Algorithmic Regulation using AI and Blockchain Technology

Hirsh Jaykrishnan Pithadia

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

November 6, 2021

I, Hirsh Jaykrishnan Pithadia, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis investigates the application of AI and blockchain technology to the domain of Algorithmic Regulation. Algorithmic Regulation refers to the use of intelligent systems for the enabling and enforcement of regulation (often referred to as RegTech in financial services). The research work focuses on three problems: a) Machine interpretability of regulation; b) Regulatory reporting of data; and c) Federated analytics with data compliance. Uniquely, this research was designed, implemented, tested and deployed in collaboration with the Financial Conduct Authority (FCA), Santander, RegulAItion and part funded by the InnovateUK RegNet project. I am a co-founder of RegulAItion.

Using AI to Automate the Regulatory Handbook
In this investigation we propose the use of reasoning systems for encoding financial regulation as machine readable and executable rules. We argue that our rules-based white-box approach is needed, as opposed to a black-box machine learning approach, as regulators need explainability and outline the theoretical foundation needed to encode regulation from the FCA Handbook into machine readable semantics. We then present the design and implementation of a production-grade regulatory reasoning system built on top of the Java Expert System Shell (JESS) and use it to encode a subset of regulation (consumer credit regulation) from the FCA Handbook. We then perform an empirical evaluation, with the regulator, of the system based on its performance and accuracy in handling 600 real- world queries and compare it with its human equivalent. The findings suggest that the proposed approach of using reasoning systems not only provides quicker responses, but also more accurate results to answers from queries that are explainable.

SmartReg: Using Blockchain for Regulatory Reporting
In this investigation we explore the use of distributed ledgers for real-time reporting of data for compliance between firms and regulators. Regulators and firms recognise the growing burden and complexity of regulatory reporting resulting from the lack of data standardisation, increasing complexity of regulation and the lack of machine executable rules. The investigation presents a) the design and implementation of a permissioned Quorum-Ethereum based regulatory reporting network that makes use of an off-chain reporting service to execute machine readable rules on banks data through smart contracts b) a means for cross border regulators to share

reporting data with each other that can be used to given them a true global view of systemic risk c) a means to carry out regulatory reporting using a novel pull-based approach where the regulator is able to directly pull relevant data out of the banks environments in an ad-hoc basis- enabling regulators to become more active when addressing risk. We validate the approach and implementation of our system through a pilot use case with a bank and regulator. The outputs of this investigation have informed the Digital Regulatory Reporting initiative- an FCA and UK Government led project to improve regulatory reporting in the financial services.

RegNet: Using Federated Learning and Blockchain for Privacy Preserving Data Access
In this investigation we explore the use of Federated Machine Learning and Trusted data access for analytics. With the development of stricter Data Regulation (e.g. GDPR) it is increasingly difficult to share data for collective analytics in a compliant manner. We argue that for data compliance, data does not need to be shared but rather, trusted data access is needed. The investigation presents a) the design and implementation of RegNet- an infrastructure for trusted data access in a secure and privacy preserving manner for a singular algorithmic purpose, where the algorithms (such as Federated Learning) are orchestrated to run within the infrastructure of data owners b) A taxonomy for Federated Learning c) The tokenization and orchestration of Federated Learning through smart contracts for auditable governance. We validate our approach and the infrastructure (RegNet) through a real world use case, involving a number of banks, that makes use of Federated Learning with Epsilon-Differential Privacy for improving the performance of an Anti-Money-Laundering classification model.

Contributions to Science and Industry Impact
The major contributions of this work are:

- A means to capture machine readable and executable semantics of the Financial Conduct Authority regulation Handbook; which can then be used to automate regulatory compliance

- A regulatory data reporting infrastructure that could realistically used in industry- leading to the further development of the Digital Regulatory Reporting initiative by the regulator

- A trusted data access infrastructure for singular algorithmic purposes (such as Federated Machine Learning) that is compliant with Data Regulation

- A Taxonomy for Federated Learning

- Co-founding RegulAItion- a RegTech company focused on delivering research driven, enterprise Algorithmic Solutions for industry

- UKRI InnovateUK funded RegNet project (application number 45079) for further research

Impact Statement

This thesis investigates the application of AI and blockchain technology for Algorithmic Regulation. The work has been done in conjunction with the FCA, Santander, RegulAItion, The ODI and a number of other banks and law firms. The design decisions, implementation and deployment decisions were for the three experiments: Machine Interpretability of Regulation; SmartReg and RegNet were made with their input. Experimental work was undertaken with the collaborators to validate the design choices.

The first experiment has been made into a standalone recommender system that has been employed by the regulator for its internal permissions process. The design decisions made in SmartReg have informed a wider project carried out by the regulator and a number of banks. This is the Digital Regulatory Reporting Project (DRR). DRR has become a priority for the regulator, banks and the UK financial services industry. The work undertaken as part of RegNet was part of an InnovateUK funded project and has been applied to a number of use cases and has been used to create an infrastructure platform for trusted data access that has received commercial interest. The thesis also provides a Taxonomy for Federated Learning- a growing sub-field of machine learning.

