Managing Digital Rights using Linear Logic

Adam Barth
Stanford University
abarth@cs.stanford.edu

John C. Mitchell
Stanford University
mitchell@cs.stanford.edu

Abstract

Digital music players protect songs by enforcing licenses that convey specific rights for individual songs or groups of songs. For licenses specified in industry, we show that deciding whether a license authorizes a sequence of actions is NP-complete, with a restricted version of the problem solvable efficiently using a reduction to maximum network flow. The authorization algorithm used in industry is online, deciding which rights to exercise as actions occur, but we show that all online algorithms are necessarily non-monotonic*: each allows actions under one license that it does not allow under a more flexible license. In one approach to achieving monotonicity, we exhibit the unique maximal set of licenses on which there exists a monotonic online algorithm. This set of well-behaved licenses induces an approximation algorithm by replacing each license with a well-behaved license. In a second approach, we consider allowing the player to revise its past decisions about which rights to exercise while still ensuring compliance with the license. We propose an efficient algorithm based on Linear Logic, with linear negation used to revise past decisions. We prove our algorithm monotonic, live, and sound with respect to the semantics of licenses.*

1 Introduction

Media players, such as iTunes and Windows Media Player, can impose restrictions on the use of media though Digital Rights Management (DRM). DRM systems typically separate content from licenses, employing a trusted DRM agent in media players to ensure that consumers do not exceed the digital rights granted by licenses. In this model, consumers can download encrypted songs from unsecured servers but are unable to play the songs without a license. A music provider can distribute promotional "play once" licenses to allow potential customers to hear songs and decide whether to purchase licenses for additional plays or for additional devices. Music providers can also sell time-limited licenses that allow unlimited plays during a fixed time period or sell subscription licenses that allow a fixed total number of plays the consumer can allocate among music from a large library.

In this paper, we explore some logical and algorithmic problems related to the use and management of licenses both concretely, in an language from industry, and abstractly, in a language based on Propositional Linear Logic. A license is a collection of individual rights, each of which is defined by one or more constraints. To authorize an action, a DRM agent exercises a right, perhaps consuming it completely or leaving behind a residual right. A DRM agent might hold several rights that authorize the same action. The choice of which of these rights to exercise when the consumer performs the action impacts what future actions are allowed under the license because rights maintain state. For example, if an agent holds a right to play either song a or b once, and the agent exercises that right to play song a, then the agent has foreclosed the ability to exercise that right to play song b. The agent can either prompt the consumer to make this choice, or it make the choice automatically.

Previous work on rights expression languages [4, 7, 1] treats the choice of which right to exercise exogenously, requiring the assignment of actions to rights as input to the authorization algorithm. We believe this provides a poor user experience, however, and that the choice should be made by the agent automatically. This choice is complicated because many languages, including XrML [2] (based on [9]) and ODRL [5] (a descendant of DigiBox [8]), support rights allowing the consumer to allocate a fixed total number of plays among several songs, for example allowing a consumer a fixed number of plays from a particular album. Although many of our results are general, we focus our discussion on version 2.0 of the Rights Expression Language [6] specified by the Open Mobile Alliance (OMA), a large industry consortium whose members include Cingular, Intel, Microsoft, and Nokia. In addition to providing for manual allocation, the OMA explicitly specify an algorithm for allocating actions to rights, but this algorithm is unsatisfactory because it leads to certain anomalies, as illustrated in the following example.

Example. Consider an online music scenario in which Alice receives some promotional rights to play several digital songs on a mobile music player. Alice visits the music web site and enters her promotional code. The site encrypts several songs (a, b, and c) and transfers them to her mobile music player, along with the right to play the songs a total of ten times. Alice plays song a twice, b four times, and c three times, for a total of nine plays. The DRM agent in her music player decrypts the songs, allows Alice to play the songs, and notes that she has one play remaining.

The following day, Alice receives another promotion. Pleased with her previous experience, she returns to the music web site and is offered the choice of two rights. The first is the right to play song d once before the end of the month. The second is the right to play either song a or song d once before the end of the month. She opts for the second right because she reasons that it is more flexible. The rights Alice now possesses are summarized below:

1. Play either song a, b, or c (acquired the first day).
2. Play either song a or d before the end of the month.

Alice decides to first play song a and then play song d. She reason this should be permitted by her rights because the first play should be authorized by her first right and the second play should be authorized by her second right. The DRM agent in her mobile music player, however, forbids her from playing song d because it exercised the second right to play song a, leaving Alice without the right to play song d. In fact, had Alice opted for the less flexible first option, the DRM agent would have allowed Alice to play those songs (as it would have been forced to assign the play of a to the first right). Alice is infuriated.

Monotonicity. The particular algorithm the OMA specifies for assigning actions to rights is non-monotonic in the sense that a sequence of actions allowed under one license might not be allowed under a more flexible license. In fact, we show that all algorithms that assign actions to rights *as the actions occur* are non-monotonic with respect to license flexibility. Each such online algorithm allows a sequence of actions under one right that it does not allow under a more flexible right because it must commit to exercising certain rights without knowledge of future actions.

We suggest that a trusted DRM agent need not assign actions to rights as the actions occur. The agent need only ensure that the complete sequence of actions performed does not exceed the license. After being requested to perform further actions, the agent is free to reassign actions to different rights because no one observes which rights are exercised for which actions. If the consumer wishes to perform an otherwise foreclosed action, the agent can revise its past commitments and authorize the action. With this added flexibility, the agent is able to perform monotonically.

Results. In this paper, we consider the static case, where consumers first acquire rights and then perform actions. We show that offline authorization for the OMA Rights Expression Language is NP-complete, but we exhibit a polynomial-time algorithm for a substantial fragment of the language by reducing the restricted problem to maximum network flow. We then show (under some technical conditions) that all online algorithms, including the algorithm specified by the OMA, fail to be monotonic.

To investigate monotonic algorithms, we interpret a conjunctive fragment of Propositional Linear Logic [3] as a rights expression language (extending the tractable OMA licenses). The logical formalism of syntax, semantics, and deduction facilitate a precise exposition of two approaches to achieving monotonicity. The first approach syntactically characterizes the unique maximal set of licenses on which there exists a monotonic online algorithm. These well-behaved licenses induce a monotonic "over-approximation" by approximating arbitrary licenses with "nearby" well-behaved licenses. The second approach relaxes the requirement that a DRM agent commit to exercising rights as actions occur. An agent using a rights expression language enriched with linear negation can use the negated terms to revise its past assignment of actions to rights. These more expressive rights admit an efficient authorization algorithm that is both sound and monotonic.

