Chapter I
Rights Expression Languages

Pramod A. Jamkhedkar  
University of New Mexico, USA

Gregory L. Heileman  
University of New Mexico, USA

ABSTRACT

Rights expression languages (RELs) form a central component of digital rights management (DRM) systems. The process of development of RELs transforms the rights requirements to a formal language ready to be used in DRM systems. Decisions regarding the design of the conceptual model, syntax, semantics, and other such properties of the language, affect not only each other, but also the integration of the language in DRM systems, and the design of DRM system as a whole. This chapter provides a detailed analysis of each step of this process and the tradeoffs involved that not only affect the properties of the REL, but also the DRM system using that REL.

INTRODUCTION

Every time there has been progress in the ability to make copies of some intellectual work, the need for copyright has arisen. Traditionally, copyright has been expressed using natural languages, in sufficient detail, so as to have standing in a court of law. The advent of computers, the Internet, and digital content, has created the need for management of copyright electronically, also known as digital rights management (DRM). To manage copyright electronically, it is necessary that copyright agreements be expressed in a machine-readable form. Computers or intelligent devices can then interpret copyright agreements, and ensure that usage of copyrighted digital content is in accordance with the copyright agreement associated with the content. In DRM terminology, languages used to express copyright agreements in a machine-readable form are called rights expression languages (RELs).

Over the past several years, a number of RELs have been developed, with the eXtensible rights Markup Language (XrML) (“XrML 2.0 Technical Overview”, 2002) and the Open Digital Rights Language (ODRL) (Iannella, 2002) becoming...
the most popular. Recently, XrML was adopted as the standard REL for inclusion in the MPEG-21 standard (“The MPEG-21 Rights Expression Language”, 2003) and ODRL was accepted by the Open Mobile Alliance as the standard REL for mobile content (“Enabler Release Definition for DRM V2.0.”, 2003). Nevertheless, these RELs have not been extensively used in applications, despite the fact that many businesses suffer from problems that could be solved using appropriate DRM technologies. For instance, Apple and Microsoft have created their own lightweight DRM technologies, iTunes and Windows Media DRM, respectively, that do not make use of any commercially available REL. Lack of standardized general purpose RELs has been one of the major reasons for the fragmented nature of the DRM industry.

The fragmented nature of the DRM industry, in turn has led to a lack of interoperability, and this is one of the major reasons for limited acceptance of DRM among content users. RELs play a major role in influencing the design of DRM systems. To understand the problems with DRM systems, it is therefore necessary to understand the process underlying the development of RELs, along with their role in DRM systems. There are many aspects to the process of developing a REL, which aims to map the rights requirements to a machine-readable language that is sufficiently expressive to capture these requirements. These properties include the structure, syntax, and semantics, among many other features of RELs. In this paper, we study the factors that influence the design of RELs, in terms of these properties, and how these properties, in turn affect each other and the design of DRM systems.

There has been other literature on this topic, especially ones that provide a comprehensive survey of RELs. Coyle (2004) provides an overview of the different elements of RELs along with important ways to analyze RELs. The discussion is based on four leading REL initiatives of the time, namely, ODRL, XrML, Creative Commons and METS Rights. Guth (2003) provides a similar analysis of RELs, along with additional discussion on applications supplemented with examples on sample licenses of different RELs. Barlas (2006), similarly, provides a comprehensive discussion on the role of RELs, along with an explanation of different RELs along with various standards adopted. Wang (2005) provides an analysis on the design principles of RELs, in which several issues such as interoperability, extensibility, identification, etc. are discussed. Jamkhedkar, Heileman, and Martinez-Ortiz (2006) provide another view for the design of RELs, in which they propose refactoring and simplification of RELs to allow easy interoperability and formalization. This paper provides a different approach in which it discusses different stages in the development of RELs, along with an emphasis on formal rights expression calculus.

The rest of the paper is divided into four sections followed by the conclusions. The following section provides the history of the attempts to develop RELs, along with the current RELs developed in industry and academia. This is followed by a section that provides an overview of the process of the development of a REL. After that we discuss the development of a conceptual model for rights, and the issues involved in the process. In Section “Formalization of RELs”, we discuss two types of RELs, namely XML-based and logic-based, and how formalization is achieved in these RELs. This is followed by a discussion on, how the decisions in the process of development of RELs affect the DRM system properties such as trust management, interoperability, and other system properties. Finally, we provide some useful conclusions.

AN OVERVIEW OF RIGHTS EXPRESSION LANGUAGES

Some of the earliest attempts to develop a formal language for expression of legal discourse date
back to the late 1980’s. McCarthy (1989) proposed a Language for Legal Discourse (LLD) that was based on a logical framework. The central idea underlying LLD was to develop a deep conceptual model. Such a model is created by selecting a small set of common categories such as, space, time, action, permissions, obligations, constraints, and so on, relevant to a particular legal domain, and then developing a knowledge representation language that reflects the structure of this set.

A legal domain that gained increasing interest in the early 1990’s was copyright law. Radical changes in information technology coupled with the evolution of the Internet drastically disturbed the hitherto maintained balance between intellectual property owners and consumers. Intellectual property owners pressured technologists to develop effective DRM systems to prevent violation of copyright by consumers. The central requirement of any DRM system is a machine-readable knowledge representation language for copyright contracts, known RELs (Jamkhedkar and Heileman, 2004). Rights enforcement mechanisms, operating on consumer devices, interpret copyright statements written in these languages to manage usage of intellectual property by consumers.

