A Feature Interaction View of License Conflicts

G.R. Gangadharan[1], Michael Weiss[2], Babak Esfandiari[2], and Vincenzo D'Andrea[1]

[1] Department of Information and Communication Technology, University of Trento, Via Sommarive, 14, Trento, 38050 Italy
{gr,dandrea}@dit.unitn.it
[2] School of Computer Science, Carleton university, 1125 Colonel By Drive, Ottawa, K1S 5B6, Canada
{weiss,babak}@sce.carleton.ca

Abstract. In this paper, we introduce the problem of license conflicts, which occurs when information assets (such as software, data, or multimedia files) are composed, derived or versioned. A license specifies a set of permissions granted by an asset owner to an asset consumer (as expressed in the form of licensing clauses), effectively waiving what would otherwise be an infringement of the owner's intellectual rights. Thus, a license allows producers to control how consumers may use and extend the asset. New assets can be produced by composing multiple assets or deriving an asset from an existing asset, as governed by their licenses. Licenses interact with each other either directly or indirectly during the composition or derivation of assets. Licenses can also interact with other versions of the same license during the evolution of an asset. We view interactions of licenses as feature interactions, especially if those interactions result in conflicts. Here, features correspond to licensing clauses. In this paper, we identify and analyze feature interactions of licenses during the composition, derivation, and evolution of assets.

1 Introduction

Information assets (referred to simply as *assets* in this paper) are described as information that is of value to an organization. An asset can be software, a component, service, process or content that holds intellectual value. It can be combined with other assets. A new asset can also be derived from an existing asset. New versions of an asset can, furthermore, be released as a representation of enhancements to its functional or non-functional specification. However a new asset is produced, its distribution involves a license that must represent the unified view of the licenses of the composed assets, or the parent asset.

During the formation of new asset, the licensing clauses of one asset may conflict with the licensing clauses of other assets. These conflicts are similar to the conflicts observed in feature interactions. Hence, licensing clauses are modeled as features in this paper. Understanding these interactions, and developing techniques for their detection and resolution will be critical for the legally authorized use of assets on any significant scale. Furthermore, the detection of these

feature interactions will be the cornerstone for developing a framework for the semi-automatic composition of licenses. In this paper, we take first steps towards such a framework by providing a conceptualization of license conflicts as feature interactions, and a classification of license feature interactions.

This paper introduces license conflicts as feature interactions, which may occur when information assets are composed, derived or versioned. It is organized as follows. Section 2 introduces the fundamentals of asset licensing and licensing clauses. Section 3 frames the problem of feature interactions for licenses. We classify licensing conflicts that can arise from these interactions in Section 4. In Section 5, we illustrate the various scenarios of feature interactions in the context of licenses specific to services, music, and software assets. Section 6 discusses related work in this field, followed by our conclusions in Section 7.

2 Basics of Asset Licensing

The distribution of an asset is always accompanied by a *license*, which describes the terms and conditions imposed by its producer. A license reflects the overall business value of the asset to its producers and consumers. Licensing is often used to protect the intellectual rights of asset producers, thereby turning the assets into a source of revenue, and licenses into a tool for business strategy. Also, licenses give developers control over how consumers can use the licensed assets. Consequently, asset producers rely on licenses to protect their assets from unauthorized consumption.

An asset producer (the *licensor*) never transfers ownership of the asset to the consumer. Instead, the consumer (the *licensee*) merely obtains the right to use/extend the asset subject to the restrictions imposed by the license [1]. Thus, asset licensing is considered to include all transactions between a licensor and a licensee, in which the licensor agrees to grant the licensee the right to use and/or extend (by deriving from it) the asset under predefined terms.

More broadly, an asset license is expected to have these elements [2]:

1. ***Subject of the License:*** The subject of the license relates to the definition of the asset being licensed, such as a unique identification code for the asset, a name for the asset, and other additional information.
2. ***Scope of Rights:*** The scope of rights reflects on what the licensee can do with the licensed asset [3]. This defines the extent to which the asset can be used, accessed, and value added to it (composition or derivation). Several different grants of rights are described including the right to reproduce, display, access, modify, make derivative works, sell or distribute, import, and sub-license to another party, who can do any of the above. The Scope of Rights falls into four types: Usage, Reuse, Manage, and Transfer.
   - **Usage:** Usage pertains to the end use of the asset. Usage rights are generally rendering actions like execute, play, display or print.
   - **Reuse:** Set of rights pertaining to the reuse of an asset by modifying, excerpting, or aggregating. Reuse can be in full or in part.

