Modeling and Computing Available Rights for Algorithmic Licensing of Movies based on Blockchain

Arnold Beckmann
Swansea University
Swansea, United Kingdom
a.beckmann@swansea.ac.uk

João Santos
FEUP and MOG Technologies
Porto, Portugal
jacmsantos31@gmail.com

Indirajith Vijai Ananth
Swansea University
Swansea, United Kingdom
i.vijaiananth@swansea.ac.uk

ABSTRACT

We consider the challenge of determining which rights are available for licensing within the context of algorithmic licensing of movies using blockchain technology. We model the space of rights using set theory. We define algorithms within our modelling context that implement basic transactions for our use case, and argue for their correctness. We evaluate our algorithms through experiments within a multi-node Hyperledger Fabric network, and establish the feasibility of our algorithms for the intended use case.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**; • **Information systems → Data management systems**.

KEYWORDS

Algorithmic Video Licensing; Available Rights; Blockchain; Smart Contracts; Hyperledger Fabric; Performance Test

ACM Reference Format:
Arnold Beckmann, João Santos, and Indirajith Vijai Ananth. 2021. Modeling and Computing Available Rights for Algorithmic Licensing of Movies based on Blockchain. In *1st International Workshop on Blockchain Security, Performance and Applications, December 15, 2021, Birmingham, UK*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Blockchain Technology has been identified as an opportunity to disrupt processes in many areas, and in particular in the media industry. The licensing and payment related to trading of media assets provides a genuine use case for Blockchain Technology. The algorithmic licensing of media assets consists of two parts: First, there needs to be a way to ascertain that a set of rights is available to be licensed. Second, there needs to be a way of forming legally binding licensing agreements between a seller and a buyer. In this paper, we will focus on the former challenge of providing efficient algorithms to determine available rights.

To motivate why determining available rights efficiently is a challenge, let us consider a naive approach, in which the rights used by a license are represented by the list of all points in it, that is, all combinations of selected countries, rights, and time points. With roughly 200 countries in the world, roughly 30 rights that can be selected, and roughly 4000 days in a ten-year period to which a license may refer, we are looking at around 20 million individual data points that form such a license. Thus, comparing two such licenses, to see whether they overlap or not, we are making $4 \cdot 10^{14}$ comparisons, which even on the fastest PC would take more than a day (Intel Core i9-9900K has 412,090 MIPS at 4.7 GHz, i.e., roughly $4 \cdot 10^9$ instructions per second, thus the above would take at least 10^5 seconds which is a bit more than 27 hours). This shows that a naive way of representing rights is not a feasible approach to determining available rights.

The main idea to overcome this challenge is to use set-theoretic modeling, but instead of computing with sets directly, we will compute with a representation of them. The representation will be simple as we are restricting our focus on rectangular objects in higher dimensions, which we will call Rights Hyper Ractangles. We will represent those Rights Hyper Ractangles as the list of sets for their sides. We will justify that this representation is feasible, that is, that all necessary computations for determining available rights can be done effectively on such representations. In the example above, to see that such licenses are not overlapping, it will be sufficient to compare the two sets of countries (40000 comparisons), the two sets of rights (900 comparisons), and the two intervals (4 comparisons of boundaries), which in total will be $4 \cdot 10^4$ comparisons instead of $4 \cdot 10^{14}$ comparisons.

The rest of the paper is organized as follows: The next section will discuss related work. In Section 3 we will introduce the use case behind the Media Asset Platform (MAP) project that is the basis for this paper, discuss its relevance as a blockchain application, and conclude with the main transaction that this use case needs to support. Section 4 will define a modelling of the rights space based on set theory, which is followed by a description of the algorithms implementing the identified main transactions in Section 5. In Section 6 we describe the evaluation of our algorithms in a multi-node Hyperledger Fabric Blockchain network. We conclude the paper in Section 7.

2 RELATED WORK

Although there is a lack of literature related to mathematical approaches of legal rights modeling, there are some copyright modeling and legal implementations using Blockchain and Smart Contracts. For example, [9] approaches the development of a blockchain service to improve data traceability and protection effectiveness. In [4], an adaptation of the applicable law in smart contracts is introduced and is researched the *Lex Cryptographia* theory, which

BlockSPAN 2021, December 15, 2021, Birmingham, UK

ACM ISBN 978-1-4503-XXXX-X/21/12…$15.00
https://doi.org/10.1145/1122445.1122456

predicts that algorithms will eventually replace the law and the judicial system and its interaction with the current legal system. The authors in [2] and [6] research and implement Blockchain-based copyright systems to help improve authorization, confirmation, and management of rights with Blockchain's advantages such as transparency, easy data traceability, decentralization, and immutability. Jeonghee and Lee Chi et al. [1] investigated a system to perform secure and reliable eBook transactions to prevent piracy and illegal distribution. Nugent et al. [5] introduce a pioneering approach to blockchain-based entitlement management, where content owners are matched with their content's digitally encoded rights using a proper algorithm. BMCProtector [8] is a Blockchain and Smart Contract based technology to protect and manage music copyrights using Ethereum[1]. In [7] it is also researched a blockchain approach for digital music copyright management to take advantage of Blockchain's advantages. The authors in [3] designed a Blockchain copyright protection system based on the Big Data Copyright Protection system, using Hyperledger Fabric.