Without the contraction rule, Linear Logic easily expresses that a right can be exercised only a fixed number of times. The lack of the weakening rule seems less essential, but does ensure that agents do not drop rights indiscriminately. Although a more thorough study might conclude that another substructural logic is more appropriate, Linear Logic provides a convenient logical basis for investigating digital rights. In this treatment, licenses transform via linear implication to authorize actions. The operator & captures flexibility within a right and $\otimes$ captures the combination of rights into licenses.

The remainder of the paper is organized as follows. Section 2 describes the OMA Rights Expression Language and contains our complexity results. Section 3 demonstrates the non-monotonicity of online schemes for OMA licenses. Section 4 introduces Linear Logic semantics for digital rights. Section 5 concludes.

2 Offline Authorization for DRM Licenses

In this section, we consider the offline evaluation of Digital Rights Management licenses, specifically determining whether a sequence of actions complies with a license. We show that answering this question for the OMA Rights Expression Language is NP-complete and then exhibit a fragment of the OMA language in which authorization can be decided in polynomial time.

The computational complexity arises because a given action might be authorized by several rights within a license. As these rights have state, further authorizations depend on which right is exercised for the given action. Once actions are assigned to rights, evaluating a license is simple, but determining the assignment is computationally complex. We begin by describing the OMA Rights Expression Language.

2.1 OMA Rights Expression Language

In the OMA Rights Expression Language, a license is a forest of trees whose nodes are rights. Each right is defined by a list of constraints. A right authorizes an action if all of its constraints (and all the constraints of its ancestor rights) are satisfied. After allowing an action, an agent updates its state by transforming the constraints of the exercised right.

Actions. An action is defined by a number of parameters, including the principal performing the action, the kind of action (e.g., "play"), and the digital content on which the action is performed. Formally, an action is defined as a tuple (p, s, k, d, t, m) where

- p is the principal performing the action,
- s is the system performing the action,
- k is the kind of action,
- d is the date on which the action occurs,
- t is the duration of the action, and
- m is the digital content the action is performed on.

Rights. A right is defined by a list of constraints. These constraints can be stateful (e.g., "use this right at most five times") or temporal (e.g., "use this right only after January 1, 2006"). We consider nine kinds of constraints, listed below, that are satisfied by action (p, s, k, d, t, m) as follows:

- $\mathrm{Individual}(\hat{p})$ is satisfied if $p = \hat{p}$.
- $\mathrm{System}(\hat{s})$ is satisfied if $s = \hat{s}$.
- $\mathrm{Kind}(K)$ is satisfied if $k \in K$.
- $\mathrm{Date}(d_1, d_2)$ is satisfied if $d_1 \leq d \leq d + t \leq d_2$.
- $\mathrm{Content}(\hat{m})$ is satisfied if $m = \hat{m}$.
- $\mathrm{Count}(n)$ is satisfied if $n > 0$.
- $\mathrm{Interval}(\hat{t})$ is always satisfied.
- $\mathrm{Accumulated}(\hat{t})$ is satisfied if $t \leq \hat{t}$.
- $\mathrm{TimedCount}(n, \hat{t})$ is satisfied if $n > 0$.

Constraints are "subtractive:" including additional constraints reduces the circumstances in which a right can be exercised. For example, a right with the constraints $\mathrm{Count}(5)$ and $\mathrm{Count}(3)$ can be exercised only three times.[1]

Exercising a right can modify its state, for example reducing the number of plays remaining from five to four. This is modeled by transforming the $\mathrm{Count}(5)$ constraint into a $\mathrm{Count}(4)$ constraint. The stateful constraints of a right exercised to authorize action (p, s, k, d, t, m) are transformed as follows:

$$\mathrm{Count}(n) \rightsquigarrow \mathrm{Count}(n-1)$$
$$\mathrm{Interval}(\hat{t}) \rightsquigarrow \mathrm{Date}(d, d+\hat{t})$$
$$\mathrm{Accumulated}(\hat{t}) \rightsquigarrow \mathrm{Accumulated}(\hat{t}-t)$$
$$\mathrm{TimedCount}(n, \hat{t}) \rightsquigarrow \mathrm{TimedCount}(n-1, \hat{t})$$

The timed count transformation occurs only if $t > \hat{t}$. These transformations ensure, for example, that a right with constraint $\mathrm{Count}(n)$ is exercised at most n times and that a right with constraint $\mathrm{Interval}(\hat{t})$ is only exercised within t time units of when it is first exercised. The computational complexity of deciding whether a license authorizes a sequence of actions arises from selecting which rights to exercise for which actions.

Licenses. A license organizes rights as nodes in a forest of trees. The edges impose an "inheritance" relation on rights. A right can be exercised only if (1) it has no descendants, (2) all of its constraints are satisfied, and (3) the constraints of all of its ancestors are satisfied. When a right is exercised, its constraints, as well as the constraints of all of its ancestor rights in the tree, are transformed. Thus, state maintained by a right is effectively shared by its descendants. For example, consider the license with rights r_1 and r_2, descendants of right r, constrained as follows:

$$r : \mathrm{Count}(5)$$
$$r_1 : \mathrm{Kind}(\mathrm{Play}), \mathrm{Content}(a)$$
$$r_2 : \mathrm{Kind}(\mathrm{Play}), \mathrm{Content}(b)$$

This license authorizes a consumer to play both songs a and b, but only five times in total. The consumer can play song a three times and song b twice, or a twice and b three times, but not a three times and b three times. The OMA Rights Expression Language restricts inheritance to one level, requiring that child rights not in turn be parent rights.

[1] The OMA specification is ambiguous regarding whether an action that exhausts an accumulated constraint can be continued using another right. We assume an action (p, s, k, d, t, m) that exhausts a right with constraint $\mathrm{Accumulated}(\hat{t})$ can be decomposed into two actions $(p, s, k, d, \hat{t}, m)$ and $(p, s, k, d+\hat{t}, t-\hat{t}, m)$, provided $\hat{t} \leq t$.

