Approximating Fair Use in LicenseScript

Cheun Ngen Chong, Yee Wei Law, Sandro Etalle, and Pieter H Hartel

Faculty of EEMCS, University of Twente, The Netherlands
{chong,ywlaw,etalle,pieter}@cs.utwente.nl

Abstract. Current rights management systems are not able to enforce copyright laws because of both legal and technological reasons. The *contract rights* granted by a copyright owner are often overridden by the users' *statutory rights* that are granted by the laws. In particular, *Fair Use* allows for "unauthorized but not illegal" use of content. Two technological reasons why fair use cannot be upheld: (1) the current XML-based rights expression language (REL) cannot capture user's statutory rights; and (2) the underlying architectures cannot support copyright enforcement. This paper focuses on the first problem and we propose a way of solving it by a two-pronged approach: (1) rights assertion, to allow a user assert new rights to the license, i.e. freely express her rights under fair use; and (2) audit logging, to record the asserted rights and keep track of the copies rendered and distributed under fair use. We apply this approach in LicenseScript (a logic-based REL) to demonstrate how LicenseScript can *approximate* fair use.

Keywords: Intellectual property rights, copyright, fair use, rights expression language, metadata security.

1 Introduction

Current rights management systems are basically not able to enforce properly copyright laws. The reason is both legal and technological and lies mainly in the fact that user's rights are a result of the reconciliation of two different and often conflicting rulings. On one hand there are the the rights granted *by contract* by the copyright owner (e.g. author or digital library) to a user; these are called *contract rights* because they are granted when user agrees on the terms and conditions imposed by the copyright owner. On the other hand, there exist *statutory rights* granted by the law.

An example of statutory right is the right of *fair use* [9]. Contract rights and statutory rights often contradict each other: a contract may for instance forbid making copies of a given book, while the law grants the user to make copies for educational use. Statutory rights depend on a number of *circumstances*. For instance, according to the United States Codes (U.S.C) (`http://uscode.house.gov/`), Section 107 Title 17 Chapter 1 (Fair Use Doctrine), "fair use of copyrighted content, including reproduction for purposes such as criticism, comment, news reporting, teaching, scholarship, or research does not violate or infringe the copyrights".

In general, statutory rights are restricted by the contract rights in the rights management systems. In other words, from the legal perspective, the copyright owner holds far more control than the copyright laws endorse [10]. Questions of the legality of overriding the statutory rights by contract rights are yet to be answered [5], however the legal perspective is beyond the scope of the paper.

To fully understand why it is impossible to render this situation in current rights management systems we have to take a look at their structure; which consists of: (1) a rights expression language (REL) and (2) an underlying architecture. A REL provides a vocabulary, associated with a set of grammatical rules, to express fine-grained usage control over digital content. A *license* is written in a REL and governs the terms and conditions under which the digital content should be used. The most widely-used RELs are XML-based: XrML [6] and ODRL [7].

Mulligan and Burstein [8] have pointed out the inadequacies of the aforementioned XML-based RELs in expressing a user's statutory rights: (1) the RELs can only describe contract rights; (2) the RELs provide insufficient support for rights assertion by the user; and (3) the RELs cannot provide contextual information consistent with the copyright laws that accommodate the user's statutory rights. In short, the user's statutory rights become "unauthorized" under the contract rights because they cannot be captured in the license written in an XML-based REL. On the other hand, the user's statutory rights must be upheld under the copyright laws. In other words, fair use allows the users to exercise "unauthorized but not illegal" rights. In addition, it is neither a legal nor a practical requirement for users to declare these rights explicitly before enjoying these rights. Last but not least, the architecture cannot determine if some content is used for non-profit or commercial purposes [4].

Although it is impossible for a REL to capture the semantics of fair use completely we may *approximate* fair use [8].

