Software Licenses, Coverage, and Subsumption

Thomas A. Alspaugh and Walt Scacchi
Institute for Software Research
University of California, Irvine
{*alspaugh,wscacchi*}*@ics.uci.edu*

Rihoko (Inoue) Kawai
Saitama Institute of Technology, Saitama, Japan and
National Institute of Informatics, Tokyo, Japan
rihoko@nii.ac.jp

***Abstract*—Software licensing issues for a system design, instantiation, or configuration are often complex and difficult to evaluate, and mistakes can be costly. Automated assistance requires a formal representation of the significant features of the software licenses involved. We present results from an analysis directed toward a formal representation capable of covering an entire license. The key to such a representation is to identify the license's actions, and relate them to the actions for exclusive rights defined in law and to the actions defined in other licenses. Parameterizing each action by the object(s) acted on, the instrumental entities through which the action is performed, and similar contextual variables enables a subsumption relation among the actions. The resulting formalism is lightweight, flexible enough to support the scope of legal interpretations, and extensible to a wide range of software licenses. We discuss the application of our approach to the Lesser General Public License (LGPL) version 2.1.**

I. Introduction

Heterogeneously-licensed systems are increasingly prevalent as organizations seek lower development costs, increased reliability and quality, and faster development cycles [3], [6], [11]. Such systems presents challenges in ensuring all pertinent obligations from the various possibly-conflicting licenses are met, which can easily involve evaluating dozens of distinct licenses applied on a component-by-component basis [4], [12]. The challenges arise independently during design, development, integration, distribution, configuration, and execution, and may present different concerns involving different fundamental copyright and patent rights at each of these stages. The goal of our research is to provide licensing guidance to designers, developers, system integrators, and those responsible for software acquisition.

In our previous work we presented an approach and proof of concept of automated licensing analysis integrated into system design at the level of software architecture [3], [5], [6]. The work was based on a sequence of grounded-theory analyses on (eventually) 46 software licenses, focusing on propagation of obligations through the architectural configuration (the most challenging area for manual analysis). Rights and obligations were the fundamental units of the metamodel we obtained for licenses, with actions as components of rights and obligations and implicitly parameterized in a fixed pattern accommodating the license provisions that were the focus of the work. The textual analyses aimed for coverage of key provisions across a wide range of licenses.

The present work, in contrast, focuses on *complete coverage of all license provisions*, especially those that might not fit the earlier metamodel. License provisions are represented using a more flexibly extensible approach in which the fundamental unit is an action. Actions are parameterized as needed, recognizing that the subsumption relationship that can be inferred among actions is determined by the form in which the actions are parameterized. Rights and obligations then express relationships among desired, required, and forbidden actions. During our analysis we identified subsumption relationships among actions, linking each action involved in a license right back to the exclusive right subsuming it defined in copyright and other intellectual property law, specifically the U.S. Copyright Act and the Berne convention [16], [7]. Where possible, we also identified subsumptions of the actions of license obligations by the actions of rights. Figure 1 shows the subsumption relationships identified for a single license's actions.

In this paper we focus on the Lesser General Public License (LGPL), version 2.1 [10]. LGPLv2.1 is the seventh most widely used open-source software (OSS) license, accounting for about 6.5% of open source projects [8]. At 4341 words, it is substantial (almost double the mean length of licenses we have analyzed) yet small enough to be discussed manageably. It addresses the most challenging license interaction issue, propagation of obligations to components under other licenses, in a relatively straightforward way compared to other licenses that do so. It has provisions in many of the categories that are challenging for analysis, including:

- accumulation of copyright notices,
- alternative obligations,
- clauses with null effect,
- definitional clauses,
- the distinction between collective and derivative work,
- distribution under alternative licenses,
- distinct rights and obligations for build scripts, interfaces, header files, source, object, and executable forms,
- license acceptance and termination,
- license exceptions,
- license notices of several types,
- output from licensed software, and
- relicensing under other licenses.