List of Publications and Grants

List of Publications

- Smietanka, Malgorzata and Pithadia, Hirsh and Treleaven, Philip, Federated Learning for Privacy-Preserving Data Access (September 15, 2020). Preprint available at SSRN: https://ssrn.com/abstract=3696609 or http://dx.doi.org/10.2139/ssrn.3696609
- Pithadia, H. and Treleaven, P., 2020. Blockchain Tokenization (November 07, 2020). Arxiv.org. Preprint available at arxive: https://arxiv.org/submit/3459915/view
- Treleaven, Philip, Pithadia, Hirsh and Smietanka, Malgorzata, Federated Learning (September 25, 2020). [Accepted for IEEE Computer]

List of Grants

- UKRI InnovateUK Grant: RegNet- Privacy preserving data access network for regulated sectors (Application number: 45079)

Commercialisation

- Co-founding RegulAItion [1]- a "RegTech" company that provides solutions and tooling for Algorithmic Regulation

[1]https://regulaition.com/

Acknowledgements

I'd like to offer my sincerest gratitude to my supervisor, Prof. Philip Treleaven, who has supported me throughout my research with his patience, enthusiasm and motivation whilst allowing me the room to work in my own way. I attribute the accomplishments of this research work to his encouragement and effort. One simply could not wish for a better or understanding supervisor. I am very grateful to him for broadening my view of what a the opportunity of a PhD education is- not solely about writing academic papers and presentations, but also engaging and learning from the world, beyond university walls, and more importantly having a visible impact practically and commercially [2].

I'd also like to thank Sally Sfeir-Tait for her continued guidance personally and professionally, she has always believed in my research ideas, its impact and helped materialise it into something meaningful- rather than just abstract ideas. I wouldn't have been able to achieve this work if it wasn't for her patience and understanding.

I would also like to thank my second supervisor Dr. Steven Murdoch and MSc Supervisor Prof. Chris Clack for their invaluable guidance, I wouldn't have considered research as a career without them. I'd like to also thank Andrei Margeloiu- I thoroughly enjoyed our time working together and learnt a lot about programming and systems design from him. I'd also like to thank Bogdan Batrinca who has always been there for advice.

[2] Something Adriano made me realise

This thesis would not be possible without the support and help of EPSRC, CDT Financial Computing, Yonita Carter, Dawn Bailey, Sarah Turnbull and JJ Giwa-Majekodunmi. I thank you for facilitating this research.

I'd like to also thank Mihir Pithadia, Parul Chhaya and Hina Huntington they have always believed in my work and backed me during difficult times. Thank you! It is unthinkable to finish this section without mentioning my parents, Jaykrishnan and Bharti, and my sibling Harikrishna who have always been there- resolute and ever supporting.

Over the last four years I have been able to validate the industrial impact of this research through the help and support of the FCA and Santander and thank them for giving me the opportunity to conduct this research. I would like to also thank everyone at RegulAItion- you have had a large part to play in the implementation of this research.

I'd also like to thank Darshan Soni, Karan Verma and Rishan Shah for their relentless tism. One simply could not wish for a more based collective of goblins.

Looking back, I realise the amount of people that have supported my work and pushed me throughout the last few years to complete this research. I don't think I can mention all of them but I appreciate their kind words, guidance, and advice and would like to mention just a few of them- Rodrigo Mazorra Blanco, Samuel Stern, Timothy Law, Utpal Kanbi, Ravi Majevadia, Pralad Vekaria, Adriano Soares Koshiyma and Emre Kazim. It was Adriano and Emre that pushed me to finally finish!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of this chapter is to present an overview of this thesis by discussing the motivation behind the research problem, the objectives, experiments and contributions of this study and the structure of this thesis. The chapter starts by briefly introducing some background information on the growing problems financial regulators are facing, Distributed Ledgers and Algorithmic Regulation. The chapter then outlines the objectives, experiments and contributions of this work and concludes with the thesis structure.

This thesis explores the use of Artificial Intelligence and Blockchain technology for the purposes of Algorithmic Regulation. Growing concerns about financial regulation and compliance having negative effects on the development of financial services have expressed the need for data-driven technology enforced mechanisms for regulation [79, 5, 6, 76]. To this end the thesis provides a means to capture regulatory rules in machine semantics, the design of a system for regulatory data reporting and a framework for trusted data access for singular algorithmic purposes.

The majority of the work in was conducted in collaboration with the FCA, RegulAItion and Santander, facilitating expert guidance, validation and further confirming the relevance of this study.

1.1 What is Algorithmic Regulation?

The term *Algorithmic Regulation* was first coined by O'Reilly in [63]. Although no concrete definition was provided, O'Reilly provides 4 characteristics of Algorithmic Regulation[1]:

1. A deep understanding of the desired outcome
2. Algorithms (i.e. a set of rules) that make adjustments based on new data
3. Real-time measurement to determine if that outcome is being achieved
4. Periodic, deeper analysis of whether the algorithms themselves are correct and performing as expected.

Regulation refers to the action or process of being regulated [2]. Where the task of regulating involves the control of an activity to achieve a desired outcome/goal. This is done through rules, principles or methods. O'Reilly's first and second characteristics enable this. To do this through computers, where the actions and processes are executed using algorithms, often reactive based on external feedback, is referred to as Algorithmic Regulation- O'Reilly's third and fourth characteristics.

This definition lacks clarity on what the term *algorithm* refers to or why an action needs regulating, we therefore adopt an alternative definition of Algorithmic Regulation. In [89] Yeung et al. define the term *Algorithmic Regulation* as:

> *"...decision-making systems that regulate a domain of activity in order to manage risk or alter behaviour through continual computational generation of knowledge by systemically collecting data (in a real-time, on a continuous basis) emitted directly by numerous dynamic components pertaining to the regulated environment in order to identify and, if necessary, automatically refine (or prompt the refinement of) the system's operations to attain a pre-specified goal."*

[1] Or rather the systems that enable it have
[2] As per the Oxford English Dictionary

More specifically, from the above definition, the terms:

- *"decision-making systems"*- refers to systems that make use of knowledge/ pre-defined rules to execute or inform decisions
- *"pre-specified goal"*- refers to the pre-defined objectives that achieved through the regulation. As such any regulatory system has a 'system' director (or regulator) that defines the purpose and aim of regulation. In the financial services in the UK, this is the FCA whose goal is to regulate the conduct of firms within financial services.
- *"computational generation", "automatically refine"*- refer to the use of intelligent systems such as computers
- *"managing risk"*- refers to the reason for regulating an activity
- *"alter behaviour"*- refers to the direct outcome of implementing regulation

This is a general but more comprehensive definition of *Algorithmic Regulation* and is therefore adopted throughout this thesis. This is different from the term RegTech that refers to *"technologies that may facilitate the delivery of regulatory requirements more efficiently and effectively than existing capabilities"* [5]. Whereas Algorithmic Regulation refers to the use of intelligent systems to manage risk for any processes or systems, the term RegTech is more specific and refers to the application of technology for the automation of particular regulatory processes.