The precursors of the current RELs were developed in the early 1990’s to address this requirement. In 1994, Stefik and Casey (1994) filed a patent for DRM technology he developed at Xerox PARC. This included the description of a “usage rights grammar” that was subsequently implemented in LISP and called the Digital Rights Property Language (DRPL). Through a evolutionary process DPRL became XrML. Specifically, in 1998, an eXtensible Markup Language (XML) implementation of DRPL (version 2.0) was released by Xerox PARC. Then, in 2000, ContentGuard, a Xerox/Microsoft joint venture, released XrML version 1.0, an evolution of DRPL 2.0. In 2002 ContentGuard released XrML version 2.0, a radical departure from all of the preceding versions. XrML 2.0 included an abstract rights language with very few core elements. The rights elements from previous versions were carried forward in this new version via an extension called the Content Extension. Finally, In 2003, the Motion Picture Experts Group (MPEG) released MPEG-21 Part 5, Rights Expression Language (ISO/IEC 21000-5), a substantially modified version of XrML 2.0 in which the Content Extension is removed, and a MultiMedia Extension appears in its place (“Information technology — multimedia framework (MPEG-21)”, 2001).

A similar evolution took place with respect to ODRL. Iannella introduced ODRL in 2000 out of concern for the closed approaches that were being used for DRM (Iannella, 2000) XML-based ODRL version 0.5 was released at that time as a work in progress, with the goals of providing clear DRM principles focused on interoperability across multiple sectors and support for fair-use doctrines. In 2001, ODRL Version 1.0 was submitted to ISO/IEC MPEG in response to a call for a rights data dictionary (RDD)—REL. By that time, Nokia’s Mobile Rights Voucher and Real Networks’ Extensible Media Commerce Language had been merged into the language. Then, in 2002, an Open Mobile Alliance (OMA) REL, based on ODRL 1.1, was proposed and called the OMA DRM 1.0 Enabler Release. ODRL 1.1 was also submitted to the W3C regarding possible chartering of a DRM/Rights Language activity within the W3C. By 2004, OMA released the OMA DRM 2.0 Enabler Release working drafts that addressed expanded device capabilities, improved support for audio/video rendering, streaming content, and access to protected content using multiple devices.

XrML and ODRL are the major RELs that are trying to become standardized in the digital content management industry. Each of these RELs has formed alliances with the major players in the industry and the standards bodies. Products and services companies such as Microsoft, OverDrive, Zinio Systems, DMDsecure, Integrated Management Concepts, and Content Works,
have publicly announced to build products that are XrML compliant. Standards bodies such as the Motion Pictures Experts Group (MPEG) and the Organization for the Advancement of Structured Information Standards (OASIS) also endorse XrML. The International Digital Publishing Forum (IDPF), an international trade and standards organization for the digital publishing industry, has selected XrML as a basis for its rights grammar specification. The Organization for the Advancement of Structured Information Standards (OASIS), is a global consortium that drives development and convergence of open standards for global information society. OASIS has defined a token profile, which describes the syntax and processing rules for the use of licenses with the Web Services Security. It requires the processor of a license to conform to the required validation and processing rules defined in ISO/IEC 21000-5 REL, which is based on XrML (“Web services security rights expression languages token profile 1.1.”, 2006). XrML is a patented technology, and ContentGuard Inc. requires a patenting license for its usage. ODRL on the other hand supports open and free standards, and aims to participate in the standards groups to achieve a royalty-free rights specification mechanism. ODRL enjoys the backing of industry players such as, Real Networks, Nokia, IBM, Panasonic, and Adobe. OMA DRM 2.0 and RSA Security also endorse ODRL.

Both ODRL and XrML are XML-based general-purpose RELs. Since these RELs are general purpose, their design is meant to be flexible and expandable. A flexible REL ensures copyright statements of different DRM environments can be expressed. Whereas, an expandable REL can be added with new types of copyright statements that may be required by DRM applications in the future. There are, however, a number of other RELs that are designed to be used for a specific application, business model, or content format.

One such example is the Adobe Lifecycle Server, which manages information access with dynamic and persistent access control over documents created in Adobe’s Protoble Document Format (PDF) (“Adobe® LiveCycle® ES Rights Management”, 2006). The Adobe Lifecycle Server allows authors to create documents, and assign permissions that specify a recipient’s level of access over the documents. It also allows authors to revoke rights, change rights, and change expiry date, without having to reissue the documents after they are distributed. The Publishing Requirements for the Industry Standard Metadata (PRISM) defines the XML metadata for managing, describing and aggregating documents related to the publishing industry. These include magazines, books, journals, news, catalog, etc. PRISM has refrained from recommending the use of any general purpose REL, because of the lack of emergence of any one REL, such as XrML or ODRL, as the accepted standard (“The PRISM Rights Language Name Space.”, 2005). Therefore, the PRISM working group has created a small REL, called the PRISM Rights Language (PRL), that focuses on specifying a small set of elements that encodes the most common rights information (“The PRISM Rights Language Name Space.”, 2005). Apple, with its Fairplay technology, has accomplished one of the most economically successful implementations of DRM, in the video and music industry. Apple does not make use of any commercially available general purpose REL, and instead, uses its own proprietary solution. Little information is available on how the Fairplay technology expresses and manages the usage rights of users, most of which are implicit in the system implementation.

A REL expresses statements in a copyright agreement in a machine-readable form. Most of the RELs used in the industry are based on XML. XML is a general-purpose specification for creation of specific markup languages. In an XML based REL, different terms occurring in copyright statements are represented as XML elements in a Type-Value form. The Type, which represents the semantics of different terms in copyright statements, is called metadata. Copy-
Rights Expression Languages

Figure 1. A sample XrML license. ("XrML 2.0 Technical Overview", 2002)

```xml
<license>
  <grant>
    <keyHolder>
      <info>
        <dsig:KeyValue>
          <dsig:RSAKeyValue>
            <dsig:Modulus>kigHkijhkjgG2HsdlhkljHldlf<dsig:Modulus>
            <dsig:Exponent>HJKH423HJ</dsig:Exponent>
          </dsig:RSAKeyValue>
        </dsig:KeyValue>
      </info>
    </keyHolder>
  </grant>
</license>
```

Right statements are then expressed by storing XML elements in a tree-like structure as shown in Figure 1. For DRM to work in an open, distributed and interoperable environment, it is essential that metadata is unambiguously understood by different computers enforcing rights in a given license, and semantics of the metadata are in accordance with the meaning intended by rights issuers. To achieve this goal, it is necessary to develop a common metadata dictionary of terms involved in copyright statements, called Rights Data Dictionary (RDD). The <indecs>RDD is a consortium based initiative that aims to develop such a RDD ("<indecs>rdd White Paper.", 2002). The MPEG-21 Part 6 Standard for Rights Data Dictionary has selected <indecs>RDD as its baseline ("<indecs>rdd White Paper.", 2002). The dictionary aims to standardize the terms involved in rights expressions that can be unambiguously expressed, and applied across different domains where rights need to expressed and enforced.