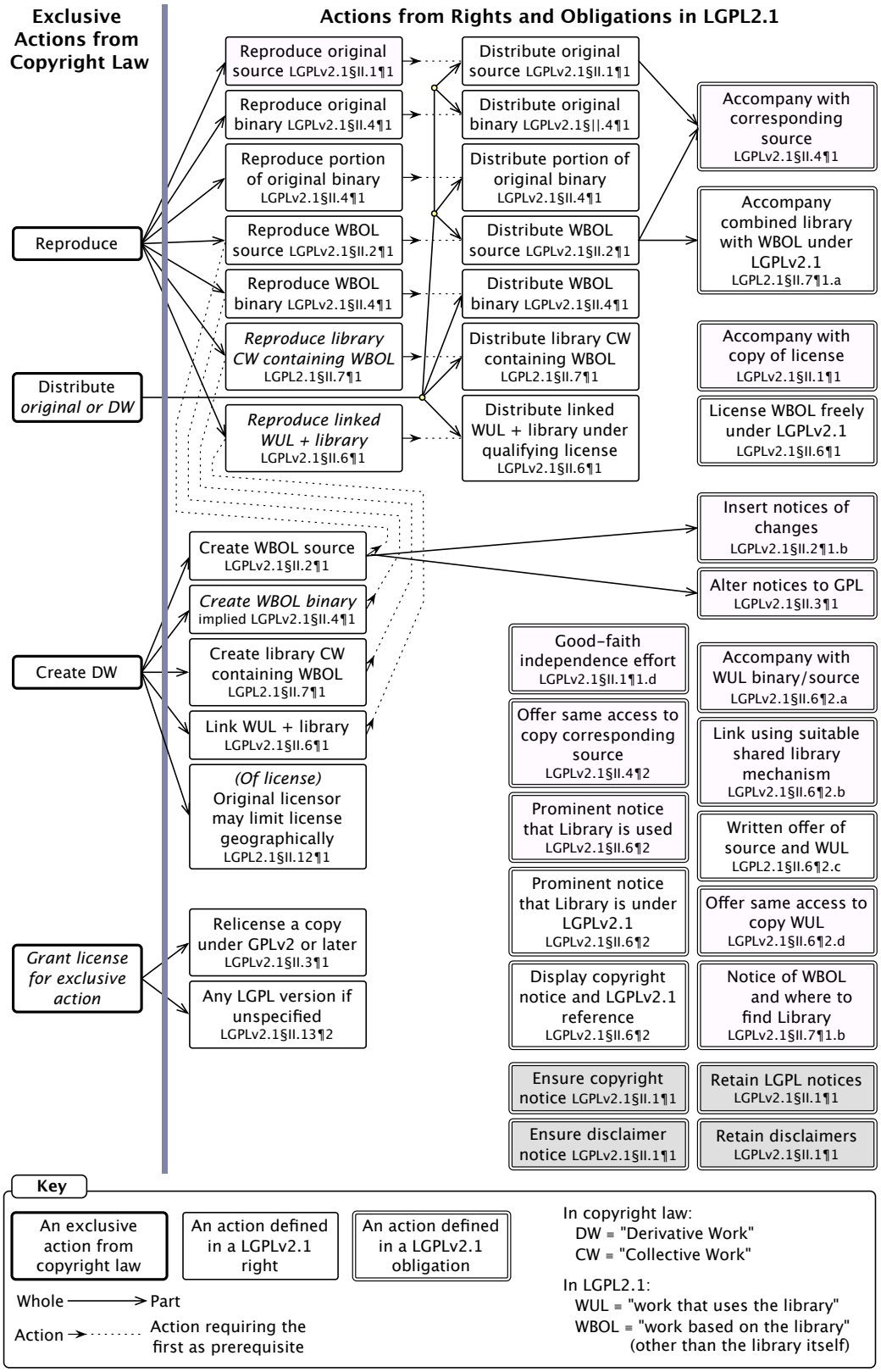

Figure 1. Subsumption among the LGPLv2.1 actions for rights and obligations and the exclusive copyright actions. 18 rights actions are explicit in the text, and three others are implied. The actions of the implied rights are italicized in the figure. Four obligations actions have no effect under the conditions in which they are obligated (because the original source must itself satisfy LGPLv2.1); they are shown with a gray background.