- **Manage:** Rights pertaining to the digital management of an asset. This includes housekeeping actions such as back up, install, or uninstall.
- **Transfer:** Transfer rights apply to the actions that allow a person or agent to transfer some specific rights to another person or agent. In general, transfer rights include the right to sell, lend, or lease. Transfer rights may involve ownership transfer, and may allow the asset to be used in perpetuity with or without exchange of value.

3. ***Financial Terms:*** Describe how the licensee will pay for the use of an asset. Consumers make payments either through royalties or a lump sum payment. Generally royalties are based on per unit sales. Lump sum payments are an alternative to royalties. Sometimes, lump sum payments are also used in addition to royalties. A lump sum payment can be paid by the consumer in advance of using the service (prepaid) or at the later stage (post-paid). Alternatively, the producer can make the asset available free of charge.
4. ***Warranties, Indemnification, and Limitations:*** Address issues of who bears the financial risk of asset defects or the legal risk of a third party claiming that the asset infringes on or violates their intellectual rights.
   - **Warranties:** A warranty is a promise regarding the description of the assets and their quality, stated by the producer.
   - **Indemnification:** Provision of defense by the licensor for the licensee if a third party sues the licensee, alleging that the licensee's use of the licensed asset infringes on or violates their intellectual rights [4].
   - **Limitation of liability:** Limitation of liability deals with the liability of each of the parties under the license agreement.
5. ***Evolution:*** Pertains to the rights over future releases or versions of an asset.

Furthermore, there are licensing clauses that provide moral support (these are also known as moral rights) to consumers and providers.

Attribution: An asset may expect attribution for its use in any form by another asset. Thus, attribution is ascribing an asset to its creator. ***Non-Commercial Use:*** An asset can allow or deny other assets to use it either for non-commercial purposes or for commercial purposes.

Sharealike: An asset may expect another asset to reflect the same terms and conditions (similar to Copyleft of GNU[3] or Sharealike of Creative Commons [5]).

As rights for assets vary based on the nature and context of the assets involved, the expression of rights for a particular asset will be more specific. For example, one of the rights for a multimedia asset can be to *play* it. The concept of playing cannot be directly applied as a right to web service assets. Similarly, the rights for a web service differentiate between the levels of interface and implementation, which are not separate for multimedia or software assets.

3 Feature Interaction Problem for Asset Licenses

In software, a *feature* is a component of additional functionality, i.e., it extends the core functionality of the software [6]. Features are added incrementally. This

[3] http://www.gnu.org/copyleft/

can happen at different stages in the lifecycle of the software and changes are usually made by different developers. Features are often developed and tested independently, or within a particular context. However, when several features are combined, there may be interactions between the features. Interactions are behavioural modifications, in the context of software, where one feature affects the behavior of another. Such effects can be benign and even required, or adverse. Thus, the feature interaction problem concerns the coordination of features such that they cooperate towards a desired result at the application level.

Applied to asset licenses, licenses can be thought of having several features (usually referred to as *licensing clauses*) such as allowing users to create derivative works from the asset. When licenses are used together (in some way) within the same context (we intentionally avoid the term "combined" here since, as we shall see later, composition has a specific meaning for licenses), there can be conflicts between the licenses. Such conflicts take the form of license clauses (i.e. features) *affecting* clauses of another license in an adverse way.

We can conceptualize the relationship between assets and their associated licenses and the interactions of licenses as shown in Figure 1. Assets are represented as circles, and their associated licenses using earmarked rectangles. Solid lines between assets show relationships between assets. Associations between assets and their licenses are shown as directed lines, pointing from licenses to assets. Interactions between licenses are shown as bidirectional dashed lines.