Semantics of XML-based languages are informal. Hence, it is not possible to reason about rights expressed in XML-based RELs. To answer questions such as, “Are rights statements A and B semantically equivalent?” “Is it possible to express copyright statement x, in language A?”, “What is the set of all action sequences that are allowed by rights statement x?”, and so on, it is necessary to follow a more formal approach for rights expression.

To address this issue, academicians have tried to design RELs using more formal approaches such as logic, set theory, algebraic methods, just to name a few (Arnab and Hutchison, 2007; Chong et al., 2003; Gunter, Weeks and Wright, 2001; Hilty, Pretschner, Basin, Schaefer, and Walter, 2007; Holzer, Katzenbeisser, and Schallhart, 2004; Pucella and Weissman, 2002). The use of these methods not only provides a machine-readable and actionable REL, but also allows using the well-established results in these theories to reason about the resulting RELs. Such a level of formal reasoning is necessary in order to answer the kind of questions mentioned above. Later, we provide a detailed discussion on these approaches, along with the trade-offs involved in the process.

Since RELs are the communication languages of DRM systems, their design, expressive power, limitations, expression format, and other such attributes influence the structure and properties of DRM systems in a significant manner. As
Rights Expression Languages

mentioned previously, numerous RELs have been developed in industry and academia. While the industry has taken a practical approach of using the less formal, albeit machine-readable, XML format for RELs, academicians have taken a purist approach by using a logical basis for developing RELs. The use of RELs in the industry is still fragmented and none of the general purpose RELs, such as ODRL and XrML, is used extensively in the market (Jamkhedkar, Heileman and Martinez-Ortiz, 2006). Additionally, general-purpose RELs are quite complex and bulky, which makes it extremely difficult to formalize them (Halpern and Weissman, 2004, 2008). Most of the closed DRM environments, on the other hand, use custom-made RELs that suit their application. There are a number of steps involved in transforming rights requirements into a REL and the subsequent use of REL in DRM systems. In the next section, we provide an outline of the process and design decisions in the development of RELs.

The Role of REL in DRM Systems

DRM Systems aim to manage rights associated with digital content. Content rendering environments (or machines in those environments) ensure that content is used in accordance with the rights associated with it. For DRM systems to work effectively, it is necessary for content creators and distributors to be able to express the intended rights associated with digital content in a machine-readable form, and digital content rendering environments must be able to interpret and enforce these rights. RELs are used for this purpose in DRM systems. This section provides an overview of the stages that are involved in the development of RELs for DRM systems. The development of a REL involves many steps that transform copyright requirements to a formal, machine-readable, and actionable language that operates within a complete DRM system.

Figure 2 provides an overview of the different stages of development of an REL. The first step is to gather the set of requirements that capture the different types of rights expression statements that are necessary in DRM systems. The set of requirements must reflect the copyright model that need to be supported by the DRM system, which will use the REL. There are many opinions regarding the types of copyright statements that need to be supported by an REL (“MPEG-21 Requirements for a RDD and an REL”, 2001; “ODRL Version 2 Requirements”, 2005; Parrott, 2001). It is easy to define a small set of such copyright statements for RELs that are developed for specific DRM applications, and operate within a well-defined business model. However, agreeing on such a small set for a general-purpose REL is much more difficult. A general purpose REL is expected to be flexible enough to express various types of rights expression scenarios that may be necessary in different applications. The most general requirement RELs is the ability to express rights and obligations for a given agent over a digital object under different conditions.

Once the requirements are agreed upon, the next step is to develop a conceptual model, as proposed by McCarty (1989), which comprises of a small set of common categories that are abstracted out of the requirements. For a general-purpose REL, the elements of this set need to be abstract enough to ensure that the resulting REL has enough flexibility to be used in different environments. Some of the common categories may include rights, obligations, agents, actions, digital objects, time and space, to name a few. The next step is to define relationships among the different entities of this set. Figure 3 shows one way to define the relationship among the set of entities defined by Subject, Object, Operation and Constraints (Chong et al., 2003) A conceptual model defines the scope and boundaries of the resulting REL to a great extent. Even though XrML and ODRL are XML-based RELs, they differ in their scope, and hence their expression capabilities, because they are based on different conceptual models.
A conceptual model captures entities in the application domain along with the relationship among these entities. Once such a model is in place, a logic (or calculus) is defined over the model that allows expressing copyright statements in terms of the entities defined in the model. This is the phase that involves the design of REL grammar and enforcement algorithm. The properties that the REL logic needs to satisfy determine the degree of formalism and expressive power of the resulting REL. The Most essential properties that need to be satisfied by all REL logics are a) the syntax must be machine readable (or map to a machine readable syntax), and b) the logic
must be actionable. In other words, given a series of user actions, there must exist an algorithm to determine if the sequence is valid or invalid with respect to a given rights statement. While these two are the most essential properties, other properties such as soundness, completeness, and computational complexity of the logic are the other factors that play a role in the design of logics underlying RELs.

Even though REL logic, by itself, is sufficient for rights expression, interpretation, and enforcement, it is necessary to add “syntactic sugar” to the logic to make it suitable for human comprehension. While this step may seem rudimentary, it is needed to enable the rights owner to specify copyright easily, and for the user to understand the rights and obligations associated with a digital object.