In [89], Yeung also provides a taxonomy of Algorithmic Regulation systems and argues that, broadly, Algorithmic Regulation is of two kinds: reactive or preemptive. Reactive Algorithmic Regulation systems trigger an automated response based on the algorithmic analysis of data. Preemptive Algorithmic Regulation systems act preemptively on the algorithmic assessment of data to infer predictions about future behaviour and act accordingly. Fundamentally both require data to act on or prevent behaviours/ activities that arise from it [3], in order to manage risk. This process is

[3] Which are predominantly based on rules/principles

called compliance and is the instrument through which regulation is carried out.

Automated regulation is crucial to the future success of the financial services industry and especially the rapidly evolving new Financial Technology (FinTech) area [79]. The vision of Algorithmic Regulation, modelled on Algorithmic Trading systems [76], is to stream compliance reports, social media data and other kinds of surveillance information from different sources to a platform where regulatory data are encoded using distributed ledger technology and automatically analysed using AI machine learning technology.

For Regulators, the data science technologies provide, on the one hand, unprecedented volumes of data and analytics tools for surveillance, but on the other 'revolutionary innovations that present new regulatory challenges [5, 79, 76]. To understand the opportunities offered by data-driven regulation, it is necessary to understand the data science technologies contributing to this.

We review the data science technologies into a) data technologies the collection and analysis of huge volumes of historic and real-time information (e.g. financial, economic, social media, alternative); b) algorithm technologies new forms of 'statistics, such as machine learning, computational statistics, and complex systems (e.g. deep neural networks, Monte Carlo simulation); c) analytics technologies covering the application of the data technologies (e.g. natural language processing, sentiment analysis); and d) infrastructure technologies providing the infrastructure for information management and automation (e.g. blockchain, computable regulations), table 1.1.

The research work carried out in this thesis investigates the use of these technologies for regulation. The first investigation investigates the application and development of machine readable semantics of regulation- recognised as the foundation for algorithmic regulation [76]. The second investigation provides the application

Table 1.1: Technologies for Regulatory Automation and their Challenges

	Regulatory Automation	Regulatory Challenges
Data Technologies	• Big data • Internet of Things (IoT) • Machine Readable Rules	• Data privacy versus sharing • Privacy-preserving access to data • ‘data’ subversion
Algorithm Technologies	• Computational Statistics • AI & Machine Learning • Complex Systems	• Combination of multiple models • Algorithm interpretability • Conduct and Ethics in algorithms
Analytics Technologies	• Natural Language Processing (NLP) • Sentiment Analysis • Behavioural Analytics • Predictive Analytics	• Multiple natural languages • `deep understanding of content • Algorithm legal status and certification • Public confidence and acceptability
Infrastructure Technologies	• Blockchain • Digital object identifiers • Federated Learning	• Regulator collaboration • International regulatory standards

of infrastructure and data technologies for regulatory reporting and collaboration; it makes use of a blockchain and machine readable regulation for regulatory reporting. The third investigation provides the application of these infrastructure, data, algorithm and analytics technologies for anti-money-laundering purposes; it makes use of federated learning, blockchains and analytics to detect fraudulent financial transactions.

1.2 Research Motivation

The problem with interpreting regulation

Regulatory handbooks, such as the FCA handbook [14], have been the key means through which regulators interact with firms. Handbooks in essence are the regulatory prescriptions set out by regulators, for firms in the industry, that characterise their conduct of behaviour. Hence all reporting and compliance related operations revolve around the handbook. The FCA handbook is computationally stolid - it has no formal semantics and all regulation is written in text based markup that cannot be used easily by machines to reason about regulation [76]. This makes compliance, registration and reporting related tasks difficult to automate.

The primary barrier to the development of RegTech is not due to technological limitations. It is rather the inability of regulators to process large volumes of financial compliance data that the technology itself generates [83]. The entire financial industry is become increasingly data-centric. Arner et al. [5] actually argue that the industry is experiencing a shift form the traditional

”Know-Your-Client” based policies to ”Know-Your-Data” based policies [4] that regulators find a challenge to accommodate within their existing regulatory frameworks [6]. Consider the FinTech sector for instance, where focus is shifting to the provision financial services using digital money & assets[5] and the monetisation of data[6] prompting the need for regulatory frameworks capable of accommodating algorithmic supervision and data sovereignty[7].

In Treleaven et al. [76] further motivate the need for efficient regulation for the success of the financial services industry, particularly within the FinTech space. To achieve this Treleaven et al. propose the use of “Algorithmic Regulation” [63]. Algorithmic Regulation refers to the automation of financial compliance monitoring and regulation through the use of Artificial Intelligence and Blockchain Technology - a proposal that addresses the growing data-centricity of financial regulation found today[8].

Treleaven et al. have modelled the concept of algorithmic regulation on algorithmic trading systems. It involves the consolidation of compliance data, social networks data and other sources of information to a platform that encodes compliance reports using distributed ledger technology, where regulation is enforced as executable programs in the form of smart contracts [76]. The platform in question would be composed of 5 operational aspects [9]:

1. Intelligent Regulatory Advice
2. Automated Monitoring
3. Automated Reporting
4. Regulatory Policy Monitoring

[4]KYC to KYD

[5]Through Cryptocurrencies and other P2P technologies

[6]Through the application of A.I and Big Data Analytics

[7]Data sovereignty refers to the legal fabric governing the data in question. In some cases, regulators may find that they cannot regulate the activities of companies if their data does not fall under their country’s jurisdiction (e.g. due to geographic boundaries).