3 USE CASE

3.1 The Media Asset Platform (MAP) Project

This work is integrated within the European MAP Project, which is being developed by a consortium in partnership with MetFilm Production[2], MOG Technologies[3], Swansea University[4], Birmingham City University[5], and CEG Analytics Ltd. The MAP Project is supported by the European Institute of Innovation and Technology[6] (EIT), a body of the European Union. The MAP Project aims to build a secure, simple and low-cost new business-to-business Marketplace between content creators (i.e., indie filmmakers) and distributors (i.e., "Over-the-top" channels).

3.2 Genuine Blockchain Application

Nowadays, there is a broad usage of centralized architecture storage systems, which provide high reliability, enhanced consistency, simplicity, and easy data management. However, these systems also embody significant drawbacks, such as low data redundancy and safety concerns due to hacker attacks and network breakdowns. Due to these drawbacks, these systems may prevent data retrieval or even originate data loss in case of node failures. The most commonly used centralized technologies are database management systems, such as MySQL[7], PostgreSQL[8], and Microsoft SQL Server[9]. Another significant disadvantage of centralized systems is the accumulation of all decision-making on only one entity or organization. This is a massive issue since any bad management decisions may significantly affect sensitive data.

In decentralized systems, the system's decisions reflect the democratic voices of all, or at least of the majority, of its members. These systems offer a high degree of security due to Crash or Byzantine Fault Tolerance, node misbehavior, and attacks, and, since all information is stored equally in every network peer, it is straightforward to retrieve information due to its redundancy. Blockchain stands out from all decentralized storage systems for its data immutability, traceability features, and being able to run smart contracts that simulate real-life contracts, execute business processes and rules through programming scripts, and automate transaction processes.

Blockchain is an append-only data structure, where new blocks of transaction data are added and linked to the previously added blocks through cryptographic keys. The hash of a block's content forms keys. The slightest change in the content of a transaction will produce a different block hash. For a hacker node to tamper with a blockchain, one needs to recompute all blockchain from the targeted node till the end. Since all information is chronologically registered and all link to each other it is effortless to trace information throughout the Blockchain storage system.

In this project, we have chosen Hyperledger Fabric[10] Blockchain Technology to execute transactions and compute available rights. Hyperledger Fabric is an open source blockchain framework supported by the Linux Foundation.[11] It provides a permissioned private blockchain solution which is suitable to support enterprise solutions. The Fabric blockchain uses RAFT consensus which is crash fault tolerant. As a permissioned blockchain there is no need to employ Byzantine Fault Tolerant consensus, because only pre-approved actors and network nodes can be part of and participate in a permissioned blockchain system.

3.3 User Journeys and Transactions

In this work, the types of actors that will interact with the system will be Buyers and Sellers.

A Seller is a Filmmaker or motion Picture rights owner who has a goal of selling licenses, and for that, he or she provides the platform with a list of films and creates a list of available rights for each film.

A Buyer is a distributor or a rights reseller with the goal of acquiring film licenses. The Buyer then searches and browses films of interest from the film catalog and customizes a license according to one's needs.

In line with the way blockchain systems operate, we will consider transactions that can be initiated by the various actors of the MAP ecosystem. Transactions will define atomic transitions of states, which are used to evolve the state of the MAP blockchain.

We will consider three basic transactions between Seller and Buyer as the basis of our investigations.

- **SellerSetsAvRights** — Seller sets the rights which are available for sale.
- **SellerBlocksRights** — Seller has the ability to block parts of the available rights which may be unavailable due to previous or independent transactions.
- **BuyerLicencesRights** — A Buyer needs to fix a licensing agreement from available rights.

[1] Ethereum: https://ethereum.org
[2] MetFilm Production: https://www.metfilmproduction.co.uk
[3] MOG Technologies: https://www.mog-technologies.com
[4] Swansea University: https://www.swansea.ac.uk
[5] Birmingham City University: https://www.bcu.ac.uk
[6] EIT: https://eit.europa.eu
[7] MySQL: https://www.mysql.com/
[8] PostgreSQL: https://www.postgresql.org
[9] Microsoft SQL Server: https://www.microsoft.com/pt-pt/sql-server/sql-server-downloads
[10] Hyperledger Fabric: https://www.hyperledger.org/use/fabric
[11] Linux Foundation: https://linuxfoundation.org/