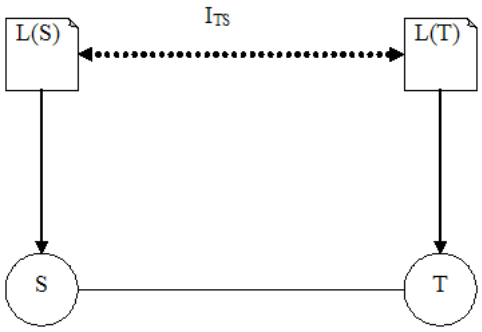

Fig. 1. Assets, Licenses, and Interactions

Licensing clauses can be classified into the following three categories based on how the interactions among them affect one another:

1. ***Independent clauses:*** These licensing clauses do not affect the resulting license. For example, a software component with a license clause similar to No-attribution of a Creative Commons (CC) license will not affect the resulting license. A No-Attribution clause leaves the choice of Attribution clause open to the resulting license, and has thus no impact on it.

2. ***Compelling clauses:*** The presence of certain clauses in a license may restrict the clauses of a resulting license, and forces the resulting license to adhere by the compelling clauses. For example, the copyleft clause of the GNU GPL makes the resulting license viral. Licensees must distribute the asset to other parties under the same terms as the GPL.
3. ***Repelling clauses:*** Certain clauses will not allow the combination with certain other licensing clauses. For example, a component with a licensing clause similar to the Non-Commercial clause of a CC license will deny the component to interact with another component under a licensing clause that allows commercial use. The Non-Commercial clause states that the licensee may not use the component for commercial purposes.

There are certain license clauses, which are broader in scope of operation than certain other clauses. Assume two assets with different license clauses, e.g. composition and derivation. If one asset allows composition, a license allowing derivation can also be used, because derivation subsumes composition. We say that derivation and composition are compatible, or derivation can be redefined as composition. The concept of redefinition (at the license clause level) is similar to the concept of redefinition of a method in a subclass [7]. Redefinition implies that two license clauses are *compatible*, if the given license clause is more permissive (accepts more) than the corresponding clause in the other license.[4]

License conflicts occur when licenses with incompatible clauses are combined. In certain cases, the absence of one or several of these clauses will not cause conflicts. Table 1 lists rules to determine the compatibility of license clauses with unspecified ("don't care") license clauses. Together with redefinition, Table 1 allows us to determine when different types of license clauses are, in effect, compatible with one another. The details of checking license compatibility are, however, out of scope for this paper, and are described in [8].

4 Types of Feature Interactions in Asset Licenses

Licenses associated with assets interact in three cases:

1. When an asset is *combined* with other assets, their associated licenses also need to be combined, and these licenses interact.
2. When new assets are *derived* from an existing asset, the license of the existing asset interacts with that of the derived one.
3. When a new *version* or release of an asset is published, the license of evolved asset interacts with the license of previous version.

We describe these feature interactions in more detail below, and provide a general template for each type. Examples of using them are given in Section 5.

[4] Two license clauses are trivially compatible if they are identical.

Specified Clause	Compatible	Rationale
Composition	*NO*	A license denying composition cannot be compatible with a license allowing composition.
Derivation	*NO*	Derivation specifies the creation of an asset based on one or more existing assets.
Attribution	*YES*	The requirement to specify attribution will not affect the compatibility when unspecified.
Sharealike	*YES*	The composite license must be similar to the license with the Sharealike clause.
Non-Commercial Use	*NO*	Commercial Use is denied by Non-Commercial Use.
Payment	*YES*	Payment clauses do not affect compatibility directly, if unspecified.

Table 1. Rules for Determining Compatibility with Unspecified Licensing Clauses

4.1 Feature Interactions During the Composition of Asset Licenses

Composition is a form of integrating assets in a way that adds value to the assets taken individually. When assets are composed, their licenses are also composed. If composition results in incompatibilities, then the corresponding assets cannot be composed. The composition of licenses can produce a set of compatible licenses for the composite asset.[5] The composite license can contain licensing clauses, which need not be present in the licenses of the assets that are composed.

Assume P and Q are assets composed to form a composite asset R, as shown in Figure 2. In the figure, I_{XY} represents the interaction between the assets X and Y. It is expected that the licenses $L(P)$ and $L(Q)$ are compatible. In turn, this requires that their license clauses $LC(P)$ and $LC(Q)$ are compatible. A detailed algorithm for checking license compatibility is provided in [8].