RELs operate in complex DRM systems, where they interact with various other components. In addition to that, RELs form a central component of any DRM system, and other components are heavily influenced by the scope and design of RELs. Trust management, license management, rights negotiation, authorized domains, superdistribution, and security, are some of the important aspects of DRM system whose design and implementation are dependent on the design of RELs. In certain DRM systems, these components are a part of REL itself, whereas, in others, they are implemented independent of the REL. Apart from this, many of the desirable properties of DRM systems, such as, interoperability, standardization, and protocols are also critically dependent on the design of RELs. RELs influence the design of DRM systems to a great extent. Therefore, while designing an REL, it is necessary to take these factors into consideration.

Many RELs have not followed all the steps mentioned above in their process of development. For example, even though XrML and ODRL have rigorously considered most of the requirements for
Rights Expression Languages

rights expressions, they have paid little attention to the development of a solid logical basis for these languages. This has forced many researchers to reverse engineer and construct a logic based calculus for a significant portion of these languages (Halpern and Weissman, 2004, 2008; Holzer, Katzenbeisser, and Schallhart, 2004; Pucella and Weissman, 2006). There are also some greenfield attempts to construct REL logics. The focus of these attempts however has been quite narrow, and they fail to explain a sound conceptual model which captures the requirements, or discuss the boundaries of RELs, and system properties that an REL need to satisfy in order to be used in a DRM system.

All the steps mentioned above, aim to capture different aspects of RELs. Over emphasis on any particular step will influence the results in the others. For example, on the one hand, it is very difficult to create a formal logic for a highly complicated conceptual model, and on the other hand, a over simplified model is unable to capture all the requirements of rights expression. Certain RELs many not even require to undergo all the processes in order to achieve satisfactory results. In order to balance these trade-offs, it is necessary to understand the properties, requirements, goals, and trade-offs with respect to each of the steps shown in Figure 2. In the rest of the paper we will discuss each of these steps in detail, along with supporting examples, starting with the requirements and conceptual model in the next section.

Conceptual Model

A conceptual model over which an REL is based aims to capture the entities in the copyright domain and defines relationships among these entities. Since the model is created on the basis of the copyright domain, it is first necessary to define what constitutes a copyright domain. A copyright domain consists of copyright agreements over intellectual property (IP) between an IP owner and an IP user. In traditional environments, these statements are written in natural languages. It is not possible to create a formal conceptual model based on a domain that is not well defined. It is therefore necessary to first define the scope of copyright domain over which the model will be developed. In this section, we discuss the principles over which the boundaries of this domain are determined, the trade-offs involved deciding the features included in this set, and finally the factors that are affected by the structure of the conceptual model.

The scope of the copyright domain is defined by specifying what concepts lie within the domain, and what concepts lie strictly outside the domain. Defining the scope has been a contentious subject, with a number of different opinions provided. The scope depends on a number of factors such as, type of digital objects, application environment, business model, etc. If these constraints are known, it is easier to define the scope of the domain. For example, the Open eBook Forum (OeBF) has detailed description about the requirements for rights grammar for expressing copyright over eBooks (Barlas et al., 2003). However, the task is much more difficult for general purpose RELs that are supposed to be used in all types of applications. One of the most comprehensive literatures on requirements for general purpose REL is provided in response to the call by proposals by MPEG for requirements for Rights Data Dictionary (RDD) and REL (Parrott, 2001). Requirement documents contain in detail the features that need to be supported by RELs such as, types of constraints, usage rights, obligation specification, pricing models, superdistribution, trust management, security protocols, and so on. There have also been suggestions of RELs having capabilities of carrying out rights negotiations (Arnab and Hutchison, 2005a, 2005b).

There are two dimensions to the problem deciding the scope of the copyright domain. First, what are the factors that decide whether a given feature must be included in the copyright domain or not?
Second, once the scope is defined, how does it affect the other processes in REL development such as formalization and the integration of the language within a DRM system?

The scope defines what features must be included within a given REL, and what features must be left out. To address the first problem, it is necessary to understand what happens to the features that are left out of a given REL. The features that are within an REL are expressed as rights statements, and the ones that are left out are implemented as separate protocols, or they are hard-coded within a DRM system. For example, XrML provides the facility to express security features, such as the encryption algorithm used to encrypt the content, within the language. If this feature is left out in a given REL, then it can be implemented in a DRM system as a separate protocol independent of that REL. Another example is that of authorized domains, which represent a set of devices over which a user can legally render content. This feature can be included within the language, to allow specification of devices in an authorized domain. If the feature is left out of an REL, then it can be hard coded in the system, in a manner similar to the one used by Apple’s Fairplay technology, which restricts the use of a purchased song to a set of user registered devices.

Including a feature within the language offers a number of advantages. First of all it allows a more fine-grained expression of copyright statements. Any feature that is left out of the language has to be expressed over complete licenses, and cannot be expressed over parts of copyright statements in a given license. For example, if expression of authorized domain is kept out of REL, then it is not possible to specify that 2 different rights in a given license are valid over different sets of device domains. To implement a particular feature, in a DRM system, that is not a part of REL, requires development of a separate set of protocols, and mechanisms to link those features back to licenses. For example, suppose that payment obligations are not a part of RELs. In such a case, it is necessary to develop protocols to convey the payment obligation information for each license separately to the user. In addition to that, separate mechanism is necessary to track user payment actions and provide them as an input to the rights enforcement mechanisms, along with the information specifying what user actions (i.e. payments made by the user) relate to what licenses. It is much easier to make payment obligations a part of REL, where user actions can be tracked and managed by a single rights enforcement mechanism.

If a feature is included in an REL, it creates dependency of that feature on the rights enforcement mechanism, leading to lack of choice and inflexibility in the system. If any changes are desired in that feature, it requires changes to be made in the language. For example, assume that the content encryption-key exchange protocol in included in the REL. In such a situation, if a different key exchange protocol is desired, it can be implemented only by bringing about changes in the language. Such a dependency on RELs is not favorable, as it leads to rigid solutions that are incapable of rapid evolution.