In this paper, we propose a method for implementing a digital right management system that takes into account statutory right. For this we refer to the LicenseScript Right Expression Architecture [2], and we use a two two-pronged approach based on (1) *rights assertion*; and (2) *audit logging* (see Figure 1). To the best of our knowledge, this is the first attempt to approximate fair use by using a REL. We elaborate this approach in the later sections.

This paper is organized as follows. Section 2 introduces our approach to approximating fair use. Section 3 briefly explains the LicenseScript language with a simple scenario. Section 4 details our approach to approximating fair use in LicenseScript. Section 5 describes briefly some related work. Finally, section 6 concludes this paper and presents future work.

2 Our Approach

In this section, we explain how LicenseScript may be used to approximate fair use. As mentioned in section 1, we are using a two-pronged approach (Figure 1): (1) *Rights assertion*: to allow the user assert new fair use-compliant rights in addition to the rights dictated by the license; and (2) *Audit logging*: to keep a record of the rights asserted by the user and the copies of licenses created by the user.

Figure 1 shows that Bob issues a license to Alice, allowing Alice to make copies of the license (and therefore she is able to copy the licensed content too). Alice in turn gives a copy of the license to Charles. Alice perform rights assertion on the license before making copies of the license. All actions performed by the users (i.e. Alice and Charles) are logged in the appropriate license.

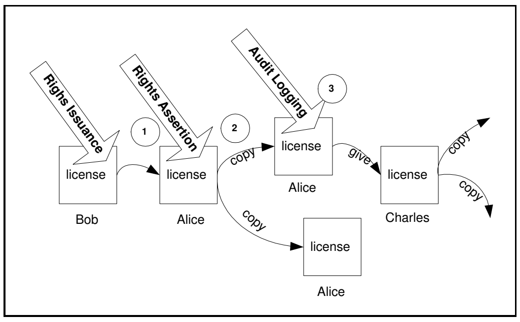

Fig. 1. Our approach to approximating fair use.

Using this approach, on one hand the users can freely exercise their statutory rights; on the other hand, the copyright owner can track the source of possible copyright infringement. Note that our proposal is more advanced than *rights issuance*, which is performed by the copyright owner for issuing and granting rights to a user. Rights issuance is already being supported by existing RELs.

We use the following illustrative scenario of a *digital library* to aid in our explanation in the next two sections:

Example 1. Alice borrows an ebook, entitled "An Example Book" from Bob's Digital Library (herefrom we simply call it Bob). Bob sends the license to Alice, allowing her to view, copy and give the rendered copies of the ebook to other users.

In subsequent sections, we elaborate how the two-pronged approach mentioned earlier can be used to approximate fair use.

2.1 Rights Assertion

Imagine a license for Alice, which states the following rights that are granted to Alice by Bob: *view*, *copy*, *give* and *assert*. Suppose Alice wants to print the ebook. The license does not state the *print* right. Therefore, Alice must assert a new *print* right by adding this right to the license. We believe that the users ability to assert new rights contributes to fair use because the user can express their rights according to their will, in addition to the rights granted by the copyright owner.

We make a few what we believe to be reasonable assumptions. Alice must have a content renderer, in this case an ebook viewer to use the ebook. A set of rules are embedded in the firmware of this ebook viewer. Bob may define these rules. Bob may not trust Alice, but he trusts the rules he defines. If Alice's asserted right in the license can be exercised by any corresponding rule Bob defines, Bob may logically trust this right (unless the asserted right causes conflicts in the license, which we will discuss later). This is because Alice's asserted rights must conform to the semantics of the rules. The implicit assumption is that the content renderer is secure.

Bob may specify some of the contextual information, e.g. usage purpose, location of use etc. that the fair use doctrine refers to (as discussed in section 1) by using LicenseScript. Then, Bob can write the rules such that oblige Alice to provide the contextual information. The rules then validate the provided information using the contextual information stated in the doctrine. In other words, Alice must declare her intention to perform fair use. The attestation of this declaration is performed by the underlying architecture (presumably by using some cryptographic means). We consider architectural support to enforce all this as our future work. Here we are concerned only with a higher level of abstraction that defines *what* may or may not happen, and not *how* actions may be performed.