This composition can be represented as:

$$LC(R) \supset (LC(P) \cap LC(Q)) \cup (LC_{NEW})$$

where LC_{NEW} is a set of licensing clauses exclusive to the composite asset. We can distinguish two types of interactions between the licenses:

1. Interactions between the licenses being composed (I_{PQ})
2. Interactions between the composite license and each of the licenses being composed (I_{RP} and I_{RQ})

In addition to those direct interactions, there can be also indirect license interactions. Figure 3 provides an example. The indirect interaction $IND_{R\gamma}$ is shown by a dot-dashed line. Consider an asset Y that composes in another asset X. For example, Y is a service that provides weather information such as

[5] That is, there are multiple licenses to choose from for the composite asset that are all compatible with the composed licenses. This aspect is further explored in [8].

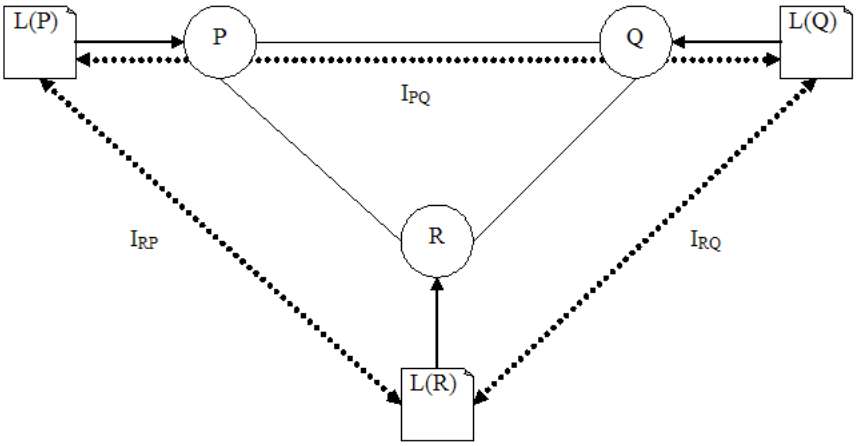

Fig. 2. License Interactions during Composition

Yahoo! Weather. In turn, it gets its weather data from another service X. Y has a licensing agreement, $L(X)$, with X for receiving the weather data. However,the copyright over the data that Y is offering as a service remains with X.

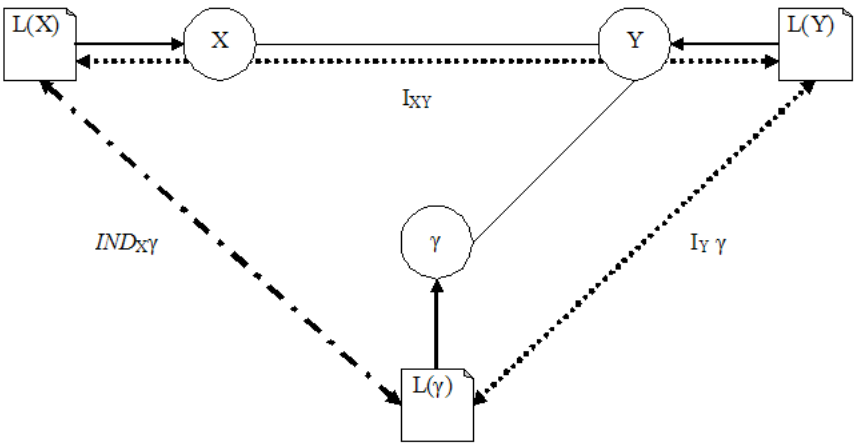

Fig. 3. Indirect License Interactions

An end user γ that wants to use the service Y is bound to the terms of a license, $L(Y)$. For example, the clauses in this license could include:

- Not to reproduce, (re)sell or exploit the service for commercial purposes.
- Not to modify, distribute or create derivative works based on the service.

– Not to access the service by any other means than through the interface provided by Y for accessing the service.

These terms restrict γ's use of Y. γ cannot use the service for any commercial purposes, nor can it derive or distribute the service. However, assume that the license of X allows any party to use the service and create derivative works based on X. The license terms imposed by Y, thus, restrict γ from doing something that $L(X)$ permits. For example, if γ were to create a derivative work of X, its license $L(\gamma)$ would now be in conflict with $L(Y)$. As the conflict arises between the licenses of γ and X via Y as the intermediary this interaction is indirect.

4.2 Feature Interactions During the Derivation of Asset Licenses

For a new asset, a new license is given by its developer. This new license might be an existing standard license like the GNU GPL. The new license can also be derived from an existing license. The concept of a *derivative license* is similar to that of derivative software in Free/Open Source Software (FOSS) [9].