The second problem is deciding how the scope of the domain affects the remaining steps of REL development. The more features an REL has, the more expressive it is, and hence can be used in different applications and business models. These qualities are desirable in general purpose RELs, which explains why languages such as ORDL and XrML have so many features. Another advantage of having a lot of features within the language makes DRM system development much easier, without having to worry about the overhead of managing the features separately, and linking them to the rights enforcement mechanisms.

However, a language having a large scope is not desirable either. Bulky languages lead to rigid, monolithic systems. In other words, the system depends too much on the language. Any changes in the system require corresponding changes in the language, as explained earlier. Another obstacle in having a language with large scope including
many features is the difficulty in development of logic for a conceptual model based on such a large domain. XrML and ODRL are the general-purpose languages that have a large scope that includes features beyond rights expression. Researchers agree that formalization of all the parts of these languages is impractical and too difficult to achieve (Halpern and Weissman, 2004, 2008). Formalization and easy integration with the system are necessary REL features for DRM interoperability, as shall be shown in the following sections. Hence, on the one hand, a language with a small scope is unable to express many of the desired features of copyright statements, while on the other hand, a bulky language is difficult to formalize and manage within the system. It is therefore necessary to carefully determine the scope of the copyright domain, on which an REL is based, taking into consideration all these factors.

Once the scope is decided, most common categories within the domain must be identified to the desired level of abstraction. For example, in case of ODRL, some of the common categories identified are as follows: rights, assets, party, permission, duty, prohibition, constraints, etc. There are trade-offs involved in the level of abstraction involved in identifying these categories. A high level of abstraction with very few categories will make the language simple, but impractical. On the other hand, a low level of abstraction with too many categories will make the language expressive, but too complex to model.

Once the categories are identified, the next step is to define relationships among these categories. The type of relational defined among the entities determines whether the resulting language can express certain scenarios or not. The structure of the conceptual model determines the ability to capture and express aspects of copyright statements such as hierarchical relationships, sequencing of rights and obligations, inheritance semantics, matching of rights to users on the one side and digital objects on the other, matching of constraints to rights, and so on. For example, structure of the conceptual model determines whether it is possible to express the following rights statement:

If a set of users, A enjoy rights $x_1$, ..., $x_n$ over a digital object $K$, then any user a belonging to A enjoys rights $x_1$, ..., $x_n$ over any part of K.

It is not possible to express this statement in an REL based on a model whose structure does not support hierarchical relationships. As mentioned earlier, the more complex the structure of a model, the harder it is to create an REL logic based on that model. RELs with XML syntax, which are less formal, can afford to have a large domain scope, and a complex structure for their conceptual model. On the other hand, logic-based RELs, which are more formal, avoid these complex relationships to keep the structure simple and easy to formalize. Figure 3 shows the complex conceptual model for ODRL, which is a XML-based REL. On the other hand, LicenseScript, a logic-based REL, has a much simpler conceptual model, as shown in Figure 4.

It is desirable to have a large scope and a complex structure for the conceptual model on which the REL is based upon. It allows including various concepts and offering versatility in REL expression capabilities. These qualities are certainly beneficial for a general purpose REL, which is used in different applications and business models. The only impediment in this approach is posed by formalization, and managing system properties, such as flexibility and interoperability. It is therefore necessary to understand the importance of these factors and the problems that might arise if they are not taken into consideration. The next section explains the importance of formalization and various approaches taken to achieve it, along with the trade-offs involved.
FORMALIZATION OF RELs

A conceptual model reflects the structure of the relationship among the entities that exist in a copyright domain. Such a conceptual model forms the basis upon which a rights expression language is designed to express copyright statements. Natural languages provide enough flexibility to easily express most of the copyright terms, and they are also comprehensible to the human mind. These languages, however, are not suitable for rights expression in DRM systems, where rights are managed not by humans, but by machines. Apart from power of expression, RELs need to satisfy a number of other requirements that necessitate varying degrees of “formalness”. In this section, we discuss these requirements starting with the ones that are absolutely necessary to the ones that can be compromised. We will further discuss how these requirements affects the other features of REL design.

Two of the most important features of a language are syntax and semantics. Syntax decides the rules for generating the set of valid strings (i.e. grammatically correct copyright statements in case of RELs) in the language. A language is decidable if it can be determined whether or not a particular string belongs to the set of valid strings in the language. Semantics refer to the meaning behind valid sentences in a language. An REL is machine-readable if it is decidable, and a machine can interpret its semantics. Machine readability is the central requirement for RELs used in DRM systems. An REL is actionable, if it is machine readable and there exists an algorithm to determine if a sequence of actions is valid or invalid with respect to a given rights statement expressed in the REL.

Natural languages do not have a formal syntax, which rules them out as a choice for RELs. In other words, it is extremely difficult for a machine to determine if a rights statement expressed in a
natural language is syntactically correct or not. Other than the syntax, the semantics of natural language can be ambiguous in certain cases, which is also a reason why they are not used for rights expression. Hence, formal approaches are taken to for rights expression, such as XML-based or logic-based, instead of using natural languages.

**XML-Based RELs**

The power of expression and flexibility offered by natural languages is still a major incentive to use them for rights expression. One of the ways to overcome the obstacle of informal syntax of natural languages is to restrict their expressive power, capture the most common structure of rights expression statements, and formalize the syntax. Steps 1 and 2 can be achieved by developing a conceptual model, and restricting the rights expression statements to be expressed only in terms of the categories used in the model. Once this is done, the structure of the conceptual model and that of the rights statements is captured in a formal syntax.