2.2 Audit Logging

Alice should not be able to assert arbitrary rights nor must she be able to override existing rights (in the license) that may undermine the rights management system. While we might be able to avoid some problems by syntactic checks (e.g. to check for duplication of rights in the license caused by the rights assertion), many other potential ambiguities will remain (e.g. if the rights asserted can expire). Therefore, we record all the asserted rights (along with the user's identity, the date the right is asserted and the purpose of asserting the rights) in the license.

Bob may check the record and the license if the asserted rights have overridden or violated the contract rights. Therefrom, Bob may take further action, e.g. to allow/disallow the Alice's asserted right or to take Alice to court if the asserted rights violate the copyrights or the contract rights.

Additionally, Bob also tracks the copies of the licenses distributed by Alice. We can put a history record in the license to log this distribution pattern. Thus, the copyright owner can trace the distribution of the licenses by inspecting the history record in these licenses. This helps the copyright owner track possible sources of copyrights infringement. Audit logging requires cryptographic support from the underlying architecture. We have already addressed the issue of secure audit logging in our previous work [3].

This concludes the introduction to our two-pronged approach towards approximating fair use. We will now present the details of the approach using LicenseScript.

3 LicenseScript Language

LicenseScript [2] is a language that is based on (1) multiset rewriting, which captures the dynamic evolution of licenses; and (2) logic programming, which captures the static terms and conditions on a license. LicenseScript provides a judicious choice of the interfacing mechanism between the static and dynamic domains.

A license specifies when certain operations on the object are permitted or denied. The *license* is associated with the *content*, as can be seen in Figure 2. A license carries *bindings*, which describe the attributes of the license; and *clauses*, which determine if a certain operation is allowed (or forbidden). The license clauses consult the license bindings for their decision making and may also alter the values of the license bindings.

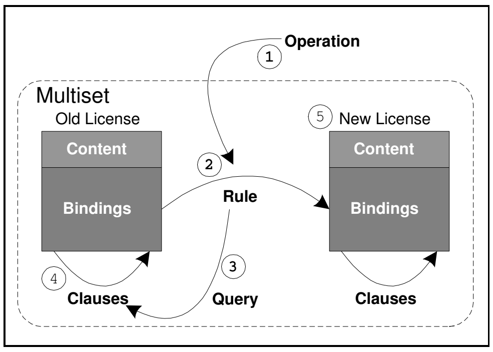

Fig. 2. Transformation of licenses.

Licenses are represented as terms that reside in multisets. A multiset can be thought as part of the user's system. For the specification of a license, we use logic programming. The readers are thus assumed to be familiar with the terminology and the basic results of the semantics of logic programs.

Figure 2 illustrates that ① an *operation* (performed by a subject) ② invokes a *rule* in the multiset. The rule then generates and executes a ③ *query* on the ④ license clauses and bindings. The ⑤ execution result of the rule is a newly generated license. Now we use a simple illustrative scenario to explain this process:
Example 2. Amanda gets an ebook, titled "A Book" from Ben Publisher. Ben issues a license with an expiry date fixed at "23/06/2004".

This license allows Amanda to print two copies of the ebook (`L01,...,L14` are line numbers included for reference purposes, they are not part of the code):

```
license(ebook:a_book,                      L01
[   (canprint(B1,B2,User) :-               L02
        get_value(B1,consumer,C),          L03
        C = User,                          L04
        get_value(B1,expires,Exp),         L05
        today(D), D>Exp,                   L06
        get_value(B1,printed,P),           L07
        get_value(B1,max_prints,Max),      L08
        P < Max,                           L09
        set_value(B1,printed,P+1,B2)],     L10
[   (company=ben_publisher),               L11
    (consumer=amanda),                     L12
    (expires=23/06/2004),                  L13
    (max_prints=2), (printed=0)      ])    L14
```