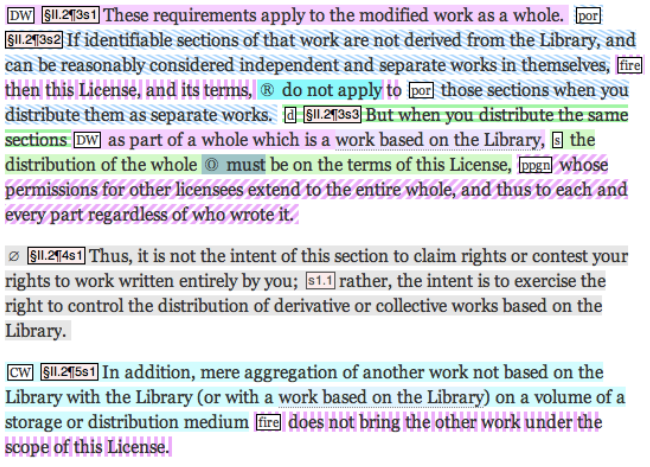

Figure 2. A portion of LGPLv2.1, divided into chunks and annotated with categories of interest. The categories appearing here are, briefly: CW: collective work; d: distribution; DW: derivative work; fire: license firewall; O: apparent obligation; por: for a portion of the licensed entity; ppgn: propagation of obligations to other entities; s: sublicensing, whether explicit or in effect; and ∅: null effect.

Figure 2 shows an excerpt of the open-coded LGPLv2.1 text annotated with some of the 93 categories that were identified here or in other licenses [9]. The entire license text was chunked and open-coded, reiterating until the boundaries of chunks of text and the conceptual code characterizing each chunk of text stabilized. The list of codes (or categories) was initialized with the codes obtained from our previous analyses of many licenses, and extended to include the kinds of features uncovered by a focused analysis of the LGPLv2.1 text. Portions of the chunking and open-coding were verified by one of the other authors at several points in the process. Axial coding was then used to identify themes and relationships in the license text, resulting for example in the categories of definitions, rights, obligations, modifiers, and null effect discussed in Section IV, and the characterization of a parameterized action as the basic unit of software licenses discussed in Section V.

LGPLv2.1, like most licenses, is only partially organized into numbered sections, hampering reference to specific parts of the text. Citations of specific license sections, paragraphs, and sentences refer to an online copy of LGPLv2.1 consistently numbered throughout by a program [2].

The remainder of this paper is organized as follows. Section II presents related work. Section III lists questions of interest that guide this work. Section IV presents the textual analysis on which the work is based. Section V discusses actions, Section VI sketches how actions are parameterized, and Section VII outlines subsumption among parameterized actions. Section VIII discusses some issues arising from the work, and Section IX concludes the paper.

II. Related Work

Hohfeld sought a theory by which to resolve the imprecise terminology and ambiguous classifications he found in use for legal relationships. In a seminal article published in 1913 and cited to the present day, he set forth a system of eight jural relations intended to express and classify all legal relationships between people. The first four regulate ordinary actions and are *right* ("may"), *no-right* ("cannot"), *duty* ("must"), and *privilege* ("need not").

There has been much work on analysis of laws in AI over the past few decades. A widely-cited example is Sergot et al.'s re-expression of the British Nationality Act as a Prolog program; the resulting program was able to apply the Act to a particular person's situation and characteristics to determine nationality [15]. Sergot asserted that the primary value of their approach was the insight that the process of expressing a statute gave into what the statute says and means, rather than any use of the Prolog program.

Otto and Antón survey the literature on modelling legal texts and reach similar conclusions [14]. They highlight the possibilities of conflicts among regulations, the evolution of case law and passage of new regulations, and the frequent cross-references within a single text or from one text to other texts. They survey a number of modelling approaches, including symbolic logic, knowledge representation (including Sergot et al.), deontic logic, defeasible logic, temporal logic, and so forth.