2.2 Complexity of Authorization

In the OMA Rights Expression Language, deciding whether a license authorizes a sequence of actions is NP-complete. The difficulty stems from interval constraints (with inheritance) because it is hard to determine when to first exercise a right with an interval constraint. Exercising a right with an interval constraint transforms that constraint into a date constraint, authorizing some additional actions, but also preventing the right from authorizing other actions. This can be related to satisfiability of Boolean formulas, where assigning "true" or "false" to a variable satisfies some clauses but also precludes using that variable to satisfy some other clauses.

Theorem 1. *Deciding whether a license authorizes a sequence of actions is NP-complete. This is the case even for licenses containing only interval and content constraints.*

Proof idea. Given a SAT instance, we construct a license and a sequence of actions such that the license authorizes the actions if, and only if, the SAT instance is satisfiable. Each clause is represented by two actions, one occurring at time t_1 and another occurring at time t_2, say at times 10 and 20. Each variable is represented by a parent right with an $\mathrm{Interval}(1)$ constraint. Each occurrence of a variable in a clause is represented by a child right inheriting from the variable's interval right. If the variable occurs positively (negatively), the right is constrained, using a content constraint, to authorizing the clause's action at time 10 (time 20). Additionally, each clause is also represented by an $\mathrm{Interval}(1)$-constrained parent right with two children: one authorizing the clause's action at time 10 and the other authorizing the clause's action at time 20.

First, we show that if the constructed license authorizes the sequence of actions, the SAT instance is satisfiable. If a variable's interval right is first exercised before time 10, the variable is assigned "true." Otherwise, the variable is assigned "false." Of the actions representing a given clause, one must be authorized by a variable's interval right, and that variable satisfies the clause under this truth assignment.

Second, we show that if the SAT instance is satisfiable, the license authorizes the actions. If a variable is assigned "true" ("false"), exercise its interval right at time 10 (time 20). Of the actions representing a given clause, at least one must be authorized. Authorize the other one with the interval right representing the clause. Thus, the license authorizes the sequence of actions if, and only if, the SAT instance is satisfiable. □

While deciding whether an OMA license authorizes a sequence of actions is NP-complete in general, there exists a fragment of the OMA language for which authorization is decidable efficiently. A license is called *manageable* if it does not contain interval constraints and does not contain more than one kind of constraint among count, timed count, and accumulated (but can freely contain individual, system, kind, date, and content constraints). Authorization for manageable licenses can be decided in polynomial time. The class of manageable licenses contains many useful licenses, for example licenses allowing a limited number of promotional plays from various music collections and allowing unlimited plays of purchased music. For this class, authorization can be computed via a reduction to maximum network flow through the license graph, where each unit of flow represents an action.

Theorem 2. *It can be decided in polynomial time whether a manageable license authorizes a sequence of actions.*

Proof sketch. We sketch the proof for licenses containing count constraints. The argument is similar for licenses containing timed count or accumulated constraints. The reduction to MAX-FLOW proceeds by building a network flow graph. The actions in the sequence are represented by nodes in the graph, as are the rights in the license. The source is connected with a unit capacity edge to each of the action nodes. Each action node is connected with a unit capacity edge to a rights node if the action can be authorized by the right (ignoring count constraints). The nodes for each rights constrained with $\mathrm{Count}(n)$ are connected to their parents with a capacity n edge (those without parents are connected to the sink). If a rights node does not have a count constraint, it is connected with an infinite capacity edge.

If the maximum flow saturates the edges leaving the source, then the license authorizes the sequence of actions. Tracing the flow from an action node to the sink reveals which right authorizes that action. The edge capacities ensure the count constraints are satisfied. Conversely, if there is some satisfactory assignment of actions to rights, there is a flow that saturates the edges leaving the source. □

There is a gap between these two theorems. The complexity of deciding authorization for arbitrary licenses without interval constraints is left open.

3 Online Authorization of Actions

In the preceding section, we considered the offline problem of determining whether a license authorizes a sequence of actions. In practice, however, DRM agents are usually asked to compute such authorizations online, as consumers attempt to perform actions. The Open Mobile Alliance specifies a particular online algorithm for determining which rights to exercise, but that algorithm is problematic, as in fact are all online algorithms.

Consider a consumer Alice who possesses a certain license. She might reasonably believe that if she instead possessed a more flexible license, she would still be authorized

to perform every sequence of actions she can perform with the less flexible license. We call schemes that enjoy this property *monotonic* (defined precisely in Sect. 3.2). However, no online authorization scheme for OMA licenses is monotonic. Thus, Alice might "upgrade" to a more flexible license and lose the ability to perform a sequence of actions.

3.1 Automata Semantics for Licenses

In order to authorize an action, a DRM agent must exercise a right. Exercising a right might consume it, for example when it contains the constraint $\mathrm{Count}(1)$, or more generally simply modify its state. Exercising a stateful right can foreclose the authorization to perform some future action. In order to study how licenses evolve as actions are performed, we represent licenses as states in an (infinite) automata. When a consumer performs an action, the agent follows a transition labeled by the action in the automata, arriving at a new state. This new state represents the rights remaining after performing the action. In general there might be multiple transitions labeled with the same action, corresponding to exercising different rights.

The *authorization transition relation* for an action a (written $\xrightarrow{a}$) is a binary relation on $\mathcal{L}$, the set of licenses. If $\varphi \xrightarrow{a} \varphi'$, then license φ can transform into license φ' by authorizing a. For example, if φ is a license to play a certain song three times and a is the act of playing that song on a particular date, then φ' is a license to play that song twice. We call $\mathcal{L}$ together with the transition relations $\xrightarrow{a}$ the *license automata* $\mathfrak{A}$. The non-deterministic automata $\mathfrak{A}$ provides semantics for licenses:

Def. License φ *authorizes* a sequence of actions $a_1 \cdots a_n$ (written $a_1 \cdots a_n \in \mathfrak{A}[\![\varphi]\!]$) if there is a path starting at φ labeled by $a_1 \cdots a_n$ in the license automata $\mathfrak{A}$.