A derivative license must include all licensing clauses from its parent license, but can add new clauses. The template for license derivation is as shown in Figure 4. The derivation of a license $L(T)$ from $L(S)$ can be represented as:

$$LC(T) \supseteq LC(S) \cup LC_{NEW}$$

where LC_{NEW} is a set of licensing clauses exclusive to the derivative asset.

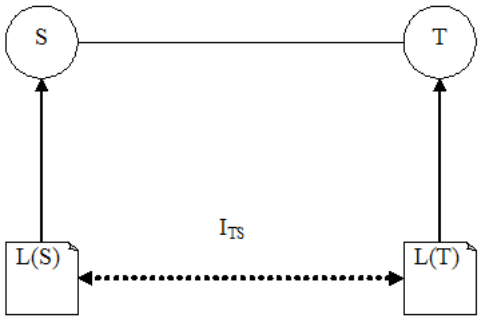

Fig. 4. License Interactions during Derivation

There can be interactions between the newly added clauses LC_{NEW} with the existing clauses $L(S)$. We expect LC_{NEW} to be compatible with $L(S)$.

4.3 Feature Interactions During the Evolution of Asset Licenses

Over time, an asset can evolve in the following ways:

– Modifications by the producer of functional or non-functional properties of the asset, represented by new releases or new versions.
– Termination of the current running asset and substitution by a new asset with different behavior.
– Same asset, but switching to a different asset license.

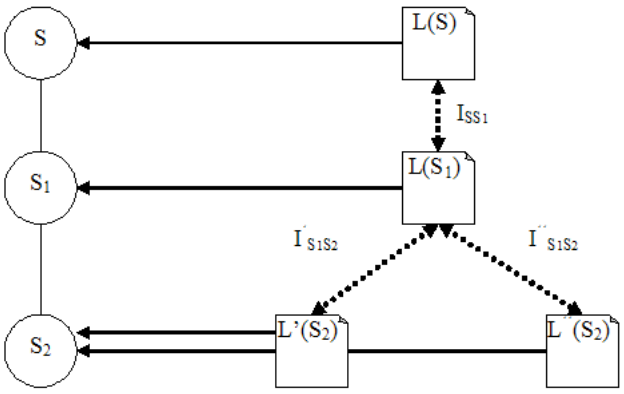

Fig. 5. License Interactions during Evolution

When an asset is released in multiple versions, each version of the asset can have a different license. However, the licenses of a particular version must not contradict that of a previous version. Here, the licensor is not creating a new license based on an existing one (different from a derivative license). The versions of licenses interact as shown in Figure 5 as the asset evolves over time.

5 License Conflicts Scenarios As Feature Interactions

As assets are composed, derived or evolved, license conflicts can arise as a result of feature interactions of licensing clauses. Below, we describe three scenarios of licensing interactions for different types of assets. We analyze each scenario to determine the cause of the license conflict in terms of the type of feature interaction it represents, and instantiate the corresponding template.

5.1 License Conflicts involving Web Services

Example 1. **(Web service composition)** Consider a restaurant service R that composes a map service M and a resource allocation service I. Assume that M allows composition and permits the service to be used for any purposes, and that I allows derivation (subsumes composition as per Section 3), but can be used only for non-commercial purposes. When M and I interact during composition,

a license conflict occurs, because I denies commercial use. Based on Table 1, these license interactions cause a conflict, resulting in the incompatible licenses.

The Non-Commercial Use feature in the license of I causes a conflict with the unspecified Non-Commercial Use feature in the license of M. If Non-Commercial Use is not specified, Commercial Use is deemed to be incompatible. Figure 6 instantiates the template for Composition of Asset Licenses.