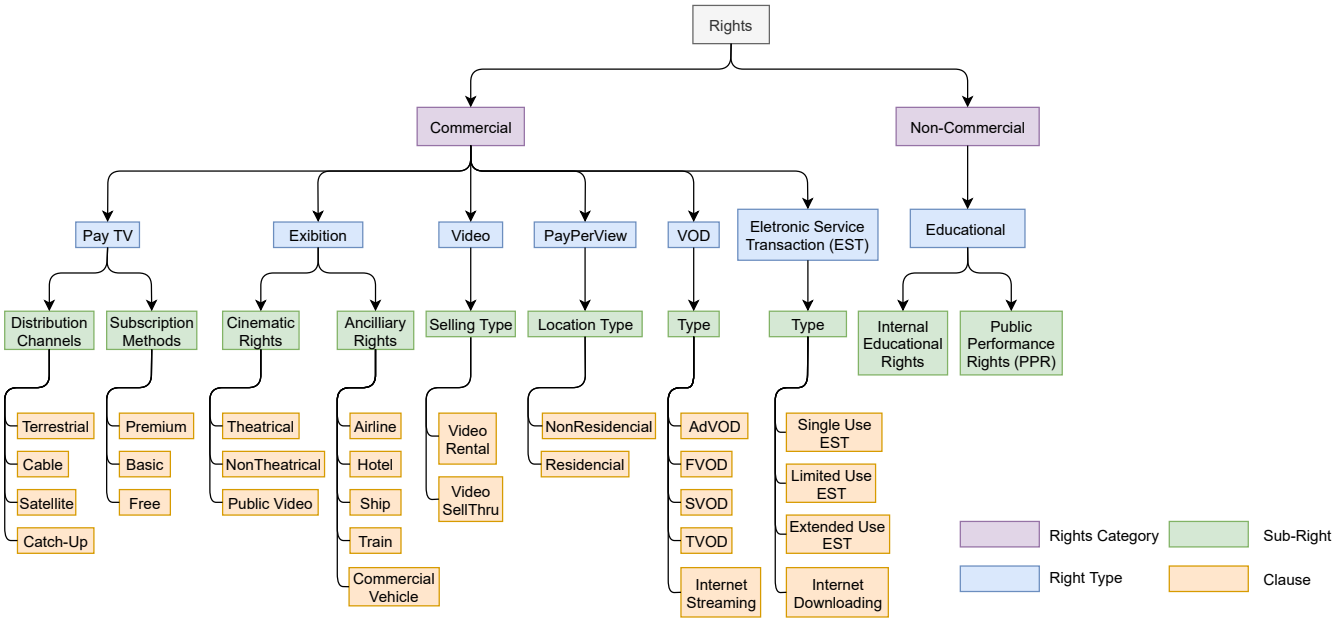

Figure 1: Type of Rights

4 MODELING

The rights we consider within the MAP project are made up from several components coming from different spaces. In the following we identify those spaces, develop a mathematical abstraction for them, and derive essential properties for our use case.

4.1 The Space of Rights (SoR)

4.1.1 Type of rights. With $\mathcal{R}$ we denote the set of rights that can be in principle licensed individually. In Fig. 1 it will be the set of all orange colored field options. We will use r etc to denote members of $\mathcal{R}$, and R for subsets of $\mathcal{R}$.

Remark. *The data structure implementing $\mathcal{R}$ will also allow identifiers denoting groups of rights as in Fig. 1. For example, we will use an identifier for 'VOD' to denote all VOD rights.*

4.1.2 Territories. With C we denote the set of territories (countries), or more precisely, the set of smallest geographical entities that can be part of a right. We will use c etc to denote members of C, and C to denote subsets of C.

Remark. *The data structure implementing C will in addition to identifiers for territories also allow other identifiers denoting groups of territories as in Fig. 2. For example, we will use an identifier for 'world', i.e., C.*

4.1.3 Time space. With $\mathcal{T}$ we denote the set that denotes time. A continuous space like reals is unnecessary for our use case, because a license will be bound to discrete dates. We will use integers. which are intended to refer to days, viewed as time points. We will use t etc to denote members of $\mathcal{T}$, and T to denote subsets of $\mathcal{T}$.

4.1.4 Exclusivity. With $\mathcal{E} = \{\text{ex}, \text{nex}\}$ we denote that rights can be taken exclusively or non-exclusively. We will use e etc to denote members of $\mathcal{E}$.

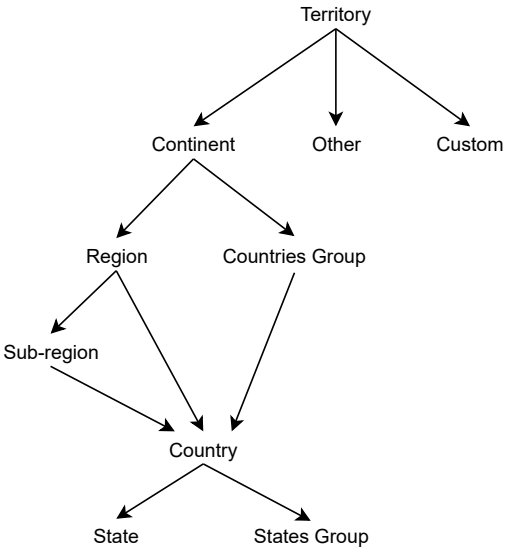

Figure 2: Territories

4.1.5 Space of Rights (SoR). Based on the components territories C, type of rights $\mathcal{R}$, time space $\mathcal{T}$, and exclusivity $\mathcal{E}$, we define the *Space of Rights* SoR as

$$\text{SoR} \quad = \quad C \times \mathcal{R} \times \mathcal{T} \times \mathcal{E}$$

As a 4-dimensional space, SoR is difficult to visualize in 2-dimensions, thus the drawing in Fig. 3 has to be viewed with caution. A point in this space is of the form

$$(c, r, t, e) \in \text{SoR}$$

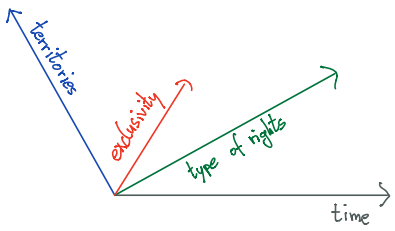

Figure 3: 4-dimensional Space of Rights

meaning that c is a territory in C; r is a type of rights in $\mathcal{R}$; t is a time point in $\mathcal{T}$; e is either ex or nex.