None of these approaches appear well suited to the challenges of licenses. The problem of references among documents, prominent in the modelling of statutes and regulations, is not significant for the licenses we have examined, which make few references to other documents and exhibit comparatively straightforward references within the license. While the modelling approaches offer a certain degree of automatic calculation, the calculations they support well do not appear particularly useful for OSS licenses. The key issue we have found for OSS licenses, namely how license provisions refer to specific entities in the licensed system and how obligations resulting from rights for one entity are propagated to other entities based on the architectural structure connecting them, is unlike anything addressed by these approaches. While they may possibly offer an efficient run-time implementation for the calculations needed for licenses, it is not clear that they are particularly appropriate for modelling licenses.

III. Questions of Interest

Our work is in the context of software development and the issues and concerns that arise there. The primary goal in that context is to produce a software system for which the desired rights are available in exchange for acceptable obligations. Since the work originated in the context of the OSS community, we assume good will on the part of the actors involved, and do not concern our work with

approaches for subverting license provisions. We list the following questions of interest that guide our research.

1) What rights are potentially available for a single component under a given license, or for a given architectural configuration of components and connectors under their licenses?
2) What obligations must be fulfilled in order for specific rights to be granted for a given single component under its license, or for a given architectural configuration of components and connectors under their licenses?
3) What license conflicts (if any) arise for a given architectural configuration of components and connectors under their licenses?

We find that in addressing these questions we need not consider any license provision that is neither *enactable* nor *testable*. A surprising number of license provisions fall into these categories, including text that we characterize as exhortations, examples or informal explanations of other clauses, and hopes on the part of the licensor that have no legal force. Our focus on testable provisions also leads in the intriguing direction of automated verification of whether a testable license obligation has been fulfilled.

IV. Textual Analysis

We find that everything in the text of LGPLv2.1 may be classified as either

1) the *definition* of a term,
2) a *right*,
3) an *obligation*,
4) a *modifier* to a definition, right, or obligation, or
5) *text without legal effect*.

These five categories cover the entire text and partition everything in it. Examples of each from LGPLv2.1 are given below for readers unfamiliar with OSS licenses.

A. Definitions of terms

The first example is an explicit definition of a named term, "work that uses the Library".

> A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". (§II.0¶2)

The second example is an implicit definition of an anonymous category of executables that might be termed "work using the Library and linked with it". Executables in this category have rights and obligations different from those for other executables. LGPLv2.1 gives this category no name.

> ... you may also combine or link a work that uses the Library with the Library to produce a work containing portions of the Library .... (§II.6¶1s1)

B. Rights

The first example, as is common for statements of rights in OSS licenses, grants several rights at once (the right to copy and the right to distribute). The actions in this right might be summarized as "reproduce complete original" and "distribute complete original". We use such summaries here as tokens representing the full definitions.

> You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium ... (§II.1¶1s1)

The second example grants an interesting right to license a specific copy of a work received under LGPLv2.1 under another license. In both these examples, the word "may" signals that a right is probably being defined. The action might be summarized as "license a given copy under GPL".

> You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. (§II.3¶1s1)

C. Obligations

The first example is signaled by the word "provided", unlike most LGPLv2.1 obligations which are signaled by "must". This obligation is notable because it would seem to require no action unless the original source code, in violation of LGPLv2.1, failed to include such a notice and disclaimer of warranty. The actions might be summarized as "ensure appropriate copyright notice" and "ensure disclaimer of warranty".

> ... provided that you conspicuously and appropriately publish on each copy [of the complete original source code] an appropriate copyright notice and disclaimer of warranty ... (§II.1¶1s1)

The second example contains no such identifying words, but is the first of a list of alternatives preceded by "... you must do one of these things". Its action might be summarized as "accompany with corresponding source". Many OSS licenses contain similar obligations.

> Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work .... (§II.6¶2.as1)

D. Modifiers

This first example contains the signal word "provided" that often indicates an obligation, but it does not function as such. Instead its effect is to restrict what "terms of your choice" refers to.