A license is represented by a term of the form `license(content,C,B)`, where `content` is a unique identifier referring to the real content; `C` is a list of *license clauses* consisting of Prolog programs describing when operations are permitted or denied; and `B` is a list of *license bindings* capturing the attributes of the license. We define two multiset-rewrite *rules*, as shown below, to model the interface between the system and the licenses. The rules can be thought of as a firmware in the user's system. The user's content renderer would contain the rules as embedded firmware. Only the copyright owner (or a trusted third party on behalf of the copyright owner) can define a set of rules for the firmware of the content renderer.

The syntax of the rules is based on the Gamma notation [1] of multiset rewriting (again, `R01,...,R04` are line numbers):

```
print(Ebook,User) :                        R01
    license(Ebook,C,B1) ->                 R02
    license(Ebook,C,B2)                    R03
 <= C |- canprint(B1,B2,User)              R04
```

We will step through the example assuming Amanda would like to print the eBook with the available license on her system (as shown above):

1. Amanda's system wants to know whether she has the print *right* on the ebook. This is achieved by applying the print *rule* (line `R01`) with appropriate parameters: `print(ebook:a_book,amanda)`.
2. The rule finds the `license(ebook:a_book,[...],[...])` (line `R02`, line `L01`) in the system. The first list refers to the license clauses (lines `L02`–`L14`), while the second list refers to the set of license bindings (lines `L15`–`L18`).
3. The rule then executes a *query* in the form of `canprint(B1,B2,User)` (line `R04`), where `B2` designates the output generated by the query; This will form a new set of license bindings.
4. The license interpreter retrieves the value of the license binding `consumer` from the list of license bindings `B1` (line `L03`) and compares the retrieved value with the user identity `User` (line `L04`). `User` is passed in as an argument to the license clause (line `L02`).
5. Similarly, the interpreter retrieves the value of the binding `expires` (line `L05`).
6. The interpreter calls the primitive (which is discussed later) `today(D)` (line `L06`) to obtain the current time and date.
7. The expiry date of the license must be greater than current time and date (line `L07`).
8. Similarly, the value of `printed` is checked if it is smaller than the value of `max_prints` (lines `L07`–`L09`).
9. If all conditions are satisfied (the user is valid, the license has not expired and the number of printed copies does not exceed the allowable maximum copies), the query returns `yes` (line `R04`) to the interpreter, with the newly generated license bindings in `B2`.
10. The value of the license binding `printed` is incremented (line `L10`) every time the print operation succeeds.
11. The value `yes` indicates that the execution of `canprint(B1,B2,User)` yields *success* in the license clauses `C`.
12. The rule `print(Ebook,User)` generates a new license with the newly generated license bindings `license(Ebook,C,B2)` (line `R03`).

There are a number of primitives to model the interface of the system with the license (interpreter): (1) `get_value(B,n,V)`, to report in `V` the value of `n` from `B`; (2) `set_value(B1,n,V,B2)`, to give value `V` to `n` in `B2`; (3) `today(D)`, to bind `D` to the current system date/time; and (4) `identify(L)`, to identify the current environment to `L`. For further details of the LicenseScript language, see Reference [2].

In the following section, we explain how LicenseScript can be used to approximate fair use.

4 Fair Use in LicenseScript

As we have seen, licenses are just objects in the multiset. Many other types of objects can be modelled, such as wallets and policies. We use this LicenseScript-specific feature to define (1) the *license* issued by Bob to Alice, which allows her to *view*, *copy* and *give* the ebook, as well as *assert* new rights (section 4.1); (2) the *doctrine* that carries the contextual information consistent with fair use (section 4.3); (3) the *record* that logs Alice's asserted rights (section 4.2); and (4) the *rules* defined by Bob as the firmware of Alice's system, which include *view*, *copy*, *give*, *print* and *assert* (section 4.4).

