can generate part of the lattice *on the fly* to implement a license compliant tool.

5.2 Implementation of a license-based search engine

Since the complete lattice cannot be generated, we propose two algorithms to generate *on the fly* part of the lattice. This approach is in $O(n)$ where n is the number of existing nodes.

We chose to generate the most used licenses in datahub[17] and in the OpenDataSoft platform[18]. We extracted the sets of permissions, obligations and prohibitions from an existing set of RDF licenses[19]. As currently we do not take into account other aspects of a license like its jurisdiction, all licenses having the same sets will be considered in the same node.

The search engine maintains an index of licenses. The condition to add a license to our index is to have at least the URI of one RDF dataset associated to this license. As in this implementation the combining operator cannot combine all licenses to generate the complete classification (because the

[16]https://www.w3.org/TR/odrl-vocab/#actionConcepts
[17]https://old.datahub.io/
[18]https://data.opendatasoft.com/pages/home/
[19]http://purl.org/NET/rdflicense

Label	Permissions	Obligations	Prohibitions
l_1	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:CommercialUse, cc:Notice,	cc:Attribution	$\emptyset$
l_2	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:CommercialUse, cc:Attribution	cc:Notice	$\emptyset$
l_3	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:Notice, cc:Attribution	$\emptyset$	cc:CommercialUse
l_4	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:Notice, cc:Attribution	cc:CommercialUse	$\emptyset$

Table 1: Some licenses of the first level of the lattice.

Label	Permissions	Obligations	Prohibitions
$l_1 \bigoplus l_2$ (CC BY)	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:CommercialUse	cc:Notice, cc:Attribution	$\emptyset$
$l_1 \bigoplus l_2 \bigoplus l_3$ (CC BY NC)	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks	cc:Notice, cc:Attribution	cc:CommercialUse

Table 2: Some Creative Commons licenses of the lattice.

complete lattice cannot be generated), we make a verification of compatibility and compliance of the new license over (potentially all) existing licenses in the classification. The goal is to be conform to the partial order of the lattice. The challenge is to find the right place to insert new licenses.

First, the classification graph G is initialized with the infimum I, the supremum S and the compatibility relation $I \rightarrow S$. Then, in order to insert a new license in the classification, Algorithm 1 adds a new license l_i and calls Algorithm 2 to position l_i in the right place in the classification.

Algorithm 1 adds l_i to the classification G only if it is viable. License l_i is viable if it respects all constraints in C_L (in the particular case of the license-based search engine, C_1 and C_4). Then, it calls the recursive Algorithm 2 with I and G, to classify l_i in licenses compliant with I, i.e., the recursive link starts from the Infimum. Licenses compliant with I are all the licenses of G.

To be conform to the partial order of the lattice, Algorithm 2 uses the combining operator to check if two licenses are compatible. Using CaLi, license l_i is compatible with l_j iff $l_i \bigoplus l_j = l_j$. We recall that if l_i is compatible with l_j

Label	Permissions	Obligations	Prohibitions
l_3	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:Notice, cc:Attribution	$\emptyset$	cc:CommercialUse
$l_1 \bigoplus l_2 \bigoplus l_m$	cc:Distribution, cc:ShareAlike, cc:DerivativeWorks, cc:CommercialUse	cc:Notice, cc:Attribution	cc:Reproduction
$l_1 \bigoplus l_2 \bigoplus l_n \bigoplus l_o$	cc:Distribution, cc:ShareAlike, cc:CommercialUse	cc:Notice, cc:Attribution, cc:Reproduction	cc:DerivativeWorks
$l_1 \bigoplus l_2 \bigoplus l_3 \bigoplus l_n \bigoplus l_o$	cc:Distribution, cc:ShareAlike	cc:Notice, cc:Attribution, cc:Reproduction	cc:DerivativeWorks, cc:CommercialUse

Table 3: Some non Creative Commons licenses of the lattice.

Label	Permissions	Obligations	Prohibitions
$l_3 \bigoplus l_4$	cc:Distribution, cc:Reproduction, cc:ShareAlike, cc:DerivativeWorks, cc:Notice, cc:Attribution	cc:CommercialUse	cc:CommercialUse

Table 4: One non-viable license of the lattice.