These semantics induce a natural "flexibility" partial ordering on licenses:

$$\varphi_1 \leq \varphi_2 \qquad \text{if } \mathfrak{A}[\![\varphi_1]\!] \subseteq \mathfrak{A}[\![\varphi_2]\!]$$

For example, a license that allows four plays of a song is less flexible than a license that allows five plays of that song.

OMA Licenses. For licenses φ and φ' in the OMA Rights Expression Language, $\varphi \xrightarrow{a} \varphi'$ if (1) action a is authorized by φ and (2) φ' is a license resulting from exercising some right of φ to authorize action a, as defined in Sect. 2.1. For the OMA licenses, only a finite portion of the automata is reachable from a given license. This finite subautomata can be used to model-check properties of that license.

The automata semantics of OMA licenses are *commutative*: if a sequence of actions is authorized, then every permutation of that sequence is also authorized. In the license automata, whenever $\varphi \xrightarrow{a} \varphi'$ it is the case that $\varphi' \leq \varphi$. We could have represented the semantics of licenses using multisets, but we chose sequences in anticipation of online authorization schemes that are not commutative.

3.2 Authorization Schemes

DRM agents employ authorization schemes to interpret licenses. Each scheme induces its own semantics on licenses, based on the sequences of actions allowed by an agent employing the scheme. In principle, authorization schemes could transform licenses arbitrarily, bearing no relation to the usual license automata.

Def. An *authorization scheme* A is an automata whose states are licenses and whose edges are labeled by actions.

As with the license automata, an authorization scheme allows a sequence of actions under a license if there is a path starting at the license labeled by the actions. We focus on four properties of authorization schemes:

Def. An authorization scheme A is

- *sound* if A is a subautomata of $\mathfrak{A}$,
- *online* if A is deterministic,
- *live* if $a \in \mathfrak{A}[\![\varphi]\!]$ implies $a \in \mathsf{A}[\![\varphi]\!]$,
- *monotonic* if $\varphi_1 \leq \varphi_2$ implies $\mathsf{A}[\![\varphi_1]\!] \subseteq \mathsf{A}[\![\varphi_2]\!]$,

for all licenses φ, φ_1, and φ_2 and all actions a.

A sound authorization scheme never allows a sequence of actions that is not authorized by the license automata.

A DRM agent employing an online authorization scheme decides which rights to exercise as actions occur, without knowledge of what future actions the consumer wishes to perform. A sound, online authorization scheme includes at most one transition from the full license automata for each license and each action. The included transitions indicate which right to exercise.

Some sound, online authorization schemes are degenerate, such as the empty automata, which does not allow any actions. Requiring schemes to be live removes this degeneracy. A live authorization scheme always allows an individual action if that action is authorized by the license automata. The sound, live, online authorization schemes can be characterized as subautomata:

Lemma 3. *An authorization scheme* A *is sound, live, and online if, and only if,* A *is a maximal deterministic subautomata of* $\mathfrak{A}$

In a monotonic authorization scheme, if a sequence of actions is allowed under a license, it is allowed under every more flexible license as well. The notion of flexibility here is semantic, but a weaker notion of monotonicity could be defined syntactically (for example, monotonic with respect to adding rights to a license). The authorization scheme defined in the OMA specification is sound, live, and online, but not monotonic. In fact, all sound, live, online schemes for OMA licenses fail to be monotonic even in a weak sense.

Theorem 4. *No sound, live, online authorization scheme for OMA licenses is monotonic.*

Proof. Given a sound, live, online authorization scheme A, consider three actions, a, b, and c, and the license φ:

$$\begin{aligned} r &: \mathrm{Count}(1) \\ r_1 &: \text{A descendant of } r \text{ that authorizes } a. \\ r_2 &: \text{A descendant of } r \text{ that authorizes } b. \\ s &: \mathrm{Count}(1) \\ s_1 &: \text{A descendant of } s \text{ that authorizes } a. \\ s_2 &: \text{A descendant of } s \text{ that authorizes } c. \end{aligned}$$

Because A is live and deterministic, there must be a unique license φ' such that $\varphi \xrightarrow{a}_{\mathsf{A}} \varphi'$. The scheme must have exercised either r_1 or s_1 (say r_1). Then, the count constraint of r in φ' is zero and $b \notin \mathfrak{A}[\![\varphi']\!]$. Thus, the sequence $a \cdot b \notin \mathsf{A}[\![\varphi]\!]$ (by soundness). Now consider the license θ consisting of r, r_2, s, and s_1. We have $\theta \leq \varphi$, but $a \cdot b \in \mathsf{A}[\![\theta]\!]$ (by liveness of A). Therefore, A is not monotonic. □

Notice every sound, live, online authorization scheme on OMA licenses fails to be monotonic even with respect to adding rights to a license. A DRM agent employing scheme A allows a sequence of actions under the license consisting of rights r, r_2, s, and s_1 that it would not allow if the rights r_1 and s_2 were added to the license.

4 Linear Semantics for Digital Rights

The semantics of digital rights can be understood using Linear Logic. Linear Logic eschews the usual weakening and contraction rules in favor of maintaining control over the multiplicity of syntactic terms, mirroring the control over the multiplicity of actions in DRM licenses. We interpret a conjunctive fragment of Propositional Linear Logic as a DRM Rights Expression Language, taking the regulated actions as propositions and building more complex licenses using the familiar Linear Logic operators. Authorization is captured through linear implication. If φ is the formula representing a license and $\varphi \multimap a \otimes \varphi'$, then φ authorizes action a with remaining rights φ'. The right to perform action a has been "separated" from the remaining rights φ'.

In this section, we use Linear Logic to investigate monotonic approximations and to describe a monotonic authorization scheme on licenses enriched with linear negation. Negated terms are used to specify the rights remaining after performing an action both exactly and compactly. These negated terms enable agents to revise their decisions about which rights to exercise for which actions, giving rise to a efficient authorization algorithm that is both sound and monotonic for these licenses.

4.1 Syntax, Semantics, and Deduction

Syntax. Licenses are built from a countable set of possible actions, denoted $\mathcal{A}$, which serve as the linear propositions. From these actions, licenses are built using the usual Linear Logic operators. We use formulas from the following grammar:

$$\psi ::= a \mid \psi \,\&\, \psi \qquad \varphi ::= \mathbf{1} \mid\, !a \mid \psi \mid \varphi \otimes \varphi,$$

where $a \in \mathcal{A}$. The symbol & represents an internal choice between authorizing one of two actions, ! represents unbounded replication, and $\otimes$ represents parallel combination of rights. Formulas from this grammar are the parallel combination of *atomic rights*, each of which is a choice between authorizing several individual actions.