> ... you may also combine or link a work that uses the Library with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for

> the customer's own use and reverse engineering for debugging such modifications. (§II.6¶1s1)

The second example limits the scope of the anonymous category of "works that use the Library" that are also "works based on the Library" because they incorporate material from header files.

> If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted .... (§II.5¶4s1)

E. Text without Legal Effect

The first example below is an explanation and statement of the intent of the license's authors; we are told that if the explanation differs from what it purports to explain, their stated intent would be trumped by what the license actually says.

> Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you .... (§II.2¶4)

The second example is more problematic. It is phrased as an obligation, but the action involved ("make a good faith effort") is in our view not testable; compare for example the undoubtedly testable action "conspicuously and appropriately publish on each copy an appropriate copyright notice" (§II.1¶1s1). Of course, a specific legal interpretation of LGPLv2.1 might give this text a testable interpretation, for example by operationalizing "good faith effort" in some way.

> If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (§II.2¶1.d)

F. Other features

In addition to the five categories of definitions, rights, obligations, modifiers, and null effect that jointly partition and cover the entire LGPLv2.1 text, we identified or confirmed several other significant license features.

1) Right/Obligation Structure: All rights and obligations shared the conceptual structure of an actor, a Hohfeld jural relation [13], and an action. The actor was the licensee for each of LGPLv2.1's 18 rights and 20 obligations. The jural relation was that of a Hohfeld *right* ("may") or *privilege* ("need-not") for each right, and of a *duty* ("must") or *no-right* ("cannot") for each obligation. Actions will be discussed in Section V.

2) Time and State: Time and state are barely present in LGPLv2.1, figuring only in license acceptance (§II.9) and termination (§II.8). There is no provision for reinstatement after termination.

3) Obligation Propagation: Propagation of obligations to other entities is mediated structurally by the architecture in which LGPLv2.1-licensed entities are combined:

1) to other elements incorporated into the same library (§II.2¶1.c);
2) to programs designed to use an LGPLv2.1-licensed library, when linked to the library (§II.5¶2), except if certain obligations are met (§II.6¶1); and
3) to the object code for modules that include more than a stated amount from an LGPLv2.1-licensed header file (§II.5¶3).

4) Enactability and Testability: The constructs that appear intended as LGPLv2.1 rights or obligations all involve actions that are clearly testable, with the single exception of the "good faith effort" obligation discussed above. Every action (even the questionable one) is, unsurprisingly, enactable.

V. Actions, the Central Construct

Actions are the most common constructs in LGPLv2.1, and are essential in how the license is applied in the world. Focusing on actions as the key element of licenses brings several advantages.

- Actions are more manageable than rights and obligations. Each action is a concept representing an unbounded set of instances of the action; e.g. "distribute the Library ... in object code ... form" (§II.4¶1) is instantiated by "distribute `glibc` to John Doe on 2012 June 18" along with any number of similar instances. Therefore set operations may be used on actions. The operations on rights and obligations, in contrast, are quite limited. For example, the common idiom of first stating an obligation to do action X, then reducing it by granting the right to not do W where W overlaps with or is part of X, is easily expressed as set subtraction on the actions ($X - W$) but has no simple expression in terms of the obligation and right themselves.
- A single action, or two actions related by subsumption, often appear in both a right and an obligation. In LGPLv2.1 examples are numerous, for example the obligation "You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change" (§II.2¶1.b) whose action is subsumed by that of the right "You may modify your copy or copies of the Library" (§II.2¶1). This phenomenon is essential to the propagation of obligations from one license to entities under another license, which doesn't work unless the other license permits the actions required by the propagated obligations.
- Distinguishing an actual right or obligation from a modifier in the form of a right or obligation can be

problematic, as observed in Section IV, but in our analysis we found identifying actions to be uniformly straightforward.

- If rights and obligations are the primary constructs, then their similarities (both comprise an actor, a Hohfeld jural relation, and an action) and unwieldy difference (though each contains a Hohfeld relation, it can't be the same one) are prominent and difficult to justify. But if actions are the primary construct, then rights and obligations become emergent phenomena arising from the relationships among a license's desired, required, and forbidden actions, and the description of the license metamodel becomes simpler and more uniform.

