MODELLING THE COMPATIBILITY OF LICENSES

Benjamin Moreau,
Patricia Serrano-Alvarado,
Matthieu Perrin and
Emmanuel Desmontils

ESWC 2019 - June 2019 - Portorož
The Web facilitates the combination of resources: linked data, source code, Web services.

Resource producers use licenses to protect their resources.

Licenses specify the conditions of reuse of resources, i.e., what actions are permitted, obliged, prohibited, etc.
A license is a set of **status** assigned to **actions**.

**ODRL Information Model 2.2**
W3C Recommendation 15 February 2018

**Creative Commons**
Describing Copyright in RDF
Creative Commons Rights Expression Language
$l_3$ must be **compliant** with $l_1$ and $l_2$. 

**COMBINATION OF LICENSED RESOURCES**
$l_3$ must be **compliant** with $l_1$ and $l_2$.

$l_1$ and $l_2$ must be **compatible** with $l_3$. 
**COMBINATION OF LICENSED RESOURCES**

\[ l_3 \text{ must be compliant with } l_1 \text{ and } l_2. \]

\[ l_1 \text{ and } l_2 \text{ must be compatible with } l_3. \]

\[ r_1 \text{ and } r_2 \text{ are reusable with } r_3. \]
L_3 must be compliant with L_1 and L_2. L_1 and L_2 must be compatible with L_3. R_1 and R_2 are reusable with R_3.

In general, L_1 and L_2 are less restrictive than L_3.
Choosing an appropriate license for a combined resource is a difficult task.

The risk is either:
- to choose a license too restrictive
- to choose a license very permissive
Given a license, how to automatically position it over a set of licenses in terms of compatibility and compliance?

The challenge is to generalise the order relation while taking into account the semantics of actions.
RELATED WORK
## RELATED WORKS

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Check compatibility</th>
<th>Combine licenses</th>
<th>Order licenses in terms of compatibility</th>
<th>Order automatically</th>
</tr>
</thead>
<tbody>
<tr>
<td>License-based tools [1-4]</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>License combination [7-11]</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>License compatibility in FOSS [12]</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Lattice-based access control [13]</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>CaLi</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

[1] https://tldrlegal.com/
[2] https://creativecommons.org/choose/
CaLi, a lattice-based model that partially orders licenses in terms of compatibility using restrictiveness relations and constraints.

Demo of license-based search engine that orders most used licenses in the LOD.
A compatibility relation is always a restrictiveness relation

But how to decide if a license is less restrictive than another?

Is an action assigned to a status less restrictive than the same action assigned to another status?

Is a status less restrictive than another status?
A lattice of status $\mathcal{L}_S$ defines

1. all possible status $S$ for a license
2. the restrictiveness relations $\leq_S$ among status.
A lattice of status $\mathcal{S}$ defines

1. all possible status $S$ for a license
2. the restrictiveness relations $\leq_S$ among status.
A set of actions $\mathcal{A}$ is a set of actions that can be used to describe licenses. 
Example, $\mathcal{A} = \{\text{read, modify}\}$

A license is a function $l_i : \mathcal{A} \rightarrow S$
Example, $l_i(\text{read}) = \text{Permission}$
$l_i(\text{modify}) = \text{Prohibition}$

$L_{\mathcal{A},LS}$ is the set of all licenses
RESTRICTIVENESS LATTICE OF LICENSES

The set of all licenses $\mathcal{L}_{A,LS}$ partially ordered by restrictiveness.

\begin{align*}
  l_2(\text{read}) &= \text{Prohibition} \\
  l_2(\text{modify}) &= \text{Prohibition} \\
  l_1(\text{read}) &= \text{Permission} \\
  l_1(\text{modify}) &= \text{Permission}
\end{align*}
The set of all licenses $\mathcal{L}_{A,LS}$ partially ordered by restrictiveness.
License constraints identify non-valid licenses

\[ \omega_L : L \rightarrow \text{Boolean} \]

- \textit{True} : is valid,
- \textit{False} : is non-valid
IDENTIFY NON-VALID LICENSES

\[ \omega_{L1}(l_i) = \begin{cases} 
\text{False} & \text{if } l_i(\text{read}) = \text{Prohibition} \text{ and } l_i(\text{modify}) = \text{Permission}; \\
\text{True} & \text{otherwise.} 
\end{cases} \]
Compatibility constraints identify restrictiveness relations that are not compatibility relations.

\[ \omega : \mathcal{L}_{A,LS} \times \mathcal{L}_{A,LS} \rightarrow \text{Boolean} \]

- \textbf{True} : is a compatibility relation,
- \textbf{False} : is not a compatibility relation
IDENTIFY COMPATIBILITY CONSTRAINTS

\[ \omega_i \rightarrow_{I} (l_i, l_j) = \begin{cases} \text{False} & \text{if } l_i(\text{modify}) = \text{Prohibition}; \\ \text{True} & \text{otherwise.} \end{cases} \]
A CaLi ordering is a tuple $<\mathcal{A}, \mathcal{LS}, C_L, C_\rightarrow>$:

- $\mathcal{A}$ is a set of actions
- $\mathcal{LS}$ is a restrictiveness lattice of status
- $C_L$ is a set of license constraints
- $C_\rightarrow$ is a set of compatibility constraints
IMPLEMENTATION OF CALI ORDERINGS

- Size growth of cali orderings is $|\mathcal{L}_S| |\mathcal{A}|$
- Implemented algorithm
  - $n^2/2$ to generate a subgraph of a CaLi ordering ($O(n^2)$)
  - $n$ to insert a license ($O(n)$)

github.com/benjimor/CaLi
A LICENSE BASED SEARCH ENGINE

CaLi ordering size is $|\mathcal{LS}|^{|\mathcal{A}|} = 4^{72}$:

$\mathcal{A} = 72$ actions from ODRL

$\mathcal{LS} = \text{Undefined} \leq_s \text{Permission} \leq_s \text{Duty} \leq_s \text{Prohibition}$

$C_L = \{\omega_{L1}, \omega_{L2}, \omega_{L3}\}$

$C_\rightarrow = \{\omega_{\rightarrow 1}, \omega_{\rightarrow 2}\}$

What are the licensed resources that can be reused under a given license?

github.com/benjimor/CaLi-Search-Engine
CONCLUSION AND PERSPECTIVES
CONCLUSION

- CaLi, a lattice-based model
  - partially orders licenses: compatibility and compliance
  - implemented algorithm: orders a subset of licenses
  - use case: a license based search engine
- CaLi does not provide legal advice but aims at encouraging and facilitating the reuse of licensed resources.
● Extend CaLi to take into account
  ○ other aspects of licenses (jurisdiction, time validity, ...).
  ○ compatibility relations that are not based on restrictiveness relations.

● Define function to pass from a CaLi ordering to another.
MODELLING THE COMPATIBILITY OF LICENSES

THANK YOU!

ESWC 2019 - June 2019 - Portorož
RESTRICTIVENESS RELATION AMONG LICENSES

I1: cc-by-nd4.0 a odrl:Policy ;
   rdfs:label "CC BY";
   ... odrl:permission
       [odrl:action
           cc:Distribution,
           cc:Reproduction,
           cc:DerivativeWorks,
           cc:CommercialUse];
   odrl:duty
       [odrl:action
           cc:Notice,
           cc:Attribution];

Compliant

Restrictiveness

Compatible

I2: cc-by-sa4.0 a odrl:Policy ;
   rdfs:label "CC BY-SA";
   ... odrl:permission
       [odrl:action
           cc:Distribution,
           cc:Reproduction,
           cc:CommercialUse,
           cc:DerivativeWorks];
   odrl:duty
       [odrl:action
           cc:Notice,
           cc:Attribution,
           cc:ShareAlike];

I3: cc-by-nc-sa4.0 a odrl:Policy ;
   rdfs:label "CC BY-NC-SA";
   ... odrl:permission
       [odrl:action
           cc:Distribution,
           cc:Reproduction,
           cc:DerivativeWorks];
   odrl:duty
       [odrl:action
           cc:Notice,
           cc:Attribution,
           cc:ShareAlike];
   odrl:prohibition
       [odrl:action
           cc:CommercialUse].
LICENSE BASED TOOLS

● Help users to choose a license
  ○ DALICC, TL;DrLegal, CC Chose, Licentia, Choose a license

● Search of images with particular licenses
  ○ CC search

[1] https://tldrlegal.com/
[2] https://creativecommons.org/choose/
[5] https://ccsearch.creativecommons.org/
FCA-based license classification [11]

- Uses **Formal Concept Analysis** (FCA)
- But, cannot infer compatibility relation

FOSS license classification [12]

- Proposes a hand-made directed acyclic graph
- But, based on a manual interpretation

LATTICE-BASED ACCESS CONTROL

- Proposes a lattice model of secure information flow
- But, not used with licenses

Orders a set of n licenses with at most $n^2/2$ comparisons?

● A CaLi ordering of $3^7 = 2187$ licenses:

- $\mathcal{A} = 7$ actions from Creative Commons vocabulary
- $\mathcal{L}$ = Permission $\leq_s$ Duty $\leq_s$ Prohibition
- $C_L = C_\rightarrow = \emptyset$

● 20 subsets of licenses (from 100 up to to 2187 licenses)
● We measure the average number of comparisons needed to sort each subset (3 shuffles).
EXPERIMENTAL RESULT

![Graph showing the average number of comparisons for different sizes of graphs. The graph compares the actual performance of the Insertion sort algorithm with the upper bound of $n^2/2$.](image)