4.2 State of Available Rights

4.2.1 Rights Hyper Rectangle (RHR). We will consider hyper rectangular sub-spaces in SoR which we also call boxes. Again, it is difficult to draw such 4-dimensional boxes in 2 dimensions, an attempt is given in Fig. 4.

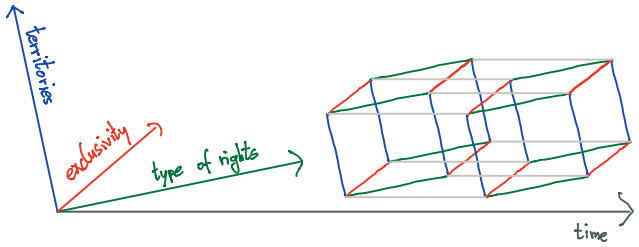

Figure 4: 4-dimensional Rights Hyper Rectangle

Definition 1. A *Rights Hyper Rectangle (RHR)*, or *Box*, β is given by a nonempty set

$$\beta \quad = \quad C \times R \times T \times E$$

where $C \subseteq \mathcal{C}$, $R \subseteq \mathcal{R}$, T is an interval in $\mathcal{T}$, and $E \subseteq \mathcal{E}$.

4.2.2 State of Available Rights. The *state of available rights* Σ is determined by a RHR μ (that is set by Seller) and a sequence $\beta_1, \ldots, \beta_k$ of RHRs (that are either blocked by Seller or blocked through a license taken by a buyer). We denote this state as

$$\Sigma \quad = \quad \Sigma(\mu, \beta_1, \ldots, \beta_k)$$

The set of available rights in state $\Sigma(\mu, \beta_1, \ldots, \beta_k)$ is given by

$$\text{available}(\Sigma(\mu, \beta_1, \ldots, \beta_k)) \quad = \quad \mu \setminus \bigcup_{1 \le i \le k} \beta_i$$

Remark. *We use RHRs as the basic building blocks for defining the space of potential rights that the seller wants to offer, for defining spaces of rights that the seller wants to remove from previously offered ones, for expressing rights related to licenses that seller and buyer agree upon, and for expressing the space of rights that needs to be removed in the latter case.*

5 ALGORITHMS

5.1 Transactions revisited

In Subsection 3.3 we have identified basic transactions between Buyer and Seller that need to be supported. We will now formalise those transactions and define how they act on the state of available rights.

- For transaction **SellerSetsAvRights**, Seller identifies a RHR μ which defines the initial state $\Sigma_0 := \Sigma(\mu)$.

For other transactions we assume that the system is in state $\Sigma_k = \Sigma(\mu, \vec{\beta})$.

- For transaction **SellerBlocksRights**, Seller chooses a RHR β. The new state is $\Sigma_{k+1} = \Sigma(\mu, \vec{\beta}, \beta)$.
  A seller may do this at the beginning to exclude already sold rights, however it may also be allowed at later stages.
- With transaction **BuyerLicencesRights**, Seller and Buyer fix a licensing agreement for a RHR of the form $C \times R \times T \times \{e\}$. Let $\alpha = C \times R \times T$, see Fig. 5.

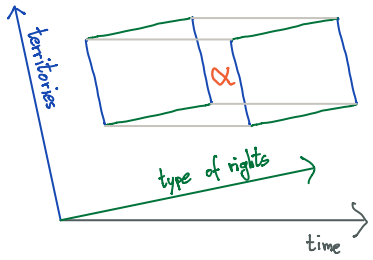

Figure 5: 3-dimensional sub Rights Hyper Rectangle

If $e = \text{ex}$, α is fixed *exclusively*. Thus, the condition

$$\alpha \times \{\text{ex}\} \subseteq \text{available}(\Sigma_k)$$

needs to be satisfied. As a consequence of executing this transaction, both the exclusive and non-exclusive versions of α need to be removed from the set of available rights, which is expressed in the following way: Let β be $\alpha \times \{\text{ex}, \text{nex}\}$. Then the new state is $\Sigma_{k+1} = \Sigma(\mu, \vec{\beta}, \beta)$.
If $e = \text{nex}$, α is fixed *non-exclusively*. Thus, the condition

$$\alpha \times \{\text{nex}\} \subseteq \text{available}(\Sigma_k)$$

needs to be satisfied. As a consequence of fixing rights non-exclusively, their exclusive version will become unavailable, which is expressed by removing the exclusive versions of α from the current state, while keeping the non-exclusive version. This is defined formally in the following way:

- If $\alpha \times \{\text{ex}\} \subseteq \text{available}(\Sigma_k)$, then let $\beta = \alpha \times \{\text{ex}\}$ and define $\Sigma_{k+1} = \Sigma(\mu, \vec{\beta}, \beta)$.
- Otherwise, let $\Sigma_{k+1} = \Sigma_k$.