VI. Action Parameterization

In our previous work we proposed that actions be parameterized with the entity on which they acted, if any, and the license used in the action, if any. However, a more careful examination of license rights and obligations, showed that this simple pattern, while sufficient for the majority of current rights and obligations, is neither universally sufficient nor conceptually necessary. Our current work has shown that no fixed pattern or patterns is necessary for formalization and automated inference, and that a small but stubborn set of actions cannot be accommodated in that way. LGPLv2.1 offers two examples, of which the following (from a right) is the clearest.

In the action "distribute that work under terms of your choice" (§II.6¶1), the work in question is a "work that uses the Library" combined or linked with the Library, and the terms in question must meet two conditions (licensee may modify the work, and may reverse-engineer the work). This action thus involve two entities:

1) "that work", upon which the action is taken, and
2) the "terms of your choice" through which the distribution is licensed.

Each of these should be a separate parameter of the action, since they vary from instance to instance of the action and may vary independently of each other. If the action is parameterized in this way, then it becomes a special case of the general action "distribute an entity under a license" and its parameters place it properly in the subsumption hierarchy, as discussed in Section VII.

(We note in passing that the approach in our previous work for parameterizing the actions of obligations through functions or quantifiers operating on the parameter(s) of the corresponding right continues to suffice, based on the results reported here; we refer interested readers to that work [3], [5], [6].)

VII. Parameterized Subsumption

In addressing subsumption among parameterized actions, we follow the approach of Abadi and Cardelli in the area of object-oriented type systems [1]. Figure 3 illustrates subsumption between pairs of simple actions and pairs of parameterized actions.

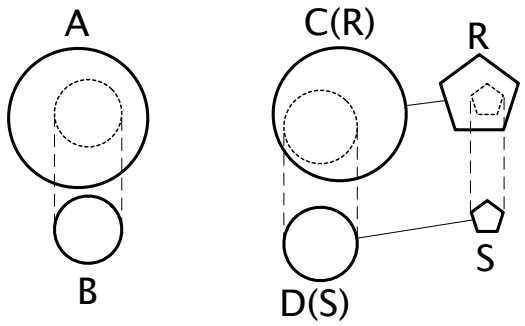

Figure 3. Subsumption of simple entities and parameterized entities

In the figure, every instance of action B is also an instance of action A; we say A subsumes B.

On the right is a more complex situation. Actions $C(P)$ and $D(P)$ are parameterized with arguments R and S respectively. As is normally the case for parameterized actions in licenses, the parameter is *covariant*: the sense of the subsumption of the arguments matches their effect on the subsumption of the actions they parameterize. Every instance of $D(S)$ is an instance of $C(R)$ if every instance of S is an instance of R; argument S is subsumed by argument R so therefore $D(S)$ is subsumed by $C(R)$.

An example from LGPLv2.1 is the right "You may modify your copy or copies of the Library" (§II.2¶1) and the obligation "You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change" (§II.2¶1.b) In other licenses we have seen actions to modify licensed entities other than libraries, and to insert various kinds of notices appropriate for the license in question, so we propose generalizing these actions to covariantly parameterized actions informally defined as

$$\begin{aligned} M(F,L) &= \text{"modify source file F licensed under L"} \\ N(F,L) &= \text{"add change notices appropriate for license L to source file F licensed under L"} \end{aligned}$$

Let us assert that M subsumes N covariantly

$$N(g,m) \subseteq M(f,l) \text{ if } g \subseteq f \text{ and } m \subseteq l$$

and also assert that

> "modify your copy or copies of the Library" (§II.2¶1)

is equivalent to the union of $M(F, \text{LGPLv2.1})$ for each Library file F you modify, and that

> "cause the files modified to carry prominent notices stating that you changed the files and the date of any change" (§II.2¶1.b)

is equivalent to the union of $N(F, \text{LGPLv2.1})$ for each Library file F you modified. Then we have taken several

steps towards being able to automatically determine that modifying `libfile.c` under LGPLv2.1 subsumes adding LGPLv2.1 changes notices to `libfile.c`. Our assertions have expressed part of the interpretation of the two actions, and constituted a step in the formalization of an interpretation of LGPLv2.1 as a whole.