[8]A more formal definition has been discussed and used in the next chapter

[9]Discussed in greater detail, chapter 3

5. Automated Regulation

Figure 1.1 provides an overview of the platform and the interdependencies of each component. A long term research objective is to develop a proof of concept system that integrates all the components to carry our algorithmic regulation systemically. We first require a sub-system that is capable of reasoning with regulation. This would be a trivial task if regulation in its current form was expressed as a semantic model. However, therein lies the challenge.

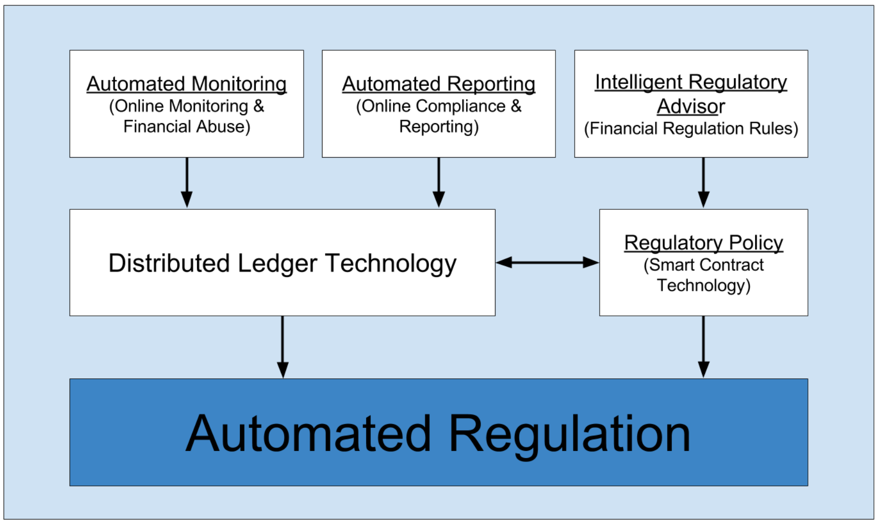

Automated Monitoring
(Online Monitoring & Financial Abuse)
Automated Reporting
(Online Compliance & Reporting)
Intelligent Regulatory Advisor
(Financial Regulation Rules)
Distributed Ledger Technology
Regulatory Policy
(Smart Contract Technology)
Automated Regulation

Figure 1.1: Algorithmic Regulation, adapted from [76]

The project's motivation stems from this long term research objective of a holistic system for algorithmic regulation. The aim of this investigation is to develop a system that is capable of reasoning with regulation via a reasoning engine. This reasoning engine will be tested by the regulator through a platform that enables financial companies to navigate through the handbook to carry out tasks such as license registration. The platform will in turn serve as the Intelligent Regulatory Advisor component noted in figure 1.1 and [76].

The problem of reporting data for regulation

In order to supervise firms, regulators need to examine the practices of institutions they regulate. All regulated firms are required to submit data of their practice through regulatory reports. Whereas handbooks, such as the FCA Handbook, provide the instructions of how regulatory reports should be built and delivered, the reports are explicit instructions of what fields of data need to be submitted for supervision. The level of regulation and regulatory reporting has significantly increased since the financial crisis [5].

Given this increase in reporting requirements, the complexity and time it takes for financial institutions to manage compliance reporting has also grown. Moreover regulators make ad-hoc data requests over and above routine reports submitted by regulated firms. The only way most firms have managed to scale this increase in requirements and complexity is by increasing the headcount within their compliance departments.

There are a number of reasons why the process of supplying regulatory reports is increasingly complex [12]. The process of building reports from the FCA Handbook is difficult and open to interpretation by legal departments in [79]. Moreover the entire set of instructions for compiling a report can be spread across many different areas of interlinking regulation. At times, the wording found in the Handbook is insufficient or unclear for firms to understand. On the other side, regulators struggle to provide precise reporting instructions for about 50,000 firms that operate across financial services [12] [13]. Often times firms need to make judgements based on their practice that makes it difficult to provide unambiguous, definitive requirements. Additionally, firms could be operating across several jurisdictions, forcing them to repeat the reporting process in multiple regimes and jurisdictions. This repetition is often across the same data sets with similar requirements.

The second motivation stems from this problem of reporting data, can we architect an automated mechanism to report data to multiple regulators without the trusted intermediation?

Data regulation compliance and collaboration

The debate around Data Regulation is amplifying and the need for policies and mechanisms to manage data compliance, governance and regulation is growing. There are growing political, commercial and social challenges concerning data that have resulted in the need for the Algorithmic Regulation of data [79, 89, 77, 86].

Whereas collaboration, ownership and harvesting are all challenges that deal with Data Management (including analytics, application and administration of data) [77]. Legislation and sovereignty are challenges deal with Data Governance. Security and privacy challenges deal with the assurance and safekeeping of data. All three factors: Data Management, Governance and Security need to be considered when regulating data [89]. The fundamental research question of this chapter is based on the premise of all these three factors: How can we better manage and govern data in a secure, privacy preserving manner for algorithmic purposes? The third research motivation addresses this problem.