Remark. *The cases of fixing a licensing agreement also cover other forms of exclusive or non-exclusive scenarios, like selling number of emissions, or educational licenses.*

5.2 Properties of RHRs

The state of available rights has been modeled on the basis of RHRs as it simplifies the algorithmic approach for computing available rights. Here we will give the properties that will later form the basis of our algorithms

Lemma 1. $\beta \subseteq \text{available}(\Sigma(\mu, \vec{\beta}))$ *iff*

$$\beta \subseteq \mu \ \wedge \ \forall i, \beta \cap \beta_i = \emptyset$$

Proof. The proof follows immediately from the definition of $\text{available}(\Sigma(\mu, \vec{\beta}))$ as $\mu \setminus \bigcup_{1 \le i \le k} \beta_i$ using elementary set theoretic identities. □

For the following two Lemmas we consider two RHRs β_1 and β_2, with

$$\beta_i \quad = \quad C_i \times R_i \times T_i \times E_i$$

Lemma 2. $\beta_1 \subseteq \beta_2$ *iff*

$$C_1 \subseteq C_2 \ \wedge \ R_1 \subseteq R_2 \ \wedge \ T_1 \subseteq T_2 \ \wedge \ E_1 \subseteq E_2$$

Proof. $\Leftarrow$ clear.
$\Rightarrow$: Assume $\beta_1 \subseteq \beta_2$. Fix $(c, r, t, e) \in \beta_1$. To show $C_1 \subseteq C_2$ let $x \in C_1$. Then $(x, r, t, e) \in \beta_1 \subseteq \beta_2$, hence $x \in C_2$.
Similar for other components. □

Lemma 3. $\beta_1 \cap \beta_2 = \emptyset$ *iff*

$$C_1 \cap C_2 = \emptyset \ \vee \ R_1 \cap R_2 = \emptyset \ \vee \ T_1 \cap T_2 = \emptyset \ \vee \ E_1 \cap E_2 = \emptyset$$

Proof. $\Leftarrow$ clear.
$\Rightarrow$: We argue indirectly. Let $c \in C_1 \cap C_2$, $r \in R_1 \cap R_2$, $t \in T_1 \cap T_2$, and $e \in E_1 \cap E_2$. Then $(c, r, t, e) \in \beta_1 \cap \beta_2$. □

As we can see, we mainly need to implement tests of disjointness and subset for components. For those we can also allow data structures that include some form of hierarchical representation of subsets, see remarks in Subsection 4.1. One example of this is that we can allow 'world' to represent the set of all territories.

5.3 Data Structures

5.3.1 Base types. We will need base types for representing the set of territories $\mathcal{C}$, the set of rights $\mathcal{R}$, the time space $\mathcal{T}$, and the exclusivity space $\mathcal{E}$ from Section 4.1. We call these types

- CType for territories $\mathcal{C}$
- RType for rights $\mathcal{R}$
- TType for time $\mathcal{T}$
- EType for exclusivity $\mathcal{E}$

5.3.2 Interface for unsorted sets. We define an interface for a data structure of unsorted sets over our base types. An unsorted set contains distinct elements: no element appears more than once, and the elements are in no specific order. The following operations are supported:

- create() creates an empty set.
- add(x) adds x to set if not already present.
- disjoint(s_1, s_2) tests whether sets s_1 and s_2 are disjoint.
- subset(s_1, s_2) tests whether s_1 is a subset of s_2, that is, that all elements of s_1 are also elements of s_2.

5.3.3 Interface for intervals. We define a special interface for intervals that are defined by start and end dates, where the start date has to come before the end date. The following operations are supported:

- create(s, e) creates an interval with start date s and end date e.
- disjoint(i_1, i_2) tests whether intervals i_1 and i_2 are disjoint.
- subset(i_1, i_2) tests whether i_1 is a subset of i_2.

5.3.4 Interfaces for Rights Hyper Rectangles. Rights Hyper Rectangles, denoted RHR, will be 4-tuples with components consisting of a set of type CType, a set of type RType, an interval of type TType, and a set of type EType. They will support the following operations:

- createRHR(C, R, T, E) creates an RHR from the listed components.
- projC extracts the CType component of an RHR.
- projR extracts the RType component of an RHR.
- projT extracts the TType component of an RHR.
- projE extracts the EType component of an RHR.

5.3.5 State of available rights. The state of available rights Σ consists of 2 components, the μ component which is a RHR set be seller, and a list of β_i which are RHRs denoting blocked parts. The following operations will be supported:

- create(x) creates a state with μ-component x and empty list of βs.
- append(y) appends y to the list of βs.
- projμ extracts the μ component from a state.
- projβ extracts the β-list component from a state.

5.4 Algorithms

We start by describing basic algorithms for disjointedness- and subset-relations between RHRs.

5.4.1 Basic algorithms. Following Lemma 3, disjointedness between RHRs can be computed via the disjointedness of their components as described in Algorithm 1.