4.1 The `license`

Following Example 1, this is the `license` that Bob issues to Alice:

```
license(ebook:an_example_book,
[   (canloan(B1,B2,Loaner,User) :-
        get_value(B1,digital_library,L), L=Loaner,
        get_value(B1,loaned,Loaned), Loaned=false,
        set_value(B1,loaned,true,B2), set_value(B1,user,User),
        today(D), set_value(B1,expires,D+7,B2)),
    (canreturn(B1,B2) :-
        set_value(B1,loaned,false,B2)),
    (canview(B1,B2,User) :-
        get_value(B1,user,U), U=User,
        get_value(B1,expires,Exp), today(D), D>Exp),
    (cancopy(B1,B2,B3,User) :-
        get_value(B1,user,U), U=User, get_value(B1,expires,Exp),
        today(D), D>Exp, get_value(B1,copies,N),
        append([(User,D),N,NN), set_value(B1,copies,NN,B2),
        set_value(B1,copies,NN,B3)),
    (cangive(B1,B2,User1,User2) :-
        get_value(B1,user,U), U=User1,
        set_value(B1,user,User2,B2), get_value(B1,trace,T),
        today(D), append([(User1,D,User2)],T,T2),
        set_value(B1,trace,T2,B2)),
    (canassert(C1,C2,B1,B2,Clause,Binds,User,Purpose) :-
        get_value(B1,user,U), U=User,
        get_value(B1,expires,Exp), today(D), D>Exp,
        get_value(B1,asserted,As),
        append([(User,D,Purpose)],Clause,NC),
        append(NC,As,As2), set_value(B1,asserted,As2,B2),
        append(C1,Clause,C2), append(B1,Binds,B2)),
    (canperform(B1,B2,User) :-
        get_value(B1,expires,Exp), today(D), D>Exp,
        get_value(B1,user,U), U=User)                        ],
[   (user=alice), (digital_library=bob), (loaned=true),
    (asserted=[]), (trace=[]), (copies=[]), (expires=15/8/03) ])
```

The license clause `canloan` determines if the ebook can be loaned to the user, and *only* by the digital library. The return date (represented by `expires`) is set at the seventh day from the date this ebook is loaned. The license clause `canreturn` is the counterpart of the license clause `canloan`. It resets the binding `loaned` to `false`.

The license clause `canview` determines that *only* Alice (the user) can view the ebook and the return date `expires` has not expired. The license clause `canassert` allows Alice to assert new rights (represented as the `Clause` with necessary bindings `Binds`).

The license clause `canperform` determines if the user who performs fair use is the genuine user who owns the license. The function `append` is a built-in Prolog program that produces a new list (`C2`) by combining two lists (`C1` and `Clause`). The license binding `asserted` records all the rights asserted by the user. The license binding `trace` records the distribution of the license when it is given away. The license binding `copies` records the user who generates a new copy of this license.

4.2 The `record`

The `record` that belongs to Bob and which logs the rigths asserted by Alice to a license looks like this:

```
record(ebook:an_example_book,
[    (canlog(B1,B2,User,Action) :-
         get_value(B1,history,H), today(Date),
         append([(User,Action,Date)],H,NH),
         set_value(B1,history,NH,B2))                       ],
[    (history=[]), (digital_library=bob)                    ])
```

The term `record` records (clause `canlog`) all the actions (the argument `Action`) performed by the user (the argument `User`) at the current time (the value `Date`) on the ebook.

4.3 The `doctrine`

The `doctrine` (defined by Bob) that encodes the contextual information of fair use is:

```
doctrine(fairuse,
[    (canallow(B1,B2,Purpose) :-
         get_value(B1,purposes,Ps), member(Purpose,Ps),
         identify(Loc), get_value(B1,location,L), L=Loc)     ],
[    (purposes=[criticism,comment,newsreport,
                eduction,scholarship,research]),
     (location=united_states_of_america)                    ])
```

The license clause `canallow` determines if the purpose attested by the user for using the content is under the fair use context and if the user is in the U.S. The license binding `purposes` state all the usage purposes allowed under the fair use doctrine. The license binding `location` indicates that this doctrine applies in United States. The copyright owner can define different types of `doctrine`, e.g. first sale `doctrine` to encapsulate the corresponding contextual information.