We have made such formalizations of portions of licenses in our previous work, and believe the present work provides a foundation for doing the same for entire licenses.

VIII. Discussion

In this section we discuss some of the issues arising from this approach.

A. What is and is not formalized?

These are formalized:

- The subsumption relation among actions, including the effect of the actions' argument.
- The entity types over which each parameter ranges, and the subsumption relation among the types.
- The entailment relation from obligations to the rights granted in exchange from them.

This is not formalized:

- The interpretation of each action.

This formalization is sufficient to support automated licensing calculations to support designers, developers, integrators, and acquisition analysts.

The task of creating the formalization is not small, but the resulting automation saves substantial work with every licensing analysis.

B. Testable leaves

We hypothesize that software oracles can be constructed for many or even nearly all actions in license obligations. Examples are:

- an oracle to check for specific warranty disclaimers in source code
- an oracle to check whether a specific source file is available at a specific URL

And so forth. In most cases it will be impractical or impossible to automatically check for *any* condition that would fulfill the obligation (say, for any copyright notice that would serve for LGPLv2.1), but it will be practical to check for a *specific* condition known to fulfill it (say, for one specific copyright notice that serves).

C. How subsumption fits into license analysis

In our previous work we explored license relationships among rights and obligations using Hohfeld jural relations, and described how we automated licensing analyses for specific systems and incorporated the analyses into a software architecture development environment [3], [5], [6]. The work presented here provides a new and more extensible foundation for those analyses.

For example, LGPLv2.1 states

> You may modify your copy ... of the Library ... and ... distribute such modifications ... provided that you ... cause the files modified to carry prominent notices stating that you changed the files and the date of any change. (§II.2¶1)

A specified subsumption relation among actions might classify "Richard Roe distributed `glibc` v1.2.3 source on 2012 July 26" as an instance of the rights action "Distribute WBOL source" (§II.1¶1) and "Jane Doe placed LGPLv2.1 change notices in `glibc` v1.2.3 on 2012 July 25" as an instance of the obligations action "Insert notices of changes"(§II.1¶1), which is itself subsumed by rights action "Create WBOL source"(§II.1¶1) (Figure 1). It still remains to relate the fulfilled obligation of placing the notices to the desired right of distributing the modified source (as well as the other obligations imposed by LGPLv2.1), and presumably to decide whether Jane Doe and Richard Roe were acting jointly so that her fulfilled obligation supported his exercised right.

D. Legal interpretations

We believe the interpretation provided by the formalization provides sufficient scope for legal interpretations, based on informal conversations with lawyers and researchers in law, but the choices provided by the formalization do not appear to be a natural expression of the choices a lawyer would make. This will require future work.

E. Questions we need not ask

For the goals we have set for this research, it is not necessary to be able to answer certain research questions. Examples we have identified are:

- *Given two software licenses, how are they related?*
  We believe the answer for any pair of existing licenses is that they are not equivalent and neither one subsumes the other. Our research indicates that useful comparisons are only possible for individual rights or obligations, or at most for groups of a few.
- *Can our approach account for delegation?*
  We are not aware of any licenses with provisions that explicitly address delegation of obligations. Licensing is of course a delegation of one or more rights. Our approach does not analyze this kind of delegation beyond a surface level.
- *Would it be advantageous to apply a temporal, deontic, or other specific logic to licensing analysis?*
  Perhaps; such logics might enable the asking of different kinds of questions that we cannot address at present. Our combination of a logic based on Hohfeld and description logic has sufficed so for our stated goals. We continue to look for contexts in which additional kinds of reasoning would be beneficial.

F. Future work

We hypothesize that license-based reasoning about software systems offers benefits in domains beyond that of intellectual property licenses such as LGPL. Our ongoing research program is examining the use of license-like structures for security, envisioning "security licenses" to fulfill the goals of security policies and similar measures but more manageable and scalable manner, with direct application to software engineering processes such as open-architecture OSS development of systems integrated from components from many sources. Data licensing is another promising area, especially since data licenses already exist.