1.3 Research Objectives

The motivation of this thesis is to investigate the application of blockchain technology and artificial intelligence to the domain of Algorithmic Regulation. The intention is to explore the problems with regulatory compliance and supervision in financial services as well as an emerging form of regulation- the regulation of data, with close industry collaboration. The understanding and findings from these will then be used to architect Algorithmic Regulation solutions, which could realistically be adopted by industry, that automate regulation and compliance. All the work has been validated by industry partners and collaborators. To this end, the thesis explores 3 investigations:

1. Using AI to Automate the Regulatory Handbook

2. SmartReg: Using Blockchain for Regulatory Reporting

3. RegNet: Using Federated Learning and Blockchain for Privacy Preserving Data Access

Using AI to Automate the Regulatory Handbook

This study has a number of sub-objectives:

- Study the semantic structure of regulatory rules and how they can be encoded as machine readable rules
- Propose a formal specification for a regulatory reasoning system
- Implement the design of this reasoning engine for use by the regulator as a recommender system
- Assess the suitability and accuracy of automated guidance from this system

SmartReg: Using Blockchain for Regulatory Reporting

The following objectives were established:

- Identify problems with regulatory reporting currently from the perspective of each entity ()
- Establish systems design constraints based on these
- Propose a solution architecture that addresses these design constraints
- Implement, analyse and iterate the proposed solution
- Test the solution in the“real world” with an actual reporting use case pilot
- Analyse and compare the proposed solution to the current system

RegNet: A Framework for Privacy Preserving Data Access

The following objectives were established:

- Identify relevant technologies and fields of research in data science and information security to support trusted data access for a singular algorithmic purpose

- Architect a framework based on these technologies that facilitates data access for industry applications
- Implement, analyse and iterate over the design of this framework and its emerging infrastructure
- Test and validate this infrastructure with the collaborators
- Analyse the proposed solution and its efficacy in governing data in a secure, compliant manner for algorithmic purposes

1.4 Research Methodology

All three studies involve the development of Algorithmic Regulation systems, therefore a Domain Driven Design [66] approach has been used to architect the systems. The goal/requirements of each system have been the primary focus upon which all architectural decisions have been made. These have been informed by the relevant industrial collaborators and the resulting systems have been implemented and tested through their application to a number of use cases provided by the collaborators. Once these were established, an agile methodology has been used for implementation, testing and iteration.

More specifically, the development, implementation and testing of each system has involved the following:

1. **Using AI to Automate the Regulatory Handbook**
    - Decomposition of regulatory rules for graph based rule representation
    - Formal design of the engine, knowledge store and rules representation
    - System Implementation, benchmarks and improvements
    - Application of a backtracking algorithm and comparisons
2. **SmartReg: Using Blockchain for Regulatory Reporting**
    - Standardise of reporting firms' data through open interfacing

- Develop machine readable and executable reporting rules from the Handbook
- Develop a standardised automated reporting mechanism

3. **RegNet: Federated Learning and Blockchain for Privacy Preserving Data Access**

    - Development of Federated Data Infrastructure: privacy-preserving data infrastructure and framework for collaboration that allows secure communication with collaborating parties, such that 'raw' data does not leave the owner.
    - Development of Federate Machine Learning algorithms for relevant use cases: decentralised training of a machine learning models that enable collaborative learning while keeping data sources in their original location

1.5 Scientific and Industry Contributions

Although discussed in detail in their relevant chapters the impact and contributions of the work undertaken in this thesis is as follows:

1.5.1 Machine Interpretability of Regulation

The work in capturing regulatory semantics has made the following contributions:

1. A semi-formal model capable of reasoning with financial regulation has been developed. The model has been used to represent regulation from the FCA Regulatory Handbook, proving that it is capable of representing and reasoning with financial regulation.

2. The model has been developed into a working reasoning engine

3. The reasoning engine has been incorporated into a platform that is capable integrating existing regulatory systems or be used as a standalone tool for regulatory advice

The work in this project is part of a long-term goal and collaboration with the FCA to deliver a system capable of reasoning with regulation. The work provides the groundwork needed to realise this goal through the development of a system capable of reasoning with regulation. The proof-of-concept, based on this work, has also been used by the FCA for their internal regulatory guidance for providing licenses. The work presented in this chapter has also fed into a Master's dissertation on reasoning with regulation.

1.5.2 SmartReg: Using Blockchain for Regulatory Reporting

The design decisions made in SmartReg have informed a wider project carried out by the regulator and a number of banks. This is the Digital Regulatory Reporting Project (DRR). DRR has become a priority for the regulator, banks and the UK financial services industry [12, 13]. This has been the greatest impact of the work presented in this chapter. The work in SmartReg has wide reaching impact through the following benefits of its approach:

1. Easier regulatory compliance
2. More precise and agile regulation
3. Accurate and real-time information
4. Standardisation
5. Improved systemic risk control

1.5.3 RegNet: Using Federated Learning and Blockchain for Privacy Preserving Data Access

Through the use of Federated Learning and Tokenization, RegNet, removes the need to explicitly share data, for the purposes of analytics and provides access in an auditable, privacy preserving manner that is able comply with data policies such as GDPR. Tokenization also offers the ability to provide a means to administer data governance, particularly at the channel level. The work undertaken within this chapter proves the use of both technologies for trusted data access (also validated by its application on an applied use case within industry).

The need for such a trusted infrastructure that is compliant with data regulation has been further strengthened by a report released recently by the UK AI Council. It recognises that AI advances with diverse data and stresses the need for trustworthy accessible data, the work undertaken in RegNet provides such a solution.

1.6 Thesis Structure

This thesis is structured as follows:

- **Chapter 2** reviews background information on algorithmic regulation, algorithmic regulation in the financial services, financial information systems and the regulatory challenges that the industry faces. It provides a technical overview of financial regulations and distributed ledgers (Ethereum and Quorum in particular).

- **Chapter 3** introduces the design of an engine capable of reasoning with financial regulation that has been co-developed and tested with the FCA. It includes all the architectural decisions taken to address problems such as parallelisation and system downtime in order to serve in a production grade environment within the FCA. The chapter then ends with a discussion on the objectives and outcomes of the work carried out.

- **Chapter 4** presents the design for a system to carry out automated compliance and reporting through distributed ledgers. The chapter explores the use of Smart Contracts for regulatory reporting and the automation and implementation of reporting rules. The chapter also explores the design decisions taken in implementing the blockchain infrastructure. The chapter then ends with an evaluation of the results as well as the outcomes of the work undertaken.