Algorithm 1 Disjointedness relation for RHRs

```
1:  function DISJOINT(β1, β2)
2:      if disjoint(projE(β1), projE(β2)) then
3:          return true
4:      else if disjoint(projT(β1), projT(β2)) then
5:          return true
6:      else if disjoint(projR(β1), projR(β2)) then
7:          return true
8:      else if disjoint(projC(β1), projC(β2)) then
9:          return true
10:     else
11:         return false
12:     end if
13: end function
```

In a similar way following Lemma 2, Algorithm 1 computes the subset relation between RHRs utilizing the subset relation of their components.

Algorithm 2 Subset relation for RHRs

```
1:  function SUBSET(β1, β2)
2:      if ¬ subset(projE(β1), projE(β2)) then
3:          return false
4:      else if ¬ subset(projT(β1), projT(β2)) then
5:          return false
6:      else if ¬ subset(projR(β1), projR(β2)) then
7:          return false
8:      else if ¬ subset(projC(β1), projC(β2)) then
9:          return false
10:     else
11:         return true
12:     end if
13: end function
```

Algorithm 3 tests whether an RHR is available for a given state, based on the identity given in Lemma 1.

Algorithm 3 Availability relation

```
1:  function AVAILABLE(β, Σ)
2:      if ¬SUBSET(β, projμ(Σ)) then
3:          return false
4:      end if
5:      for each x in projβ(Σ) do
6:          if ¬DISJOINT(β, x) then
7:              return false
8:          end if
9:      end for
10:     return true
11: end function
```

5.4.2 State update algorithms. The following algorithms implement state transitions based by transactions identified in Subsection 5.1. They all modify a global structure which represents the current state of available rights.

We start with identifing a global state variable and its initialisation in Algorithm 4. Procedure INITIALISE STATE implements Transaction **SellerSetsAvRights**.

Algorithm 4 Creating a state

```
1:  global variables
2:      Σ
3:  end global variables
4:
5:  procedure INITIALISE STATE(μ)
6:      Σ ← create(μ)
7:  end procedure
```

Algorithm 5 blocks a set of rights by appending it to the β-list of the global state, implementing **SellerBlocksRights**.

We now come to the main algorithms for taking out available rights. We distinguish cases according to whether rights are taken out exclusively or non-exclusively, starting with the exclusive case first, following our discussion of these cases in Subsection 5.1.

Algorithm 5 Blocking a set of rights

```
1:  procedure BLOCK(β)
2:      Σ. append(β)
3:  end procedure
```

Algorithm 6 Taking out exclusive rights

```
1:  procedure EXCLUSIVE(C,R,T)
2:      β ← createRHR(C, R, T, {ex})
3:      if ¬AVAILABLE(β, Σ) then
4:          abort 'Rights not available'
5:      end if
6:      β ← createRHR(C, R, T, {ex, nex})
7:      Σ. append(β)
8:  end procedure
```

Algorithm 7 Taking out non-exclusive rights

```
1:  procedure NONEXCLUSIVE(C,R,T)
2:      β ← createRHR(C, R, T, {nex})
3:      if ¬AVAILABLE(β, Σ) then
4:          abort 'Rights not available'
5:      end if
6:      β ← createRHR(C, R, T, {ex})
7:      if AVAILABLE(β, Σ) then
8:          Σ. append(β)
9:      end if
10: end procedure
```

Algorithm 7 implements taking out non-exclusive rights.

Algorithms 6 and 7 together implement Transaction **BuyerLicencesRights**.

5.5 Correctness of Algorithms

Based on their developments and modelling, the previously defined algorithms satisfy the following correctness properties.

THEOREM 1. *Let* $\Sigma = \text{available}(\Sigma(\mu, \beta_1, \ldots, \beta_k))$, *and let* $\alpha = C \times R \times T$.

If α *is available exclusively in* Σ, *that is,* $\alpha \times \{\text{ex}\} \subseteq \text{available}(\Sigma)$, *then* Σ.EXCLUSIVE(C, R, T) *returns true. Furthermore, in the updated space* α *is not available neither exclusively nor non-exclusively.*

If α *is available non-exclusively in* Σ, *that is,* $\alpha \times \{\text{nex}\}$ *being a subset of* $\text{available}(\Sigma)$, *then* Σ.NONEXCLUSIVE(C, R, T) *returns true. Furthermore, in the updated space* α *is not available exclusively.*

6 EXPERIMENTS

We describe the evaluation of our algorithms in an experimental Hyperledger Fabric network run at Swansea University. The experimental setup consists of three parts: the blockchain network, the smart contracts (called chaincode in Fabric) running on the blockchain, and a client application interacting with the blockchain by invoking chaincode.

6.1 Blockchain Network

A blockchain network relies on cryptography and Public Key Infrastructure (PKI). Thus, Certificate Authorities (CAs) form the basis for a fabric network. An organisational structure is replicated in generating public, private cryptographic keys and certificates. These keys and certificates are used when starting various nodes of the network.

The Fabric blockchain network is made from various nodes which form the network. Peer nodes store two kinds of data. First one is the ledger data which stores all the transaction data. From the ledger data the world state can be calculated which contains the current state of the blockchain system.