One of the ways to capture the structure of the rights statements in a formal syntax is to present it in a markup language whose structure reflects the structure of the conceptual model. A descriptive markup can capture the logical structure of rights statements (or a license). To create a descriptive markup, elements of rights statements are identified and tagged accordingly. For example, consider the rights statement:

*Alice has the right to play the song “Hotel California”.*

To markup this sentence, “Alice” is tagged as the USER, “play” is tagged as the RIGHT, and “Hotel California” is tagged as the DIGITAL OBJECT. Such a tagging method allows the rights enforcement mechanism to separate and identify various elements, along with their relationship with each other. eXtensible Markup Language (XML), allows user defined tagged elements to be represented in an unranked tree structure. XML has a formal syntax and languages based on XML are decidable (Berstel and Boasson, 2002). XrML and ODRL are two of the prominent XML-based RELs. Figure 1, shows a sample XrML license.

It is important to understand what are not the capabilities of markup languages. Markup languages are not a computational automaton, a data model, or a mathematical formalism (Raymond, Tompa, and Wood, 1995). Markup languages simply allow associating different elements with their respective tags, and defining relationships among these elements in terms of an unranked tree structure. Markup languages do not define the semantics underlying the data they present. The semantics of XML-based RELs such as XrML, are hence defined informally by means of an accompanying documentation that describes the meaning behind the tags. XrML documentation that provides prose explanation of the meaning of the tags in XrML is informal, and may lead to ambiguity. The Authorization Algorithm implicitly captures the formal aspects of the semantics of XrML, which is used to enforce rights. The same argument applied to ODRL.

Before discussing formal semantics, it is first necessary to define the requirements of formal semantics in terms of RELs. This is necessary, because, the nature of these requirements will decide the degree of formalism expected in the REL calculus. The most basic requirement in terms of formal semantics is: *Unambiguous interpretation of rights statements irrespective of the external conditions.* In other words, the interpretation of a rights statement is always the same, and is not influenced by, who interprets it, in what environment it is interpreted, or at what point of time it is interpreted. This means, under similar external conditions, the set of actions permitted by a given rights statement always remains the same.

As mentioned earlier, XML-based RELs do not have formal semantics. In order to have order to achieve that, one approach is to construct a
layer of semantics, based on some mathematical formalism, over these languages. The main constructs available in XML are: “containment” (hierarchy), “adjacency” (A ‘followed by’ B), “co-occurrence” (if A then [also/not] B), “attribute”, and “opaque reference”. These constructs allow limited expressive power for modeling rights statements, however, its extensible tree structure allows tremendous flexibility. The mathematical formalism that is used to provide formal semantics to XML-based RELs must be able to capture these constructs. Flexibility provided by XML is exploited by these RELs, which has resulted in them having a large scope. Such a large scope makes it impossible to provide formal semantics to all parts of the language. As a result, researchers have been successful in providing formal semantics to only parts of XrML and ODRL. The other approach is to use mathematical logic, which provides in-built semantics to express many complex concepts and reason about them.

Logic-Based RELs

Developing a formal calculus for rights expression, interpretation, and enforcement can eliminate ambiguous syntax and semantics. Grammar of a language associates a structure for each of the strings in the language. This structure is then used to determine the semantics of those strings. A grammar that associates more than one structure for a particular string is said to be ambiguous. Since the string is associated with more than one structure, the semantics of the string also become ambiguous. Therefore, in order to avoid ambiguous semantics, it is necessary for the REL to have an unambiguous syntax.

The inherent semantic constructs available in XML are limited, and the remaining interpretation either depends on the intuition of the reader or is implicitly captured by the enforcement algorithm. Mathematical logic, on the other hand, provides tools to create languages where a significant number of deductions can be made from the structure of the strings in the language itself. Different types of mathematical logics are available such as Propositional logic, first-order logic, modal logic, etc. Each of these logics can express different types of concepts along with precise semantics to capture these concepts. For example, Propositional logic allows to express relationship among sentences using sentential connectives, first order logic, in addition to the capabilities of Propositional logic, allows to express the fine grained structure that exists within each of the sentences, and modal logic allows to model action, temporal, spatial, deontic, and other such concepts. Since these concepts have well defined semantics in these logics, their interpretation across different systems is precise and consistent. An REL calculus must therefore aim to model the language in such a way that most of the concepts in the conceptual model are defined by means of these logics, with minimal dependence on intuition and enforcement algorithm for capturing the semantics. Once these concepts are captured in these logics, it is possible to reason about other properties of the rights statements expressed in the calculus.

The ultimate goal of a rights owner is to ensure that the user uses the content in accordance with the copyright agreement. For this purpose, rights statements in the copyright agreement are encoded in a machine-readable form by the use of an REL. It is therefore necessary to prove that the action sequences permitted conform to the ones specified in the copyright statements. Apart from this, it is also essential to be able to prove whether or not a particular rights statement can be expressed using a certain REL. While these requirements are sufficient for RELs, which are tailor-made and used in standalone DRM systems, more requirements are necessary in interoperable DRM systems. In interoperable DRM systems, it is necessary to check whether a given rights statement $A$ in one language is semantically equivalent to another rights statement $B$ in another REL. Also, it is also sometimes necessary to generate a
Rights Expression Languages

rights statement in one REL, which is semantically equivalent to another statement in some different REL. It is also essential to know whether a given statement is a subset of another statement in terms of action sequences permitted. To solve this problem it must be possible to generate all the valid action sequences for a given rights statement.

It is extremely difficult to address these issues in RELs where most of the semantics are informally described or implicitly captured by an authorization algorithm. Mathematical logic, on the other hand, has precise semantics, and well-established methods for verification, a process of determining the correctness of the system built. In terms of rights expression, these methods can be employed to determine the above-mentioned properties of rights statements, which are expressed in mathematical logic. There are two approaches for verification, one is proof-based, and the other is model-based. In a proof-based approach, the license (consisting of rights statements), is expressed as a set of formulas $\Gamma$. The property to be verified is expressed as another formula $\phi$. Formal proof techniques are used to determine if it is possible to prove that $\phi$ follows from $\Gamma$ (i.e., $\Gamma \vdash \phi$). In a model-based approach, the license is represented by a model $M$ and the property is represented by a formula $\phi$. Verification method consists of determining if the formula $\phi$ is satisfied by the model $M$ (i.e., $M \models \phi$).