4.4 The rules

The rules that Bob defines for Alice's firmware are:

```
loan(Ebook,User) :
     license(Ebook,C,B1) -> license(Ebook,C,B2)
 <= C |- canloan(B1,B2,User)
return(Ebook) :
    license(Ebook,C,B1) -> license(Ebook,C,B2)
 <= C |- canreturn(B1,B2)
view(Ebook,User,Purpose) :
    license(Ebook,C,B1) -> license(Ebook,C,B2)
 <= C |- canview(B1,B2,User)
view(Ebook,User,Purpose) :
    doctrine(fairuse,Cp,Bp1),license(Ebook,C,B1) ->
    doctrine(fairuse,Cp,Bp2),license(Ebook,C,B2)
 <= C |- canperform(B1,B2,User), Cp |- canallow(Bp1,Bp2,Purpose)
copy(Ebook,User) :
    license(Ebook,C,B1) -> license(Ebook,C,B2),license(Ebook,C,B3)
 <= C |- cancopy(B1,B2,B3,User)
copy(Ebook,User,Purpose) :
    doctrine(fairuse,Cp,Bp1),license(Ebook,C,B1) ->
    doctrine(fairuse,Cp,Bp2),license(Ebook,C,B1),
    license(Ebook,C,B2)
 <= C |- canperform(B1,B2,User), Cp |- canallow(Bp1,Bp2,Purpose)
give(Ebook,User,Purpose) :
    license(Ebook,C,B1) -> license(Ebook,C,B2)
 <= C |- cangive(B1,B2,User1,User2)
give(Ebook,User,Purpose) :
    doctrine(fairuse,Cp,Bp1),license(Ebook,C,B1) ->
    doctrine(fairuse,Cp,Bp2),license(Ebook,C,B2)
 <= C |- canperform(B1,B2,User), Cp |- canallow(Bp1,Bp2,Purpose)
print(Ebook,User,Purpose) :
    license(Ebook,C,B1) -> license(Ebook,C,B2)
 <= C |- canprint(B1,B2,User)
print(Ebook,User,Purpose) :
    doctrine(fairuse,Cp,Bp1),license(Ebook,C,B1) ->
    doctrine(fairuse,Cp,Bp2),license(Ebook,C,B2)
 <= C |- canperform(B1,B2,User), Cp |- canallow(Bp1,Bp2,Purpose)
assert(Ebook,Clause,Binds,User,Purpose) :
    license(Ebook,C1,B1),record(Ebook,Cr,Br1) ->
    license(Ebook,C2,B2),record(Ebook,Cr,Br2)
 <= C1 |- canassert(C1,C2,B1,B2,Clause,Binds,User,Purpose)
    Cr |- canlog(Br1,Br2,User,Clause)
```

The `assert` rule says: to assert a new `Clause` with corresponding `Binds`, the user must show her identity `User` and states the `Purpose` of asserting this right; and the user's asserted clause is recorded by the term `record` (see section 4.2).

The first `view` rule says: to view the `Ebook`, the user must present her identity `User` and declare to the usage `Purpose`; the license must contain the license clause `canview`. If the first rule does not apply to Alice's execution, the second rule may be executed: the usage `Purpose` attested by the `User` must conform to the contextual information stated in the `doctrine` (see section 4.3).