In this grammar, the ! modality can be applied only to individual actions (and not their &-combination). This might appear overly restrictive as the license $!(a \,\&\, b)$ seems reasonable, but formulas $!\psi$ are equivalent to formulas in the grammar as $!(\psi_1 \,\&\, \psi_2) \multimap\!\circ\, !\psi_1 \otimes\, !\psi_2$. We also assume, without loss of expressiveness, that actions occurring in the scope of a ! do not occur elsewhere in the formula. Such formulas are in *normal form*. Formulas without & or ! are called *basic* licenses.

OMA licenses containing individual, system, kind, data, content, and count constraints are expressible as Linear Logic formulas from the above grammar. Accumulated and timed count constraints are excluded for simplicity, but interval constraints are excluded in order to maintain tractability. Individual, system, kind, date, and content constraints are simple to capture because they are stateless. Count constraints are expressible because atomic rights occurring outside the scope of a ! can be exercised only a limited number of times. Inheritance between rights is handled by the & operator, and the combination of rights into licenses is handled by the $\otimes$ operator. For example, the license used in the proof of Theorem 4 can be expressed as $(a \,\&\, b) \otimes (a \,\&\, c)$.

Semantics. With each formula φ, we associate a set of multisets of actions $\mathcal{S}[\![\varphi]\!]$, defined in Fig. 1. Performing all the actions in one of these multisets exhausts the license. Formally, a multiset is a function from $\mathcal{A}$ to $\mathbb{N}$. The multiset operator $\uplus$ is the usual disjoint union that adds the contents

$$
\begin{array}{rl}
s \in \mathcal{S}[\![\mathbf{1}]\!] & \text{if } s = \varnothing \\
s \in \mathcal{S}[\![a]\!] & \text{if } s = \{a\} \\
s \in \mathcal{S}[\![\varphi_1 \,\&\, \varphi_2]\!] & \text{if } s \in \mathcal{S}[\![\varphi_1]\!] \text{ or } s \in \mathcal{S}[\![\varphi_2]\!] \\
s_1 \uplus s_2 \in \mathcal{S}[\![\varphi_1 \otimes \varphi_2]\!] & \text{if } s_1 \in \mathcal{S}[\![\varphi_1]\!] \text{ and } s_2 \in \mathcal{S}[\![\varphi_2]\!] \\
s \in \mathcal{S}[\![!\varphi]\!] & \text{if } s = s_1 \uplus \cdots \uplus s_n, \text{ where} \\
 & s_i \in \mathcal{S}[\![\varphi]\!] \text{ for all } i \text{ from 1 to } n
\end{array}
$$

Figure 1. Multiset semantics for licenses

of its two operands. For each formula φ, the set $\mathcal{S}[\![\varphi]\!]$ contains exactly those multisets "promised" by the formula. If we take sets of multisets as models, we can provide a connection between semantic entailment and linear implication by defining every superset of $\mathcal{S}[\![\varphi]\!]$ to be a model of φ:

$$S \models \varphi \text{ if, and only if, } S \supseteq \mathcal{S}[\![\varphi]\!]$$

Each model of a formula contains every choice of multiset "promised" by the formula. Semantic entailment is as usual: $\varphi_1 \models \varphi_2$ if every model of φ_1 is a model of φ_2.

Deduction. Before defining the transition relation for the license automata, we define a leftist deductive system over formulas with the inference rules in Fig. 2 and a single axiom, $\varphi \vdash \varphi$. These rules are the usual left deduction rules of two-sided Classical Linear Logic for these connectives, with the exception of the $\mathbf{1}$-elimination rule. This rule is usually captured through right deduction rules. Note that none of these inference rules modify Δ. The deductive system is sound with respect to semantic implication.

Lemma 5 (Soundness)**.** *For all formulas φ and φ',*

$$\varphi \vdash \varphi' \textit{ implies } \varphi \models \varphi'.$$

This deductive system is not complete for our semantics, for example semantically $(a \& b) \otimes (a \& c) \models a \otimes (a \& b \& c)$ but not syntactically. If a complete deductive system were used, licenses could reach exponential length. The relation $\vdash$ is the "exercise" relation on licenses. As a formula moves across $\vdash$, non-determinism is resolved, corresponding to the DRM agent deciding which rights to exercise. Specifically, a right is *exercised* whenever a &-rule is applied to it. The sequent $\varphi \vdash a \otimes \varphi'$ indicates that action a can be authorized under license φ with remaining rights φ'. The authorization for a has been "separated" from the remaining rights φ'. This motivates our definition of the transition relation for the license automata $\mathfrak{A}$:

$$\varphi \xrightarrow{a} \varphi' \qquad \text{if, and only if,} \qquad \varphi \vdash a \otimes \varphi'$$

$$
\frac{\Gamma, \psi_1 \vdash \Delta}{\Gamma, \psi_1 \,\&\, \psi_2 \vdash \Delta} \qquad \frac{\Gamma, \psi_2 \vdash \Delta}{\Gamma, \psi_1 \,\&\, \psi_2 \vdash \Delta}
$$

$$
\frac{\Gamma, !\psi, !\psi \vdash \Delta}{\Gamma, !\psi \vdash \Delta} \qquad \frac{\Gamma, \psi \vdash \Delta}{\Gamma, !\psi \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta} \qquad \frac{\Gamma, \mathbf{1} \vdash \Delta}{\Gamma \vdash \Delta}
$$

$$
\frac{\Gamma, \varphi_1, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \otimes \varphi_2 \vdash \Delta} \qquad \frac{\Gamma, \varphi_1 \otimes \varphi_2 \vdash \Delta}{\Gamma, \varphi_1, \varphi_2 \vdash \Delta} \qquad \frac{\Gamma, \varphi_2, \varphi_1 \vdash \Delta}{\Gamma, \varphi_1, \varphi_2 \vdash \Delta}
$$

Figure 2. Rules for negation-free licenses

This transition relation yields a close connection between the automata semantics of a license formula, $\mathfrak{A}[\![\varphi]\!]$, and the multiset semantics of the formula, $\mathcal{S}[\![\varphi]\!]$.