Peers can either be endorsing peers or ledger peers. Peers with installed smart contracts are involved in transaction verification and endorsement along with hosting ledgers. Ledger peers do not have installed smart contracts. They just host the ledger.

Orderer nodes verify the identities are endorsements and check for endorsement policy requirements before creating blocks of transactions. The blocks are then disseminated to peers for updating the ledgers.

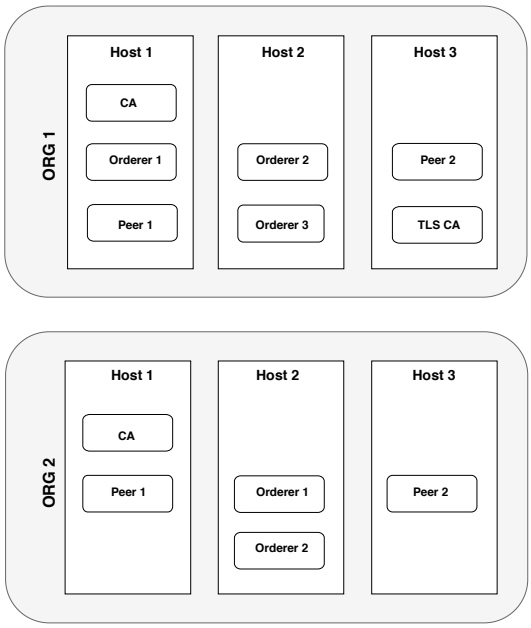

Figure 6: Fabric Blockchain Network

The experimental setup is based on two organisations, Org1 and Org2. Fig. 6 shows the Blockchain network and its components with their organisational affiliation. Each organisation has two peers to avoid a single point of failure. All peers are endorsing peers involving in transaction endorsements.

The number of orderers are choosen to fullfil the RAFT consensus algorithm. To withstand n number of node failures, $2n + 1$ orderer nodes are required. If more than n number of nodes go down the network will not achieve quorum, and consensus would not be reached rendering the blockchain network to stop processing transactions. In our network, Org1 has three orderers, Org2 two, for a total of five orderers to withstand two orderer nodes failures.

All fabric nodes are hosted across six virtual servers running docker containers. TLS is enabled for communication between the nodes. One TLS CA for both organisations is used to achieve this. To enable advanced queries and store the world state persistently, CouchDB is used with all peers.

6.2 Chaincode and Clients

The basic transaction flow starts from a client who is part of an organisation participating in the fabric network. A client initiates a transaction by forming a transaction proposal. The client makes a request to invoke a chaincode function with certain parameters and sent them to endorsing peers. Endorsing peers verify signatures and execute the chaincode independently to generate a transaction proposal response. These responses are called endorsements and they are sent back to the client. The client collects all the endorsements and sends them to orderer. Orderer nodes verify their signature and checks for endorsement policy.

Endorsement policy is set at channel level. The endorsements are accepted and made into a block only if it fulfils endorsement policy. In our experimental setup both organisation need to endorse a transaction to be valid. This requires peers from both organisations to run the chaincode and endorse a transaction to be valid. The blocks are then sent to all peers in the channel.

The core of the experiment are smart contract. The algorithms from the previous section are implemented in Golang and deployed onto our blockchain network 6. A client application written in javascript using Fabric Nodejs API is used to interact with the blockchain, invoke the chaincode functions and test it.

6.3 Evaluations

The performance of algorithms has been evaluated in the following way. The client initiates the experiment by invoking chaincode to create a state based on 200 countries, 30 rights, exclusive and non-exclusive, and a time span of 50000 days (which approximates to roughly 137 year), using Algorithm 4. We then block or take out rights based on random RHRs formed from 20 countries, 3 rights, either exlusive or non-exclusive, and an interval of 200 days. We do this by alternating the following steps: We either call chaincode that implements Algorithm 5 to block 500 of such randomly chosen RHRs. Or we repeatedly for 100 times choose one randomly chosen RHR and call chaincode that tries to take out rights by implementing Algorithms 6 and 7. In the latter case we measure the time the invocation of chaincode takes. The min, max and mean of those times for different sizes of blocked rights spaces are displayed in Fig. 7.

The execution time jumps between number of blocked RHRs going from 1500 to 2000. The reason for this is the configuration of the ordering system in the Fabric network: The ordering service is minting a new block either after a timeout has occurred, or the size of transaction proposals exceeding a set value. The default timout is 2secs, which needs to be waited until the size of the state build from blocked RHRs exceeds the threshold value somewhere between 1500 and 2000 blocks. Therefore, we consider adjusted times where the timeout has been subtracted.