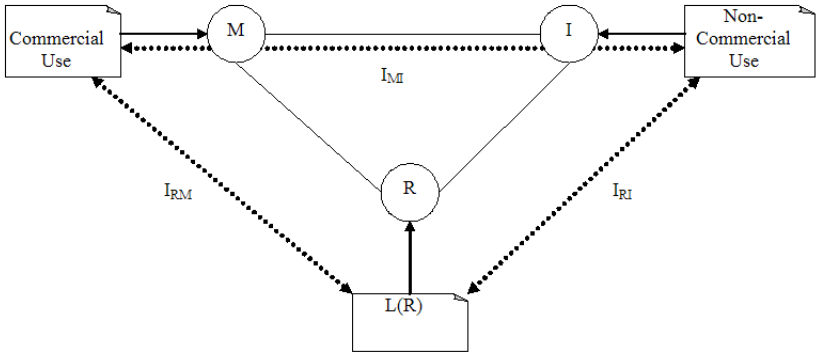

Fig. 6. Web Service Composition and License Interactions

5.2 License Conflicts involving Multimedia Files

Example 2. **(Modification and re-licensing of a music file)** Consider a music file S with a license $L_{Play,Derive}$ that allows users to *play* the file (render the asset in audio form), and to *create derivative works* based on it. Assume, that another music file T is derived from this file. If the owner of the derived music file issues the file under a license $L_{Play,Save}$ that allows users to *save* the file (save a copy including any changes to permanent storage), this license conflicts with the license of the parent music file S.

The *play* feature in the license of the parent music file does not, by itself, grant the right to store the file in modified form, although the *derive* feature allows modification. Thus, there is a conflict with the *save* feature in the license of the derived file. The interaction is the result of a Derivation of Asset Licenses. Different from Evolution of Asset Licenses, the modifications of file and license are not made by their original creator. Figure 7 instantiates the template for Derivation of Asset Licenses interactions as applied to this scenario.

5.3 License Conflicts involving Software Assets

Example 3. **(Re-releasing an asset under a new license)** Consider a software component S_1 released under the GNU General Public License (GPL)

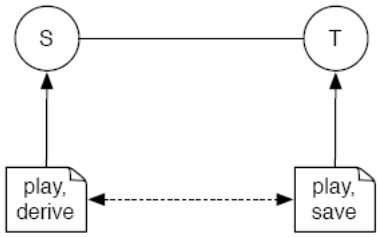

Fig. 7. Modification and Re-licensing of a Music File as License Derivation

license. At some point in the future, the licensor may decide to release a new version S_1 under two different licenses say, GNU GPL[6] and Affero GPL[7]. However, the Affero GPL is incompatible with GNU GPL version 2 because of section 2(d) that covers the distribution of application programs via web services or computer networks. Thus, the release of S_2 under Affero GPL conflicts with the license of the previous version S_1. Software released under the GNU GPL cannot be re-released under a GPL incompatible license.

This conflict is due to changes made to the license of a component. It does not fall under Derivation of Asset Licenses, however, as the licensor did not create a new license based on an existing one. Instead, the licensor re-released a new version S_2 of a component S_1 under a license that was incompatible with the existing license. This situation is shown in Figure 8, which instantiates the template for Evolution of Asset Licenses interactions. Here, the existing license was GNU GPL, which requires that software released under this license cannot be re-released under an incompatible license such as the Affero GPL.

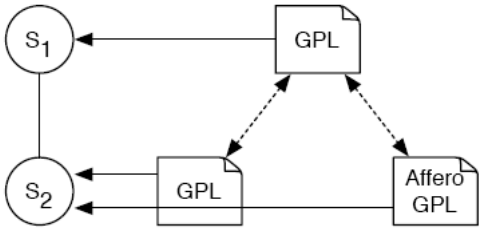

Fig. 8. Re-releasing an Asset Under a New License as License Evolution

[6] http://www.gnu.org/copyleft/gpl.html
[7] http://www.affero.org/oagpl.html

6 Related Work and Discussions

An asset is generally distributed with a license that governs what asset licensees can do. A license [1] includes all transactions between the licensor and the licensee, in which the licensor agrees to grant the licensee the right to use and access the asset under predefined terms and conditions. Asset licenses interact with one another during the course of the composition, derivation and evolution of assets, and conflicts may arise due to incompatibilities between license clauses. In our own work, we have studied the licensing of services [10, 11], and formalized licensing clauses and license compatibility for services [12].

There has been related work on policy conflicts [13–16]. Policies and licenses are similar in that they govern what an asset does, but are not the same. Policies are commonly used for access control, quality of service and other management tasks [16]. They capture high-level goals that can be enforced automatically. Policies are meant to be defined by users, allowing them to customize the behavior of a system. Policies provide the means for specifying and modulating the behavior of a feature to align its capabilities and constraints with the requirements of its users [15]. By contrast, licenses primarily focus on usage terms and access methods to assets, thus governing what users can do with an asset. They similarly modulate the *use* (not the behavior) of the asset.