An REL based on logic must try to capture most of the categories involved in the conceptual model. The ability to do so depends on the ability of the underlying logic to capture different semantics.

The most basic of all the types of logics is the Propositional logic. Propositional logic is a calculus for reasoning about formulas created out of simple propositions connected by sentential connectives (e.g., and, or, implication, negation, etc.) The grammar is defined using the standard Backaus Normal Form (BNF). Propositional logic allows to reason about compositions of different propositions, but fails to capture the structure of the propositions themselves. For example:

If Alice is the user and Alice has made the payment, then Alice can play the song.

Let proposition $x$ state that Alice is the user, proposition $y$ state that Alice has made the payments, and proposition $z$ state that Alice can play the song. In this case, Propositional logic can infer that for $z$ to be true, both $x$ and $y$ needs to be true (i.e., ($x \land y$) $\rightarrow z$). However the semantics that Alice is a user and Alice has made payments cannot be captured by Propositional logic. Hence for an REL based purely on Propositional logic, these semantics need to be conveyed to the enforcement engine by some mechanism that operates outside the REL.

These semantics can be captured within the calculus by Predicate logic, which builds upon Propositional logic. Predicate logic consists of propositions on objects of a universe. It consists of predicates that allow defining $n$-ary relations over the over the objects in the Universal set. Since the predicates are defined using variables, it is necessary to bind those variables to have definite truth-values for those predicates. This provision is made possible by the existential and universal quantifiers. A combination of predicates along with quantifiers allows Predicate logic to capture the semantics of statement $x$ which says that Alice is a user. The expressive power of First Order Predicate logic is much greater than Propositional logic, and can capture the semantics of complex statements such as:

All members of group “grp” can play song “hotelCalifornia” on all the devices in the authorized domain “home”. Alice is a member of group “grp”. Device “dev” belongs to the domain “home”.

The above group of statements can be captured in Predicate logic, and it can be mathematically
proved that Alice can play song *hotelCalifornia* on device *dev*.

\[
\text{member(Alice, grp)}. \\
\text{belongs(dev, home)}. \\
\text{play(M, hotelCalifornia, D) :- member (D, grp), belongs (D, home)}. \\
\]

The First Order Predicate logic is powerful enough to capture the internal semantics sentences, and to model most of the concepts encountered in rights statements. It is possible to model concepts such as, time, actions, permissions, obligations, license states, etc., in First Order Predicate Logic. There are, however, other specialized logics, which provide inbuilt semantics to capture these concepts.

Validity of a license is, most of the times, never eternal. Actions permitted by a license depend on the state of the environment within which the license is applicable. Rights statements defined in a license are usually accompanied by a set of constraints for each right. The constraints can be defined in terms of abstract concepts such as time, space, numerical bounds, etc. Therefore, the right to exercise action on content changes, according to the changes in the state of the environment. For example, a license valid in the USA will be invalid when the user moves to a different country. Hence the change of state in the environment has been brought about by the user action of moving from one country to another. Many of the licenses are bound by the temporal constraints. A license may be valid, for instance, only during the weekends of a particular month, say January 2008. The rights in such a license change from being valid to invalid and back to valid during the month of January, as time progress, and will finally be invalid after the end of the month.

In Propositional and Predicate logic, the propositions are either true or false in a given model. Modal logic semantics, on the other hand, are able to capture different modes of truth. Semantics of Modal logic are based on Kripke structures, which are directed graphs, where nodes represent different possible worlds, and the edges represent the accessibility relations. The validity of a proposition is never static, but changes depending on the world in question. The state and state changes in Modal logic can be used to model the change in the environment within which rights are enforced. For example, Linear Temporal Logic (LTL), which is a special case of Modal logic, can be used to model time over which the validity of licenses with respect to user actions can be studied and reasoned about.

Rights expression statements are mainly composed of permissions and obligations of a user with respect to digital content. Permissions and obligations can be modeled in Deontic logic which is also a special case of Modal logic (Huth and Ryan, 2002, p. 318-319). Deontic logic provides inbuilt semantics for *Obligation*, *Permission*, *Prohibition*, and *non-Obligation*. If these categories in rights statements are modeled in Deontic logic, it is possible to reason about them. Dynamic logic is another type of Modal logic that analyses change that occurs as a result of performing actions on the states in the world (Goldblatt, 1987; Harel, 1979). From the perspective of rights expression, interpretation, and enforcement, Dynamic logic can be used as a formal tool to model the rules of content usage, user actions on the content, and change in the environment as a result of user actions.

One of the earliest attempts to formalize RELs was proposed by Gunter, Weeks and Wright (2001). Gunter’s model consists of sequence of events called realities that are checked against the language constructs, which are modeled as traces of permissible events. This mathematical model encodes simple licenses and includes payments and content rendering events. The idea is further expanded by Pucella and Weissman (2002), who propose a logic to reason about rights using trace-based semantics. The logic uses temporal operators to capture the temporal semantics such as “next time”, “all times in future”, “until”, etc.,
Rights Expression Languages

and permission and obligation operators that attempt to capture deontic semantics. For model checking purposes, the authors reduce the logic to LTL, and apply existing verification mechanisms developed for LTL. Using this language as the basis, the authors also propose formal semantics for substantial fragments of XrML and ODRL (Halpern and Weissman, 2004; Pucella and Weissman, 2006). Another such approach to formalize ODRL, is proposed by Holzer, Katzenbeisser, and Schallhart (2004), who suggest the use of finite automata to capture a sequence of actions a user is allowed to perform according to a specific permission stated in the license.

LicenseScript is another formal REL, where licenses are represented in a multiset (Chong et al., 2003) The clauses in a license are expressed as logic programs and the state associated with these licenses are maintained as a separate set of bindings. LicenseScript is constructed over a well-defined conceptual model for rights. Licenses themselves are multiset rules, which reside in rendering devices. The communication between the device and license (i.e. the rights enforcement mechanism) is done by means of multiset rewrite rules. LicenseScript allows expression of dynamically evolving licenses in complex environments, such as authorized domains.