The rules show how the various objects in the multiset are used, in a cooperative fashion to achieve fair use. For example, as shown in Figure 1, ① Alice has asserted a print right to the license (as shown in section 4.1) on "`1/8/2003`" for the purpose of education (by executing the `assert` rule). The new license is as follows:

```
license(ebook:an_example_book,
[    ...
     (canprint(B1,B2,User) :-
         get_value(B1,onlyalice,U), U=User)                    ],
[    ...
     (asserted=[(alice,1/8/2003,education),
       (canprint(B1,B2,User):-get_value(B1,onlyalice,U),U=User)]),
      (onlyalice=alice)                                        ])
```

(Herefrom, the symbol "..." represents the unchanged part of the object.)

The asserted clause `canprint` allows *only* Alice to print the ebook (she adds a new binding, namely `onlyalice` to the license). Alice can execute the `print` rule to print the ebook with the asserted *print* right. The asserted right is logged at the license binding `asserted` and Bob's `record` (from section 4.2) becomes:

```
record(ebook:an_example_book,
[    ...                                                       ],
[    ...
     (history=[(alice,
       (canprint(B1,B2,User):-get_value(B1,user,U),U=User),
       1/8/03)])                                               ])
```

Now, ② Alice makes an additional copy of the license by executing the `copy` rule. The third version of the license becomes:

```
license(ebook:an_example_book,
[    ...
     (canprint(B1,B2,User) :-
         get_value(B1,onlyalice,U), U=User)                    ],
[    ...
     (asserted=[(alice,1/8/2003,education),
       (canprint(B1,B2,User):-get_value(B1,onlyalice,U),U=User))]),
     (onlyalice=alice), (copies=[(alice,1/8/2003)])            ])
```

Alice's copy action is logged in the binding `copies`. ③ Alice gives a copy of the license to Charles on "`2/8/03`" (by executing the `give` rule). Charles' license (given by Alice) will look like this:

```
license(ebook:an_example_book,
[    ...
     (canprint(B1,B2,User) :-
         get_value(B1,onlyalice,U), U=User)                    ],
[    ...
     (asserted=[(alice,1/8/2003,education),
       (canprint(B1,B2,User):-get_value(B1,onlyalice,U),U=User))]),
     (onlyalice=alice), (copies=[(alice,1/8/2003)]),
     (trace=[(alice,2/8/03,charles)]), (user=charles)          ])
```

The distribution is logged in the license binding `trace`. The value of the binding `user` is assigned to `charles` indicating the transfer of the ownership of this license.

We have demonstrated how rules can transform the objects in the multiset by using the example as illustrated in Figure 1. This concludes the detailed description of the approach to approximating fair use in LicenseScript.

5 Related Work

Mulligan and Burstein [8] suggest several extensions of the XML-based RELs to approximate fair use. We summarize their suggestions as follows: (1) to define a set of rights that might simulate some "default" rights the users have with physical copies of the content, e.g. for a music album, the default rights may be *play*, *rewind*, *seek*, *excerpt* and *copy*; (2) to provide some contextual information description in the REL to support the fair use modelling, e.g. the usage purpose etc.

Similar to Mulligan and Burstein first suggestion, our approach constrains the content user's fair use rights by the firmware rules. However, the copyright owners (or content providers) cannot make the predictions on how the users would use the content. Therefore, it is a cumbersome process for the copyright owners to define a set of default rights for all available content. Our approach, on the other hand, allows user to freely express their rights. At the same time, the copyright owner may control the user's fair use actions to the extent confined by the rules. The copyright owner may flexibly define some contextual information in LicenseScript that consistent with the fair use that the rules may comply with.

Secure Telecooperation SIT, Darmstadt and the Fraunhofer Institute for Integrated Circuits, Erlangen, Ilmenau have developed the Light Weight Digital Rights Management System (LWDRM) (`http://www.iis.fraunhofer.de/amm/techinf/ipmp`). They introduce two distinct file formats, namely local media format (LMF) and signed media format (SMF). The LMF is bound to the machine where the content is generated, whereas the SMF is intended for small-scale distribution. The SMF is generated when the user mark the content with her personal digital signature.