- **Chapter 5** begins with a discussion on Federated Machine Learning, Privacy Preserving cryptography and the need for trusted data access. The chapter then presents the design of a framework for trusted data access for singular algorithmic purposes and applies it to an Anti-Money-Laundering use case. The chapter ends with a discussion on the results and the future work that needs to be carried out.

- **Chapter 6** provides an overall conclusion of this research with the summary of the key findings and their impact in Algorithmic Regulation. The thesis ends with recommendations for future work to be done in this area.

Chapter 2

Background and Literature Review

This chapter begins with a discussion an overview of Algorithmic Regulation in the Financial Services. The chapter then provides a detailed discussion of the systems and motivations for Algorithmic Regulation in Financial Services and concludes with a brief technical background on Blockchain Technology, Knowledge Base Systems and regulation within the UK Financial Services.

2.1 Algorithmic Regulation in Financial Services

In the financial services in the UK, regulation is a reactive activity and as such relies on compliance and monitoring to enforce regulation- banks/ financial institutions have to submit data to regulators that may choose to act (or rather "react") accordingly [79, 5, 11]. There is a growing concern about regulation and compliance, which is increasingly perceived to have negative effects on the development of financial services, discouraging innovation by requiring an ever-growing amount of data reporting [17, 6].

Financial regulation is becoming increasingly burdensome. Research suggests that as of July 2016 U.S. banks had paid U.S.$24 bln and 61 million employee hours to comply with Dodd-Frank Wall Street Reform and Consumer Protection Act, passed in the U.S. amid outcry over the financial crisis [89, 5, 70]. This was just one single regulatory reform act. Regulation constantly evolves to address new risks and prevent new harms that emerge in the industry.

Additionally, financial regulation faces a myriad of pressures: political pressure to curb excesses (e.g., Libor); escalating international and European Union regulations (e.g., MiFID II); individual firms simultaneously regulated in multiple jurisdictions and with frameworks; and institutions asked to produce increasing amounts of financial, risk, and compliance data [79]. All this pressure has generated the negative perception that data is being requested speculatively and not being used by the regulators. The challenge is to simplify and balance regulation while encouraging innovation for new FinTech alternative finance entrants, in rapidly changing environments [79] and as such [76], Treleaven et al. argue the need to overcome this impasse requires radical automation.

In [76] Treleaven et al. motivate and identify five potential solutions for collectively realising Algorithmic Regulation in the financial services. These involve:

1. **Intelligent Regulatory Advisor**
   Financial companies, especially FinTech startups[1] find it extremely difficult and expensive to navigate through the regulatory landscape because of the complexity of the regulatory handbook and registration processes. To overcome this hurdle, Treleaven et al. propose the creation of an artificially intelligent front-end to the handbook that guides users through regulation, identifying the aspects relevant to them and assists them through the registration process.

2. **Automated Monitoring**
   Regulators, such as the FCA, have experienced an explosion in the number of firms that they have to monitor. The FCA was previously responsible for monitoring about 25,000 firms. However, with the growth of FinTech and other developments in the industry, the FCA is faced with the dilemma of regulating an additional 21,000 firms with the same amount of resources. A solution to automate this involves the use of data scraping and sentiment anal-

[1]Often very innovative, yet underfunded

ysis tools previously employed in the retail industry [2].

3. **Automated Reporting**
   Treleaven et.al argue that an opportunity exists in the development of an automated light-weight solution for regulatory compliance reporting through the use of online analytics software techniques. However, three key requirements need to be in place:

   (a) A Reporting Language: A mark-up based reporting standard such as the ISO 20022 standard.

   (b) A Reporting Platform: A light-weight client side platform capable of interfacing with standard accounting systems.

   (c) Regulatory Analytics: For use in Anti-Money-Laundering (AML) and Know-Your-Customer (KYC) based compliance and reporting polices.

4. **Regulatory Policy Modelling**
   Developments in computational models such as Agent Based Modelling within sandboxed environments can be used to assess the impact and effectiveness of new regulatory policies. This would enable the development of more effective, less systemically risky policies.

5. **Automated Regulation**
   Involves the development of a system where blockchain and distributed ledger technology are used for recording compliance reports and smart contracts are used to encode regulations as self-enforcing computer programs.

In order to facilitate Algorithmic regulation, Treleaven et al. argue that Blockchain Technology and Smart Contracts are key [76]. Smart Contracts, by definition, refer to an agreement whose execution is both automatable and enforceable. Automatable by computer, although some parts may require human input and control. Enforceable by either legal enforcement of rights and obligations or tamper-proof execution [22]. Blockchains on the other hand are distributed ledgers

[2]For brand management and customer profiling e.g. Google Alerts and Brandwatch

where transactions and state changes of smart contracts[3] are kept in a shared, replicated, synchronised record that is secured through the use of public key cryptography [59, 55]. Its integrity is preserved through decentralisation and the use of consensus mechanisms [4] that make it difficult to alter the record [59].

Smart Contracts[5] and Blockchain Technology would be used for Automated Reporting, Regulatory Policy Modelling and Automated Regulation. In the case of Automated Reporting, a reporting standard is key. All compliance reports would be filed into a distributed ledger. Public key infrastructure and other crypto-systems can be used to manage the data within the ledger (e.g. to authorise viewership/assessment of the compliance data). The compliance reports would however need to be verified using a system that is capable of reasoning with regulation before being written onto the immutable ledger.

Similarly, automated regulatory policy assessment can be achieved through the use of Smart Contracts and Testnets. Testnets are sandboxed environments that are used to test new Smart Contracts (i.e. new regulatory policies) within a blockchain system. However, for effectiveness, a substantial amount of regulation will need to be encoded as Smart Contracts before agent-based models can be used to analyse the impact and effectiveness of proposed regulatory policy.