Another approach to formalizing RELs has been to use and extend the well-developed concepts of access control mechanisms. Usage control is the extension of access control mechanisms that incorporates the semantics of access control, trust management, and DRM in a single framework. Park and Sandhu (2004), introduce the term “usage control”, which extends traditional access control, to include trust management and DRM. They propose a usage control model, called UCON_ABC, that is based on Authorizations, Obligations and Conditions. Hilty et al. (2007), propose the use of usage control model as the basis for REL specification. These authors propose the Obligation Specification Language (OSL), which is based on the Z specification language for obligations in policies for distributed usage control. They further define translations from OSL to a significant subset of ODRL, to provide a formal basis for the latter. Rights management is often termed as persistent access control. Arnab and Hutchison (2007) propose a language LiREL that uses set notion that builds upon access control mechanisms.

Other approaches in formalization of RELs involve the use of existing specification languages such as CafeOBJ. Xiang, Bjorner, and Futatsugi (2008) use CafeOBJ along with OTS (Observational Transition System) to express formal specification of RELs. OTSs are used to model transition systems in terms of equations, and CafeOBJ can be used to model abstract machines. OTS/CafeOBJ, thus provides a common framework for specifying static and dynamic properties of licenses, and existing theorem proving facilities provided by this framework is used to further reason about properties of licenses.

Role of REL in DRM Systems

RELs do not operate by themselves. An REL is a part of a complete DRM system that uses the REL for the purpose of expressing rights from a rights owner to a content user. It is the language of communication for DRM systems. The properties of an REL determine to a large extent the system properties of the DRM system that is built around it. These properties include the scope and boundaries, the structure of the conceptual model, and the format of rights expression. The design of DRM systems may include incorporation of trust management, license management, rights negotiations, superdistribution, and security. When content needs to be managed across different systems then issues such as interoperability and standardization also need to be addressed. In this section, we will analyze how the properties of REL impact these design issues concerning the DRM system that is built around it.
There is not a clear consensus in the research community over the boundaries and scope of RELs. Since REL is the communication language of DRM systems, one extreme view is to incorporate most features within the REL. XrML takes this approach, and provides features to manage trust and express security parameters. One of the reasons this approach is taken by general purpose RELs is to provide a all purpose, ready to use solution to the DRM system designers. Such an approach, however leads to a compromise in formalization, system flexibility and interoperability (Jamkhedkar, Heileman, and Martinez-Ortiz, 2006).

The bigger the scope of an REL, the more difficult it is to formalize it. For example, researchers have argued that formalization of complete XrML is too difficult to achieve (Halpern Weissman, 2008). As mentioned earlier, different organizations have adopted different RELs for their systems. Interoperability among these systems is necessary for a smooth flow of content and consumer satisfaction. Interoperability among RELs is essential to achieve DRM interoperability. As discussed in the earlier section, formalization of RELs is a prerequisite to translate licenses from one REL to another.

Another problem with the expanding scope of RELs is the flexibility of the systems that utilize such an REL. A bulky REL will force the DRM system built around it to be more monolithic and less modular. Any changes made in the system built around such an REL will mandate changes in the REL itself. For example, if security protocol parameters, such as encryption key information, encryption format etc., are included in the language, it will restrict the DRM system to follow these protocols. Bulky RELs thus undermine the flexibility of DRM systems that are built around them. While this is acceptable for standalone, closed applications, such features are undesirable in systems that are supposed to interoperate with other systems.

The categories defined in the conceptual model along with the relationship among those categories influence the design of the DRM system utilizing the REL. Enforcement of rights expressed by a given REL is possible only if the DRM system utilizing the REL is able to identify and resolve the categories and relationships defined in the conceptual model. For example, an REL with the ability to define hierarchical semantics, is not useful if the corresponding system is unable to create such hierarchies, name them, and resolve these hierarchies and their constituent elements. Consider an REL with the ability to define rights over authorized domains. Such an REL is not useful in a system that unable to create such authorized domains, name them and resolve their constituent devices. Therefore, it is necessary that the capability of rights expression is rightly matched by the DRM system using the REL.

Thus, the decisions taken in the process of design of an REL, change the properties of the language that results from such a process. These properties, in turn, determine how the REL will be incorporated in the DRM system, by influencing the DRM system properties.

CONCLUSION

The process of the design and development of RELs involves transformation of rights requirements into a machine-readable language to express rights. The central goal of this process is to create a language that is expressive enough to best reflect the rights requirements. It is pointed out in the paper, however, that the decisions taken to achieve this goal involve a number of trade-offs that not only undermine other goals of REL, but also the system goals such as interoperability, and standardization. Because of these trade-offs it is not possible to create an REL, which simultaneously satisfies the goals of all the steps, which include expressive power, formalization, flexibility, etc. It will always be the case that over-
emphasizes on one of these properties will make it difficult for other to achieve. REL design should therefore be modularized in a manner so as to get the right balance of these properties that will eventually enable to achieve the ulterior goals of DRM systems.

REFERENCES


In Computer Security, pages 531–546, Dresden, Germany.


Rights Expression Languages


KEY TERMS

Conceptual Model: An abstract model that captures the entities, and relationship among the entities present in copyright statements.

Digital Rights Management (DRM): Management of copyright over digital content and services.

eXtensible Markup Language (XML): A general-purpose specification language that allows creating custom languages by defining language specific markups (or tags).

Interoperability: Capability of one independent DRM system to interface with other independent DRM system, and allow seamless flow of content.

Rights Expression Languages (RELs): Formal, machine-readable languages used for expressing copyright agreements in DRM systems.

Semantics: Meaning associated with the symbols and valid expressions in a language.

Superdistribution: Distribution of digital content along a chain of users, in such a way, that each user plays the role of both a consumer and a distributor.

Syntax: Structure, or form of valid expressions in a language.

Trust Management: Management of trust, in terms of rights and capabilities, among the entities in a given DRM system.