There are three levels of functionality defined in the LWDRM. The first level is the LWDRM player, to play the SMF/LMF content. The second level allows the user to generate LMF from the content. This level offers more extensive features to the user, e.g. improved compression algorithms etc. The third level allows the user to sign the LMF content, i.e. to generate the SMF content. Thereby, the user could distribute and use this SMF content in other machines. LWDRM may track the leak the copyright infringement by using the signature. However, the user must willingly sacrifice her privacy to perform fair use. Our approach is complementary to the SIT approach, in that we manipulate licenses where SIT manipulate content.

6 Conclusion and Future Work

Current rights management system can only enforce contract rights that are granted by the copyright owners to the users. Other rights, such as the statutory rights granted by copyright laws cannot be enforced in the rights management systems. Fair use is an example of statutory rights, because fair use allows "unauthorized but not illegal" actions.

In this paper, we focus on one aspect of the technological issues related to the rights expression language (REL). We argue that current RELs (1) cannot capture user's statutory rights, (2) do not support rights assertion performed by the users, and (3) cannot provide useful contextual information that is consistent with the fair use. We have introduced a two-pronged approach for approximating fair use in LicenseScript: *rights assertion* and *audit logging*. Then, we have demonstrated the use of LicenseScript to *approximate* fair use.

We would also like to investigate if LicenseScript is capable of expressing other copyright laws, e.g. first-sale doctrine as well as privacy protection in the future. In addition, we are implementing our architecture for LicenseScript, namely the LicenseScript Engine.

Acknowledgement

We would like to thank (in alphabetical order) Aaron Burstein, Séverine Dusollier, Lucie Guilbault, Pamela Samuelson and Mark Stefik for their valuable help with this paper.

References

1. J-P. Banâtre, P. Fradet, and D. L. Métayer. Gamma and the chemical reaction model: Fifteen years after. In C. Calude, G. Paun, G. Rozenberg, and A. Salomaa, editors, *Workshop on Multiset Processing (WMP)*, volume 2235 of *Lecture Notes in Computer Science*, pages 17–44. Springer-Verlag, Berlin, August 2001.
2. C. N. Chong, R. Corin, S. Etalle, P. H. Hartel, W. Jonker, and Y. W. Law. LicenseScript: A novel digital rights language and its semantics. In *3rd International Conference on Web Delivering of Music (WEDELMUSIC)*, page to appear, Los Alamitos, California, United States, September 2003. IEEE Computer Society Press.
3. C. N. Chong, Z. Peng, and P. H. Hartel. Secure audit logging with tamper-resistant hardware. In *18^{th} IFIP International Information Security Conference (IFIPSEC)*, pages 73–84. Kluwer Academic Publishers, May 2003.
4. E. W. Felten. A skeptical view of DRM and fair use. *Communications of ACM*, 46(4):57–59, April 2003.
5. L. Guibault. Copyright limitations and contracts: Are restrictive click-warp license valid? *Journal of Digital Property Law*, 2(1):144–183, November 2002.
6. H. Guo. Digital rights management (DRM) using XrML. In *T-110.501 Seminar on Network Security 2001*, page Poster paper 4, 2001. http://www.tml.hut.fi/Studies/T-110.501/2001/papers/.
7. R. Iannella. Open digital rights management. In *World Wide Web Consortium (W3C) DRM Workshop*, page Position paper 23, January 2001. http://www.w3.org/2000/12/drm-ws/pp/.
8. D. Mulligan and A. Burstein. Implementing copyright limitations in rights expression languages. In J. Feigenbaum, editor, *Proceedings of 2002 ACM CCS-9 Workshop on Security and Privacy in Digital Rights Management*, volume 2696 of *Lecture Notes in Computer Science*, page To appear. Springer-Verlag, November 2002.
9. D. K. Mulligan. Digital rights management and fair use by design. *Communications of ACM*, 46(4):31–33, April 2003.
10. P. Samuelson. Digital rights management {and,or,vs.} the law. *Communications of ACM*, 46(4):41–45, April 2003.