Automated Regulation would then use automated Monitoring and Reporting, with Analytics and Smart Contract based analytics to provide a platform capable of Algorithmic Regulation as noted in figure 2.1.

[3]e.g. Ethereum

[4]such as Proof-of-Work and Proof-of-Stake

[5]It should be strongly noted that not all aspects of regulation can be recorded as Smart Contracts, sub-chapter 2.2.1 explains why.

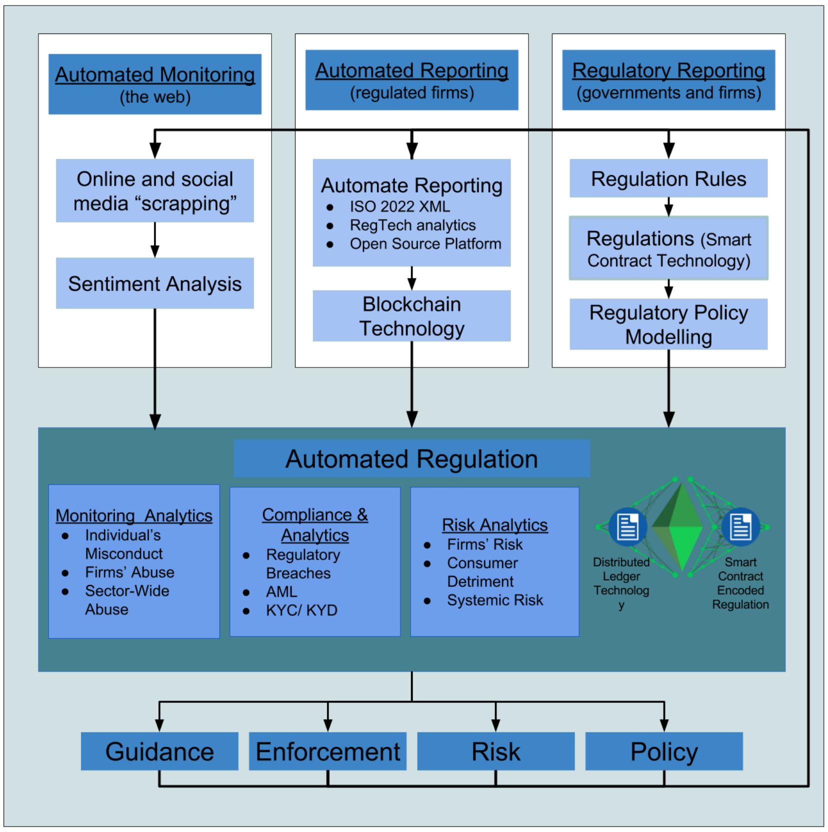


Figure 2.1: Algorithmic Regulation using blockchain technology, adapted from [76]

2.2 Technical Background

This sub chapter presents some of the general background. Each experiment provides a more detailed section of its relevant technical background.

2.2.1 Rules and Principles

According to Brummer et.al [17], there exist two categories for characterising regulation: principle-based regulation and rules-based regulation. Rules-based regulation is prescriptive, precise and specific. Principles based regulation on the other hand is broad, goal-oriented and outcome focused. It considers context and circumstances rather than being definitive and decided.

However, neither can exist in isolation, rules need to be derived from principles and principles are meaningless without rules. Tables 2.1 and 2.2 provide a detailed analysis of both. Both factors need to be considered in the design of a system capa-

Table 2.1: Rule-based Regulatory Regimes, adapted from [17]

Rules-based Regulation	
Benefits	**Limitations**
Certainty and predictability, including with respect to future enforcement	Check-the-box forms of compliance that strategically evade the underlying purpose of regulation
Clear communication of steps for compliance	High internal costs of compliance
Ensures specific behaviour	Deterrence with respect to innovation
Uniform treatment of regulated entities	Frequent disconnect between the purpose of regulation and the actual regulatory outcomes
	Obsolescence

Table 2.2: Principles based Regulatory Regimes, adapted from [17]

Principles based Regulation	
Benefits	**Limitations**
Executive level management involvement in incorporating regulatory principles into business models	Uncertainty and the risk of unpredictable post hoc application or arbitrage
Flexibility and innovation in the face of rapidly changing environments	Concerns over fairness. bias in application
Speed in the regulatory process	Inadequate deterrence of specific problematic behaviour or activities
The centrality of guidance and evolving norms/best practices	Over-reliance on the current norms and practices

ble of reasoning with regulation. Rules-based reasoning systems can be developed using traditional expert systems. However, encoding principle-based reasoning is a complex and open ended research question. Partial principle-based reasoning is possible through a combination of expert systems with backward chaining algorithms [6] and suitable rules-conflict resolution mechanisms. However, this requires a large amount of rules readily available in the system as a pre-requisite. Therefore the key is to design a rule-based system capable of both backward and forward chaining reasoning.

Rulemaking therefore is crucial. Brummer et.al [17] suggest the use of "agile and iterative" rulemaking. They argue that while innovation, management and markets have evolved as technology has evolved, regulation has not. This sentiment is consistent with a number of literatures [76, 5, 50, 6]. They therefore suggest an

[6]Discussed in greater detail in chapter 3

iterative results driven rule making process through the constant review of outcomes and regulatory goals via feedback loops.

2.2.2 Expert Systems

Expert systems are rule-based computer programs that capture the knowledge of human experts in their own fields of expertise [45]. Early, successful expert systems were built around rules (sometimes called heuristics) for medical diagnosis, engineering, chemistry, and computer sales [54]. One of the early expert system successes was MYCIN, a program for diagnosing bacterial infections of the blood. Expert systems had a number of perceived advantages over human experts [54]. For instance, unlike people, they could perform at peak efficiency, 24 hours a day, forever.