Lemma 6. *A sequence $\sigma \in \mathfrak{A}[\![\varphi]\!]$ if, and only if, the multiset of actions occurring in σ is a subset of a multiset in $\mathcal{S}[\![\varphi]\!]$.*

4.2 Monotonic Approximations

Although sound, live, online authorization schemes for these linear licenses are not monotonic, there is a unique maximal set of licenses on which there exists a monotonic, sound, live, online scheme. In each of these well-behaved licenses, the rights authorizing a given action are linearly ordered by flexibility. The monotonic scheme on these licenses can be "lifted" to arbitrary licenses by approximating a given license with the closest well-behaved license. The resulting scheme is monotonic on all licenses but it is not sound: the approximation allows some sequences of actions that are not actually authorized by the license. The total number of actions allowed is the same, but the consumer is given more flexibility as to which actions to perform.

Before describing the set of well-behaved licenses, we define the *support* of a license: the support of a license φ, written $\mathrm{support}(\varphi)$, is the set of actions that occur as propositions in φ.

Def. Let $C_{\max}$ be the set of licenses φ such that, for all atomic rights ψ_1 and ψ_2 occurring in φ, if $\mathrm{support}(\psi_1) \cap \mathrm{support}(\psi_2)$ is non-empty, then

$$\mathrm{support}(\psi_1) \subseteq \mathrm{support}(\psi_2) \text{ or}$$
$$\mathrm{support}(\psi_2) \subseteq \mathrm{support}(\psi_1).$$

In a $C_{\max}$ license, if an action is authorized by several rights, then those rights are linearly ordered by flexibility. Exercising the least flexible right forecloses the fewest future authorizations. This leads to a natural online authorization scheme on $C_{\max}$, which we call the greedy scheme. Given a

license in $C_{\max}$ and an action a, the greedy scheme exercises the atomic right containing a whose support is inclusion-minimal (this is unique by the definition of $C_{\max}$). This scheme is monotonic, sound, live, and online on $C_{\max}$.

Theorem 7. *The set of licenses C_{max} is the unique inclusion-maximal set of licenses (containing the basic licenses) on which there exists a monotonic, sound, live, online authorization scheme.*

Proof. The greedy scheme is monotonic, sound, live, and online on $C_{\max}$. Suppose, by way of contradiction, there exists a set of licenses C such that (1) C contains the basic licenses, (2) C contains $\varphi \notin C_{\max}$, and (3) there exists a monotonic, sound, live, online authorization scheme A on C. Then, φ must contain two atomic rights ψ_1 and ψ_2 such that $\text{support}(\psi_1) \cap \text{support}(\psi_2)$ is non-empty and neither is a subset of the other. In particular, fix

$$a \in \text{support}(\psi_1) \cap \text{support}(\psi_2),$$
$$b \in \text{support}(\psi_1) - \text{support}(\psi_2), \text{ and}$$
$$c \in \text{support}(\psi_2) - \text{support}(\psi_1).$$

Starting with φ, repeatedly authorize a until A exercises either ψ_1 or ψ_2 (say ψ_1). Next, repeatedly authorize b until the remaining rights no longer authorize b. Let σ be the sequence of actions authorized thus far. Sequence $\sigma \cdot b \notin \mathsf{A}[\![\varphi]\!]$ because the rights remaining after σ do not authorize b. Now, let θ be the $\otimes$-combination (with repetition) of the actions in $\sigma \cdot b$. License $\theta \in C$ (because θ is a basic license) and $\theta \leq \varphi$ (because ψ_1 need not be exercised to authorize σ). However, $\sigma \cdot b \in \mathsf{A}[\![\theta]\!]$, and therefore A is not monotonic on C, a contradiction. □

The greedy scheme on $C_{\max}$ induces an approximation scheme on all licenses by way of an approximation structure. An approximation structure projects arbitrary licenses onto an identified set of licenses.

Def. An *approximation structure* is a pair (C, π), where

- C is a set of licenses (the set of *approximates*),
- π is a $\leq$-monotonic function from licenses to C, and
- $\pi(\varphi)$ is minimal such that $\varphi \leq \pi(\varphi)$, for all $\varphi \in \mathcal{L}$.

In an approximation structure (C, π), every license φ is approximated by the license $\pi(\varphi)$. The function π chooses one of the closest licenses in C to approximate each license. The approximation structure lifts an authorization scheme A on C to an authorization scheme A_C on all licenses: $\varphi \xrightarrow{a}_{\mathsf{A}_C} \varphi'$ if, and only if, $\pi(\varphi) \xrightarrow{a}_{\mathsf{A}} \varphi'$. Thus,

$$\mathsf{A}_C[\![\varphi]\!] = \mathsf{A}[\![\pi(\varphi)]\!].$$

The lifted scheme uses the approximate license to determine the sequences of actions allowed by a license. Two key properties lift with the authorization scheme.

Lemma 8. *For all approximation structures (C, π) and all authorizations schemes A,*

- *If A is monotonic on C, then A_C is monotonic.*
- *If A is live on C, then A_C is live.*

The more approximates, the *tighter* the approximation, the fewer "extra" sequences of actions allowed by the approximation. In the extreme, if $C = \mathcal{L}$, no extra action sequences are allowed. This all-inclusive C does not admit a monotonic, sound, live, online authorization scheme, but $C_{\max}$ does. In fact, approximation scheme induced by the greedy scheme is the tightest of its class because of the unique maximality of $C_{\max}$.

Corollary 9. *The greedy scheme is the tightest monotonic approximation scheme arising from sound, live, online authorization schemes on approximation structures.*

Even though this scheme is the tightest approximation of its class, it still allows many extra actions. Particularly problematic are licenses containing many atomic rights that overlap each other slightly. The approximation scheme does not allow a greater *total* number of actions, but it does allow each individual action to be performed a greater number of times. This suggests a "price for monotonicity" with these licenses. Expanding the expressiveness of licenses, however, enables monotonic online authorization schemes that do not allow any extra actions.