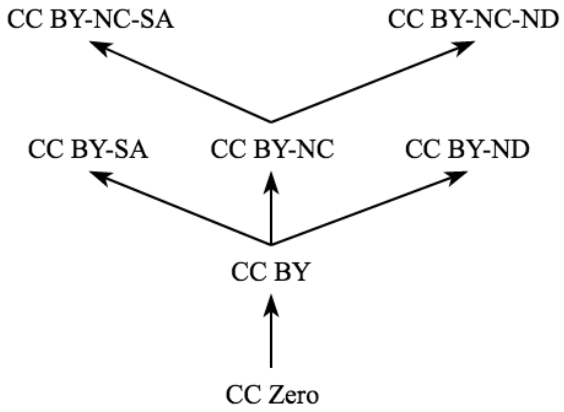

Figure 3: Compatibility of Creative Commons licenses taken from the CC classification.

then l_j is compliant with l_i. This algorithm recursively links l_i to the least restrictive licenses compliant with l_i in the classification.

Algorithm 2 tries to add compatible relations $\rightarrow$ between l_i and the least restrictive compatible licenses that are compliant with l_j (LRC_{l_j}). A compatible relation is added only if it is viable. Relation $\rightarrow$ is viable if it respects all $C_{\rightarrow}$ constraints (in the particular case of the license-based search

Algorithm 1: Adds and classifies a new license in the classification.

```
1  Function addAndClassify(l_i, G):
       Data: l_i: License, G: Classification Graph, I:
             Infimum,
       C_L: Constraints on licenses
       Result: Returns G with l_i classified
2      if viable(l_i, C_L) then
           // l_i respects all C_L constraints
3          Add l_i to G
           // Classifies l_i in licenses compliant with I
4          classify(l_i, I, G)
5      end
6      return G
7  end
```

engine, C_2 and C_3). Algorithm 2 checks if l_i is compatible with a least restrictive licenses l_{comp} compliant with l_j. If l_i is compatible with l_{comp} and l_i is compatible with l_j then l_i is inserted between l_j and l_{comp}. If l_i is only compatible with l_{comp}, l_i is linked to l_{comp}. Otherwise, Algorithm 2 is called again to find the least restrictive licenses compliant with l_i in the graph containing licenses compliant with l_{comp}.

Algorithm 2: Recursively classifies the new license to compliant licenses

```
1  Function classify(l_i, l_j, G):
       Data: l_i, l_j, l_comp: License, G: Classification Graph,
       C_→: Constraints on compatibility relations,
       LRC_{l_j}: Least restrictive licenses compliant with l_j in
       G
2      for l_comp ∈ LRC_{l_j} do
3          if l_i ⊕ l_comp = l_comp and viable(l_i → l_comp,
           C_→) then
               // l_i is compatible with l_comp
               // l_i → l_comp respects all C_→ constraints
4              add l_i → l_comp to G
5              if l_j ⊕ l_i = l_i and viable(l_j → l_i, C_→) then
                   // l_i is between l_j and l_comp
6                  add l_j → l_i to G
7                  delete l_j → l_comp from G
8              end
9          else if l_i ⊕ l_comp ≠ l_comp then
               // Classifies l_i in licenses compliant with l_comp
10             classify(l_i, l_comp, G)
11     end
12 end
```

With these algorithms, when licenses are inserted to G the partial order of the lattice is maintained. The produced graph is part of the lattice and is directed and acyclic.

Another strategy can be implemented to reduce the average number of operations needed to classify a new license. Algorithms 1 and 2 could start from the bottom (the least restrictive license I) or the top (the most restrictive license S) of G depending on the cardinality of permissions, obligations and prohibitions sets of the new license to insert.

Once the graph is enriched with licenses and associated datasets, answering our questions is straightforward. *Finding all licensed datasets whose licenses are compliant with a particular license* l_i means to retrieve datasets protected by l_i and all datasets protected by licenses that are above l_i in the graph. Similarly, *finding all licensed datasets whose licenses are compatible with a specific license* l_i means to retrieve datasets protected by l_i and all datasets protected by licenses that are below l_i in the graph.

The prototype of this search engine is available at http://cali.priloo.univ-nantes.fr[20]. The home screen[21] enables full-text and license compliant searches of RDF datasets. The CaLi classification is available through a documented API[22] containing functions that can answer our questions. For example, with /api/licenses/{licenseID}[23]/compatible, it is possible to retrieve licenses compatible with a particular license (e.g., CC BY-NC[24]). Likewise, with /api/licenses/{licenseID}/compliant it is possible to retrieve licenses compliant with a particular license (e.g., CC BY-NC[25]). A graph visualization of the classification is also available[26].

The source code is available at Github[27] under the MIT license. The classification is stored using Neo4j[28] because it facilitates the storage and exploration of graph structures.

6 CONCLUSIONS AND PERSPECTIVES

In this paper we proposed CaLi, a model to express license classifications. The goal is to encourage the publication and reuse of resources in a license compliant web.