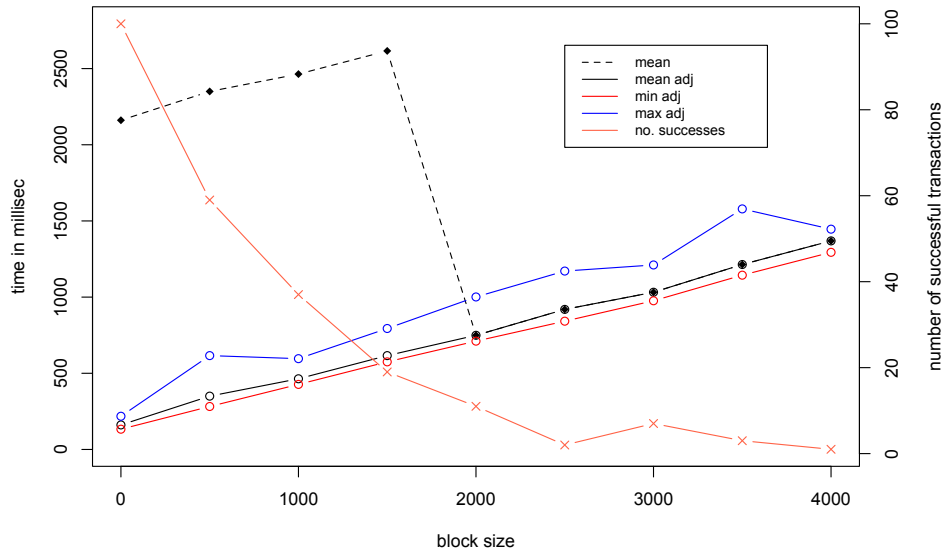

Figure 7: The performance of Algorithms 6 and 7 on blocked spaces of various sizes

We also indicate in Fig. 7 how many of the 100 attemps to take out rights have been successful: Initially, when the number of blocked RHRs is 0, all 100 attempts are successful. The more RHRs have been blocked, the less likely it is to take our further rights until hardly any are successful (only 1 of 100 attempts is successful once 4000 RHRs have been blocked).

The diagram in Fig. 7 demonstrates that there is a linear dependency between the number of blocked RHRs and the time it takes to take out another set of rights.

7 CONCLUSION

In this paper, we have provided a modeling of licensing rights based on set theory. We have defined algorithms that implement the main transactions **SellerSetsAvRights**, **SellerBlocksRights**, **BuyerLicencesRights** between a seller and a buyer for the MAP use case, and argued for correctness and efficientcy of those algorithms through careful development in relation to mathematical models. We evaluated the algorithms through suitable experiments within a multi-node Hyperledger Fabric Blockchain network that has also been used to deploy the MAP project.

In future work, we will expand our framework in several ways. We will allow a seller to choose several right spaces at the start, as this is a realistic scenario in cases where a seller wants to import existing licensing agreements into the MAP system. We will also expand the modeling of components (like countries) to support named subspaces, as this would align with the interface design and the preferred way a used would want to interact with the system.

REFERENCES

[1] Jeonghee Chi, Jangyeon Lee, Nakyung Kim, Jeewoo Choi, and Soyoung Park. 2020. Secure and reliable blockchain-based eBook transaction system for self-published eBook trading. *PLOS ONE* 15, 2 (02 2020), 1–33. https://doi.org/10.1371/journal.pone.0228418

[2] Jun Guo, Hongbo Zhou, Lan Yang, and Xuhui Chen. 2020. Research on digital copyright blockchain technology. In *2020 3rd International Conference on Smart BlockChain (SmartBlock)*. 1–5. https://doi.org/10.1109/SmartBlock52591.2020.00028

[3] Jieyi Long and Haiquan Wang. 2019. Design of Blockchain System in BDCP Using Hyperledger Fabric. In *Proceedings of the 2019 The World Symposium on Software Engineering* (Wuhan, China) *(WSSE 2019)*. Association for Computing Machinery, New York, NY, USA, 78–82. https://doi.org/10.1145/3362125.3362127

[4] Ana Mercedes López Rodríguez. 2021. Ley aplicable a los smart contracts y lex crytographia. *CUADERNOS DE DERECHO TRANSNACIONAL* 13, 1 (mar. 2021), 441–459. https://doi.org/10.20318/cdt.2021.5966

[5] Timothy Nugent, Fabio Petroni, Benedict Whittam Smith, and Jochen L. Leidner. 2019. An On-Chain Method for Automatic Entitlement Management Using Blockchain Smart Contracts. In *Business Information Systems Workshops*, Witold Abramowicz and Rafael Corchuelo (Eds.). Springer International Publishing, Cham, 255–266.

[6] Yi Ouyang, Xianghan Zheng, Xiaoliang Lu, Lin Xiaowei, and Shengyin Zhang. 2019. Copyright Protection Application Based on Blockchain Technology. In *2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*. 1271–1274. https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00182

[7] Yaping Zeng. 2020. Digital Music Resource Copyright Management Mechanism Based on Blockchain. In *2020 3rd International Conference on Smart BlockChain (SmartBlock)*. 158–162. https://doi.org/10.1109/SmartBlock52591.2020.00036

[8] Sijia Zhao and Donal O'Mahony. 2018. BMCProtector: A Blockchain and Smart Contract Based Application for Music Copyright Protection. In *Proceedings of the 2018 International Conference on Blockchain Technology and Application* (Xi'an, China) *(ICBTA 2018)*. Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/3301403.3301404

[9] Peng Zhu, Jian Hu, Xiaotong Li, and Qingyun Zhu. 2021. Using Blockchain Technology to Enhance the Traceability of Original Achievements. *IEEE Transactions on Engineering Management* (2021), 1–15. https://doi.org/10.1109/TEM.2021.3066090