4.3 Linear Characterization of Remaining Rights

Online authorization schemes on OMA licenses fail to be monotonic because they are forced to commit to exercising rights as actions occur. Without the flexibility to later revise their past commitments, agents are unable to behave monotonically. A monotonic, sound, live, online scheme does exists, however, if the license automata $\mathfrak{A}$ is expanded to enable agents to revise commitments. The states in this enlarged automata $\mathfrak{B}$ record previously exercised rights and the transitions enable agents to "unexercise" these rights (while maintaining soundness). Both automata, $\mathfrak{A}$ and $\mathfrak{B}$, authorize exactly the same sequences of actions for the licenses they share in common.

The states of $\mathfrak{B}$ record exactly the future action sequences that can be performed. The description of these sequences must be compact in order to maintain efficiency. Simply listing the available continuations would require exponential space. To achieve an efficient representation, we expand the grammar of license formulas to include linearly negated actions, such as $a^{\perp}$. These negated terms represent "undoing" an action, or an authorization debt to be paid. New transitions are included in the automata that make use of these extra terms.

Linear Negation. Consider the license $(a \,\&\, b) \otimes (a \,\&\, c)$. In the $\mathfrak{A}$ automata, $(a \,\&\, b) \otimes (a \,\&\, c) \xrightarrow{a} a \,\&\, c$, foreclosing the possibility of authorizing b, but, in the $\mathfrak{B}$ automata,

$$(a \,\&\, b) \otimes (a \,\&\, c) \xrightarrow{a} (\mathbf{1} \,\&\, (a^{\perp} \otimes b)) \otimes (a \,\&\, c),$$

and the action b is still authorized:

$$(\mathbf{1} \,\&\, (a^{\perp} \otimes b)) \otimes (a \,\&\, c) \multimap b$$

Authorizing action b causes the $a^{\perp}$ term to eliminate the $a \,\&\, c$ term. This is because $a \,\&\, c$ is exercised to a, which is used in combination with $\mathbf{1} \,\&\, (a^{\perp} \otimes b)$ to reconstruct $a \,\&\, b$. In turn, $a \,\&\, b$ used to authorize b. In order to authorize b, the authorization debt $a^{\perp}$ must be paid. The generality of this approach is elucidated by the following valid formula:

$$a_1 \,\&\, \cdots \,\&\, a_n \multimap\!\circ\, a_1 \otimes (\mathbf{1} \,\&\, (a_1^{\perp} \otimes (a_2 \,\&\, \cdots \,\&\, a_n)))$$

Syntax. To express these states, we enrich our grammar for rights with linear negation:

$$\varphi ::= \cdots \mid \mathbf{1} \,\&\, (a^{\perp} \otimes \psi) \mid \cdots$$

Notice the negated terms can appear only in specific syntactic surroundings. When exercised, an atomic right produces two terms: the authorized action and a term containing that action negated. The negated term can later be used to reassign that action to a different atomic right. This reassignment reconstructs the initial right, at the cost of the newly assigned right.

Semantics. To interpret these new pieces of syntax, we expand our notion of a multiset to include multisets with negatively occurring objects. That is, a multiset is a function from actions $\mathcal{A}$ to the integers $\mathbb{Z}$.

$$s \in \mathcal{S}[\![a^{\perp}]\!] \qquad \text{if } s = \{a : -1\}$$

The set $\mathcal{S}[\![a^{\perp}]\!]$ contains a single multiset that contains one negative occurrence of a. Every license denotes at least one multiset free of negatively occurring objects.

Deduction. We expand our deductive system to be the full two-sided Classical Linear Logic deductive system. This gives us the following deductive theorems:

$$(a \,\&\, \psi) \otimes \varphi \vdash a \otimes (\mathbf{1} \,\&\, (a^{\perp} \otimes \psi)) \otimes \varphi \quad (1)$$

$$a \otimes (\mathbf{1} \,\&\, (a^{\perp} \otimes \psi)) \otimes \varphi \vdash (a \,\&\, \psi) \otimes \varphi \quad (2)$$

Notice Deduction 1 commits a right to authorizing an action a and Deduction 2 reverses that commitment. The transitions of license automata $\mathfrak{B}$ are defined analogously to those for $\mathfrak{A}$: $\varphi \xrightarrow{a} \varphi'$ if, and only if, $\varphi \vdash a \otimes \varphi'$.

Automata $\mathfrak{B}$ contains only formulas described by the enriched grammar. Even though the deductive system derives $\mathbf{1} \vdash a \otimes a^{\perp}$, the transition $\mathbf{1} \xrightarrow{a} a^{\perp}$ is not available because $a^{\perp}$ is not a valid state of the automata, preventing actions from being inappropriately authorized.

Lemma 10 (Soundness)**.** *$\varphi \vdash \varphi'$ implies $\varphi \models \varphi'$*

The sequences of actions authorized by automata $\mathfrak{B}$ are exactly the same as those authorized by automata $\mathfrak{A}$, for licenses present in both automata.

Lemma 11. *$\mathfrak{A}[\![\varphi]\!] = \mathfrak{B}[\![\varphi]\!]$, for all licenses φ of $\mathfrak{A}$.*

An Online Scheme. Authorization scheme $\mathsf{A}_{\mathrm{opt}}$ is defined by an algorithm for constructing a proof. Given a license φ and an action a, $\mathsf{A}_{\mathrm{opt}}$ finds a license φ' such that $\varphi \vdash a \otimes \varphi'$ by running MAX-FLOW on the following graph:

- A distinguished source s and a distinguished sink t.
- A nodes for each $a_i \in \mathrm{support}(\varphi)$
- A node for each $!\psi$, with an infinite capacity edge to t.
- A node for each ψ with a unit capacity edge to t.
- A node for each $\mathbf{1} \,\&\, (b^{\perp} \otimes \psi)$.
- A unit capacity edge from s to a.
- An edge from a_i to $!\psi$ if $a_i \in \mathrm{support}(!\psi)$.
- An edge from a_i to ψ if $a_i \in \mathrm{support}(\psi)$.
- An edge from a_i to $\mathbf{1} \,\&\, (b^{\perp} \otimes \psi)$ if $a_i \in \mathrm{support}(\psi)$.
- An edge from each $\mathbf{1} \,\&\, (b^{\perp} \otimes \psi)$ to b.