CaLi can be used to classify licenses in different contexts where resources need to be reused. It considers a set of licenses, a combining operator, a compatibility relation over the set of licenses and a set of constraints. We consider licenses as sets of permissions, obligations and prohibitions. The combining operator applied automatically to the set of licenses produces a lattice that expresses both compliance and compatibility among licenses. This lattice guarantees a partial order over licenses where the relation *is compatible with* is reflexive, transitive and antisymmetric.

A limitation of our approach is the complexity in $O(2^n)$ of the lattice implementation. We demonstrate that the implementation of part of the lattice *on-the-fly*, in $O(n)$, is useful

[20] A video demonstration is available at https://youtu.be/YkSWHSiD-Ps.
[21] http://cali.priloo.univ-nantes.fr
[22] http://cali.priloo.univ-nantes.fr/api
[23] licenseID is the hash of permissions, obligations and prohibitions of a license.
[24] http://cali.priloo.univ-nantes.fr/api/licenses/3154600001362308832/compatible
[25] http://cali.priloo.univ-nantes.fr/api/licenses/3154600001362308832/compliant
[26] http://cali.priloo.univ-nantes.fr/graph
[27] https://github.com/benjimor/CaLi
[28] https://neo4j.com

for a real application like the license-based search engine we present.

A perspective of this work is to take into account other aspects of a license related to usage contexts like jurisdiction, dates of reuse, etc. Another perspective is to analyse how two CaLi classifications can be compared. That is, (1) if two classifications are defined with different sets of actions A_1 and A_2, (2) if there exist an aligment between A_1 and A_2, and (3) if the combining operators of both classifications are compatible then find a function (for instance) to pass from a lattice to another.

ACKNOWLEDGMENTS

Authors thank Matthieu Perrin for his helpful comments on this work.

REFERENCES

[1] Elena Cabrio, Alessio Palmero Aprosio, and Serena Villata. 2014. These Are Your Rights. In *European Semantic Web Conference (ESWC)*.

[2] Creative Commons. 2017. cc.licenserdf. https://github.com/creativecommons/cc.licenserdf.

[3] Enrico Daga, Mathieu d'Aquin, Enrico Motta, and Aldo Gangemi. 2015. A Bottom-up Approach for Licences Classification and Selection. In *International Semantic Web Conference (ISWC)*.

[4] Dorothy E Denning. 1976. A Lattice Model of Secure Information Flow. *Commun. ACM* 19, 5 (1976), 236–243.

[5] GR Gangadharan, Michael Weiss, Vincenzo D'Andrea, and Renato Iannella. 2007. Service License Composition and Compatibility Analysis. In *International Conference on Service-Oriented Computing (ICSOC)*. 257–269.

[6] Guido Governatori, Antonino Rotolo, Serena Villata, and Fabien Gandon. 2013. One License to Compose Them All. A Deontic Logic Approach to Data Licensing on the Web of Data. In *International Semantic Web Conference (ISWC)*.

[7] Georgia M Kapitsaki, Frederik Kramer, and Nikolaos D Tselikas. 2017. Automating the License Compatibility Process in Open Source Software With SPDX. *Journal of Systems and Software* 131 (2017), 386–401.

[8] Marco Mesiti, Paolo Perlasca, and Stefano Valtolina. 2013. On the Composition of Digital Licenses in Collaborative Environments. In *Conference on Database and Expert Systems Applications (DEXA)*.

[9] Víctor Rodríguez Doncel, A Gómez-Pérez, and Serena Villata. 2014. A Dataset of RDF Licenses. In *Legal Knowledge and Information Systems Conference (ICLKIS)*.

[10] Norman Sadeh, Alessandro Acquisti, Travis D Breaux, Lorrie Faith Cranor, Aleecia M McDonald, Joel Reidenberg, Noah A Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. 2014. Towards Usable Privacy Policies: Semi-Automatically Extracting Data Practices from Websites? Privacy Policies. *Symposium on Usable Privacy and Security (SOUPS)* (2014), 9–11.

[11] Ravi S. Sandhu. 1993. Lattice-Based Access Control Models. *Computer* 26, 11 (1993), 9–19.

[12] Valeria Soto-Mendoza, Patricia Serrano-Alvarado, Emmanuel Desmontils, and Jose Antonio Garcia-Macias. 2015. Policies Composition Based on Data Usage Context. In *Consuming Linked Data (COLD) in International Semantic Web Conference (ISWC)*.

[13] Serena Villata and Fabien Gandon. 2012. Licenses Compatibility and Composition in the Web of Data. In *Consuming Linked Data (COLD) in International Semantic Web Conference (ISWC)*, Vol. 905.