A policy conflict occurs, if there are policies (for example, authentication or privacy policies) specified on two features that refer to their corresponding operations, and the policies are not compatible [13]. Policy conflicts are particularly prone to cause user confusion, as policies are often specified by users as part of customizing a feature [14]. License conflicts similarly occur due to incompatibility of licenses. However, there are unique aspects of license conflicts:

- A composite asset (aggregation of two or more assets) license should be compatible with the licenses of all the assets being composed.
- As assets evolve over time, the changes introduced (addition of a new clause or modification of an existing clause) in the licensing clauses should be compatible with the other existing licensing clauses.
- The license of a derivative asset (as it is inherited from a parent asset license) should be compatible with the parent license.

License conflicts directly preclude the creation of composite, modified, or derivative assets, and may thus loss of business opportunities.

7 Concluding Remarks

When assets are combined with other assets, whether or not they can be combined is determined by the compatibility of their associated licenses. New assets cannot be derived from existing assets, unless the license of the existing asset is compatible with that of the derived one. The evolution of assets should be consistent with the corresponding licenses. In this paper, we have modeled interactions of licenses as feature interactions, especially if those interactions result

in conflicts. Using a feature interactions view, we have described license conflicts during composition, derivation, and evolution. We are continuing our work towards the detection and (runtime) resolution of such license conflicts.

References

1. Classen, W.: Fundamentals of Software Licensing. IDEA: The Journal of Law and Technology **37**(1) (1996)
2. World Intellectual Property Organization: Successful Technology Licensing. WIPO Publishers, Geneva, Switzerland (2004)
3. Garcia, R., Gil, R., Delgado, J.: A Web Ontologies Framework for Digital Rights Management. Journal of Artificial Intelligence and Law Online First (http://springerlink.metapress.com/content/03732x05200u7h27) (2007)
4. Chavez, A., Tornabene, C., Wiederhold, G.: Software Component Licensing: A Primer. IEEE Software **15**(5) (1998) 47–53
5. Fitzgerald, B., Oi, I.: Free Culture: Cultivating the Creative Commons. Media and Arts Law Review (2004)
6. Calder, M., Kolberg, M., Magill, E., Reiff-Marganiec, S.: Feature Interaction: A Critical Review and Considered Forecast. Computer Networks **41**(1) (2003) 115–141
7. Jezequel, J.M., Train, M., Mingins, C.: Design Patterns and Contracts. Addison-Wesley (1999)
8. Gangadharan, G.R., Weiss, M., D'Andrea, V., Iannella, R.: Service License Composition and Compatibility Analysis. In: Proceedings of the International Conference on Service Oriented Computing (ICSOC'07). (2007)
9. Feller, J., Fitzgerald, B.: A Framework Analysis of the Open Source Software Development Paradigm. In: Proc. of the 21st Annual International Conference on Information Systems. (2000) 58–69
10. D'Andrea, V., Gangadharan, G.R.: Licensing Services: The Rising. In: Proceedings of the IEEE Web Services Based Systems and Applications (ICIW'06), Guadeloupe, French Caribbean. (2006) 142–147
11. Gangadharan, G.R., D'Andrea, V., Weiss, M.: Free/Open Services: Conceptualization, Classification, and Commercialization. In: Proceedings of the Third IFIP International Conference on Open Source Systems (OSS), Limerick, Ireland. (2007)
12. Gangadharan, G.R., D'Andrea, V.: Licensing Services: Formal Analysis and Implementation. In: Proceedings of the Fourth International Conference on Service Oriented Computing (ICSOC'06), Chicago, USA. (2006) 365–377
13. Sahai, A., Thompson, C., Vambenepe, W.: Specifying and Constraining Web Services Behaviour through Policies. In: Proceedings of the W3C Workshop on Constraints and Capabilities for Web Services. (2004)
14. Reiff-Marganiec, S., Turner, K.: Feature Interaction in Policies. Computer Networks **45**(5) (2004) 569584
15. Kamoda, H., Yamaoka, M., Matsuda, S., Broda, K., Sloman, M.: Policy Conflict Analysis Using Free Variable Tableaux for Access Control in Web Services Environments. In: Proceedings of the 14th Intl. World Wide Web Conference (WWW). (2005)
16. Turner, K., Blair, L.: Policies and Conflicts in Call Control. Computer Networks **51** (2007) 496–514