If there is a non-zero flow, then a is authorized and $\mathsf{A}_{\mathrm{opt}}$ constructs φ' as follows. Tracing the flow backwards from the sink, $\mathsf{A}_{\mathrm{opt}}$ determines the inference rules to apply. The final edge of the flow travels to the sink from a node x.

1. The flow travels to x from some action b and x is either an atomic right $b \,\&\, \psi$ or a right $!b$. In the first case, apply Deduction 1. In the second case, apply $!b \otimes \varphi \vdash b \otimes !b \otimes \varphi$. In either case, a $b \otimes$-conjunct is produced.
2. If $b = a$, then $\mathsf{A}_{\mathrm{opt}}$ halts and has produced the required φ'. Otherwise, tracing the flow backwards leads to a $\mathbf{1} \,\&\, (b^{\perp} \otimes \psi)$ node, for some ψ. Apply Deduction 2 to eliminate the $b \otimes$-conjunct and relabel the node $b \,\&\, \psi$. Return to Step 1 with this node as x.

Eventually the termination condition must obtain and the desired license formula φ' is produced. The residual flow graph is essentially the flow graph for φ' (possibly differing on edges leading to the source and from the sink). A practical implementation would not reconstruct the flow graph

for each action, but reuse the residual flow graph. The algorithm can be understood as computing MAX-FLOW over the initial rights (as in Theorem 2) by using an augmenting path for each action. Reversed edges in the residual flow graph are represented in the formula by linear negation.

Theorem 12. *Online authorization scheme* $\mathsf{A}_{\mathrm{opt}}$ *is sound, live, and monotonic.*

Proof sketch. Scheme $\mathsf{A}_{\mathrm{opt}}$ is sound because the φ' is constructed using the inference rules. Because $\mathsf{A}_{\mathrm{opt}}$ is essentially computing MAX-FLOW over the initial rights, $\mathsf{A}_{\mathrm{opt}}[\![\varphi]\!] = \mathfrak{B}[\![\varphi]\!]$ for all licenses. Thus, $\mathsf{A}_{\mathrm{opt}}$ is live and monotonic. □

Moreover, because $\mathfrak{B}[\![\varphi]\!] = \mathfrak{A}[\![\varphi]\!]$ for licenses φ in $\mathfrak{A}$, $\mathsf{A}_{\mathrm{opt}}$ is a monotonic, live, online scheme that is semantically equivalent to $\mathfrak{A}$ in the sense that $\mathsf{A}_{\mathrm{opt}}$ allows a sequence of actions if, and only if, $\mathfrak{A}$ authorizes that sequence. DRM agents employing $\mathsf{A}_{\mathrm{opt}}$ for OMA licenses achieve monotonicity without allowing extra actions by revising their internal commitment as needed to authorize more actions.

4.4 Dynamically Acquired Rights

Thus far, we have considered the static case, where consumers who first acquire rights and then perform actions. In practice, however, consumers often interleave acquiring rights and performing actions. Acquiring the right ψ can be represented by additional transitions in the automata:

$$\varphi \quad \overset{\psi}{\dashrightarrow} \quad \psi \otimes \varphi$$

These acquisition transitions can be interleaved with action transitions. Something curious occurs if a term negated in φ appears in ψ. Suppose Alice initially possessed the right $a \,\&\, b$, performed action a, and then acquired right a:

$$a \,\&\, b \xrightarrow{a} \mathbf{1} \,\&\, (a^{\perp} \otimes b) \overset{a}{\dashrightarrow} a \otimes \mathbf{1} \,\&\, (a^{\perp} \otimes b) \xrightarrow{b} \mathbf{1}$$

We hope to investigate the dynamic case in future work.

5 Conclusions

In specifying a DRM Rights Expression Language, the Open Mobile Alliance failed to consider the consequences of their authorization algorithm. On the surface, their algorithm seems plausible, but it leads to anomalous semantics, which we characterize by introducing the notion of monotonicity. We investigate the question of how to assign actions to rights in Digital Rights Management licenses. We discover that finding the optimum assignment of actions to rights in OMA licenses is NP-complete. The difficulty stems from determining when to first exercise rights with "interval" constraints. We argue this complexity is not endemic to DRM licenses as optimum assignments for a substantial fragment of the OMA language are computable efficiently.

Every online authorization scheme for OMA licenses is non-monotonic. We investigate monotonicity on two fronts. First, we characterized the maximal set of licenses on which there exists a monotonic online authorization scheme, giving rise to a monotonic approximation for arbitrary licenses. Second, monotonicity can be achieved on licenses enriched with linear negation because agents can revise their past allocation of actions to rights after learning which other actions consumers wish to perform. The authorization algorithm, which essentially computes maximum network flow, is efficiently computable, sound, live, and monotonic.

Acknowledgments. We thank Darryn McDade, Jefferson Owen, and Paul Bromley from ST Microelectronics Inc. for suggesting an investigation of the OMA DRM design. This work was partially supported by the National Science Foundation though CyberTrust grants including the PORTIA project and the TRUST Science and Technology Center

References

[1] C. N. Chong, R. Corin, S. Etalle, P. Hartel, W. Jonker, and Y. W. Law. LicenseScript: A novel digital rights language and its semantics. In *WEDELMUSIC: Proceedings of the 3rd International Conference on Web Delivering of Music*, pages 122–129, Washington, DC, 2003. IEEE Computer Society.

[2] ContentGuard. eXtensible rights Markup Language, 2006. http://www.xrml.org/.

[3] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[4] C. Gunter, S. Weeks, and A. Wright. Models and languages for digital rights. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, volume 9, page 9076, Washington, DC, 2001. IEEE Computer Society.

[5] ODRL Initiative. The Open Digital Rights Language Initiative, 2006. http://www.odrl.net/.

[6] Open Mobile Alliance. DRM Rights Expression Language: Candidate Version 2.0 — 25 Aug 2005, 2005. http://www.openmobilealliance.org/.

[7] R. Pucella and V. Weissman. A logic for reasoning about digital rights. In *CSFW '02: Proceedings of the 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, page 282, Washington, DC, 2002. IEEE Computer Society.

[8] O. Sibert, D. Bernstein, and D. V. Wie. The DigiBox: A self-protecting container for information commerce. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*.

[9] M. J. Stefik. Letting loose the light: Igniting commerce in electronic publication. *Forum on Technology-Based Intellectual Property*, pages 78–81, March 1997.