IX. Conclusion

We present initial results from an analysis of LGPLv2.1 in its entirety, based on earlier work that analyzed high-value areas of a collection eventually numbering 46 licenses. The analysis covers the license textually in several senses:

1) as a grounded-theory analysis chunking and open-coding the entire text;
2) as a higher-level synthesis by which the license text was partitioned a second time (into definitions, rights, obligations, modifiers, and no-effect); and
3) as all LGPLv2.1 actions and the relations among them from which arises the structure of rights and obligations for the license.

The analysis also identified *actions* as the central concept around which license structure is organized. When actions are taken as the fundamental construct, the characteristics of rights and obligations become emergent phenomena arising from the relationships among a license's desired, required, and forbidden actions. The focus on actions also led us to a more flexible and generalized approach for parameterizing actions and deriving a subsumption relation among them. We extended the subsumption relation to include the actions for the relevant exclusive copyright rights (Figure 1), and to relate the actions for rights and obligations. Grounding the relation in the actions of the exclusive rights proved helpful in distinguishing actual rights and obligations from provisions in the textual guise of rights or obligations but serving the function of modifiers of definitions, rights, and obligations. While no analysis or interpretation of a license can be considered final, the three kinds of coverage achieved and cross-correlated (of the text at both an open-coding and an axial coding level, and of the license's actions supported by a grounding in the copyright exclusive actions) give confidence in the results.

Acknowledgements

This research is supported by grant #N00244-12-1-0004 from the Acquisition Research Program at the Naval Postgraduate School, and by grant #0808783 from the U.S. National Science Foundation. No review, approval, or endorsement implied.

The authors thank the anonymous reviewers whose insightful suggestions helped us improve the paper.

References

[1] M. Abadi and L. Cardelli. *A theory of objects*. Springer-Verlag, New York, 1996.

[2] T. A. Alspaugh. GNU Lesser General Public License, version 2.1, §/¶/sentence-numbered. http://www.thomasalspaugh.org/pub/osl-sps/lgpl2.1.html.

[3] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi. Intellectual property rights requirements for heterogeneously-licensed systems. In *17th IEEE International Requirements Engineering Conference (RE'09)*, pages 24–33, 2009.

[4] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi. The role of software licenses in open architecture ecosystems. In *First Int. Workshop on Software Ecosystems*, pages 4–18, 2009.

[5] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi. Presenting software license conflicts through argumentation. In *23rd Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 2011)*, pages 509–514, 2011.

[6] T. A. Alspaugh, W. Scacchi, and H. U. Asuncion. Software licenses in context: The challenge of heterogeneously-licensed systems. *Journal of the Association for Information Systems*, 11(11):730–755, 2010.

[7] Berne Convention for the Protection of Literary and Artistic Works, 1979. http://www.wipo.int/treaties/en/ip/berne/.

[8] Black Duck Software. Top 20 most commonly used licenses in open source projects. http://www.blackducksoftware.com/oss/licenses.

[9] J. M. Corbin and A. C. Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 2007.

[10] Free Software Foundation. GNU Lesser General Public License, version 2.1, 1999. http://www.gnu.org/licenses/lgpl-2.1.html.

[11] D. M. German and A. E. Hassan. License integration patterns: Addressing license mismatches in component-based development. In *28th International Conference on Software Engineering (ICSE '09)*, pages 188–198, 2009.

[12] R. Gobeille. The FOSSology project. In *Int. Working Conf. on Mining Software Repositories (MSR'08)*, pages 47–50, 2008.

[13] W. N. Hohfeld. Some fundamental legal conceptions as applied in judicial reasoning. *Yale Law J.*, 23(1):16–59, 1913.

[14] P. N. Otto and A. I. Antón. Addressing legal requirements in requirements engineering. In *15th Int. Requirements Engineering Conference (RE'07)*, pages 5–14, 2007.

[15] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The British Nationality Act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.

[16] U.S. Copyright Act, 17 U.S.C. http://www.copyright.gov/title17/.