2.2.2.1 Expert System Architecture

The rules in the first expert systems were implemented directly with the rest of the software, so that developing a new expert system meant starting from the ground up. The developers who wrote MYCIN, recognising this fact, created a development tool named EMYCIN. EMYCIN (Empty MYCIN) was developed by removing all the medical knowledge from MYCIN, leaving behind only a generic framework for rule-based systems. EMYCIN was the first expert system shell. An expert system shell is just the inference engine and other functional parts of an expert system with all the domain-specific knowledge removed. Typically an expert system is composed of [35]

1. An Inference Engine
2. Rule Base
3. A Working Memory

An inference engine, in turn is composed of:

1. A Pattern Matcher
2. An Agenda

3. An Execution Engine

Figure 2.2 provides an architectural overview of an expert system.

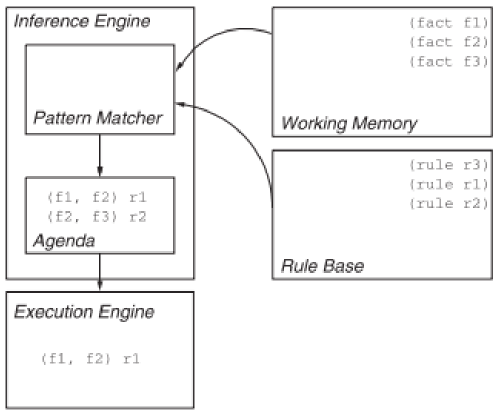


Figure 2.2: The architectural overview of an Expert System [35]

Inference Engine

The inference engine controls the whole process of applying the rules to the working memory to obtain the outputs of the system. Usually an inference engine works in discrete cycles based on the following sequence [35]:

1. All the rules are compared to working memory (using the pattern matcher) to decide which ones should be activated during this cycle. This unordered list of activated rules, together with any other rules activated in previous cycles, is called the conflict set.
2. The conflict set is ordered to form the agenda - the list of rules whose right-hand sides will be executed, or fired. The process of ordering the agenda is called conflict resolution. The conflict resolution strategy for a given rule engine will depend on many factors, only some of which will be under the programmers control.
3. To complete the cycle, the first rule on the agenda is fired (possibly changing the working memory) and the entire process is repeated. This repetition implies a large amount of redundant work, but many rule engines use sophisticated techniques to avoid most or all of the redundancy. In particular,

results from the pattern matcher and from the agendas conflict resolver can be preserved across cycles, so that only the essential, new work needs to be done

Rule Base

The rule base contains all the rules the system knows. They may simply be stored as strings of text, but most often a rule compiler processes them into some form that the inference engine can work with more efficiently. For an email filter, the rule compiler might produce tables of patterns to search for and folders to file messages in. Some compilers build a complex indexed data structure called a Rete Network. A Rete network allows rule-based systems to process rules very efficiently via backward chaining [36].

Some rule engines allow (or require) one to store the rule base in an external relational database, and others have an integrated rule base. Storing rules in a relational database allows one to select rules to be included in a system based on criteria like date, time, and user access rights.

Working Memory

In a typical rule engine, the working memory, sometimes called the fact base, contains all the pieces of information the rule-based system is working with [45]. The working memory can hold both the premises and the conclusions of the rules [45]. Typically, the rule engine maintains one or more indexes, similar to those used in relational databases, to make searching the working memory a very fast operation. Some working memories can hold only objects of a specific type, and others can include, for example, Java objects. This provides deeper embedability of the shell with the language/platform.

2.2.2.2 Historic Use of Expert Systems in Regulation

An example of the use of Expert systems in regulation and legislation is the work carried out by Van Engers er al. in the EU. funded Estrella Project [78]. The Estrella project involved the development of a “Legal Knowledge Interchange Format

- LKIF" that made use of expert systems for legislation.

The outcome from the project made use of the Legal Knowledge Interchange Format to model European Tax Legislation which was then trialled by a number of European Tax administrations. This is a close parallel to the work carried out in this thesis.

2.2.3 Expert System Shell Analysis

Given that expert system have been in use since the early 1970s, there exist a large number of shells. However, not all are alike, they could be written in different languages, have different inference algorithms and may not support forward or backward chaining. A survey was carried out to analyse the relative merits of 5 expert systems shells. These 5 shells were identified through their popularity and use on GitHub.

The properties and abilities of each shell were analysed based on the following criteria:

1. Embeddability: How deeply can the shell be embedded with the development language; can it interact with classes and external libraries easily
2. Forward Chaining: Whether the shell supports forward chaining or not
3. Backward Chaining: Whether the shell supports backward chaining or not
4. Extensibility: Can parts of the shell's existing code base be edited
5. Open Source: Is the shell open source or does it require licenses
6. Development Tools: Whether there are development tools such as IDE extensions available for use with the shell
7. Developer Support: Whether there is adequate support for the shell, either on GitHub, StackOverflow or their personal forums

As it can be seen from table 2.3, the Java Expert System Shell is the only shell that satisfies most criteria. The only criteria that fails to support is the open source criteria. Jess however can be used freely for academic purposes, without the explicit need for a license. Moreover, Jess is the only shell that supports both forward and

Table 2.3: Expert System Shell Survey

Shell	**Properties**							
	Language	Developer Tools	Developer Support	Extensibility	Forward Chaining	Backward Chaining	Embedability	Open Source
PyClips	Python	Y	Y	N	Y	N	Y	N
Jess	Java	Y	Y	Y	Y	Y	Y	N
Clips	CLIPS (lisp like)	Y	Y	Y	Y	N	N	Y
Pyke	Python	N	N	Y	Y	N	Y	Y
Eresye	ErLang	N	Y	Y	Y	N	Y	Y

backward chaining, therefore Jess is the shell that has been chosen to bootstrap development.

2.2.4 A formal model for expressing expert systems

An expert system is fundamentally composed of an Inference Engine, Working Memory and Knowledge Base as seen from figure 2.3 below.

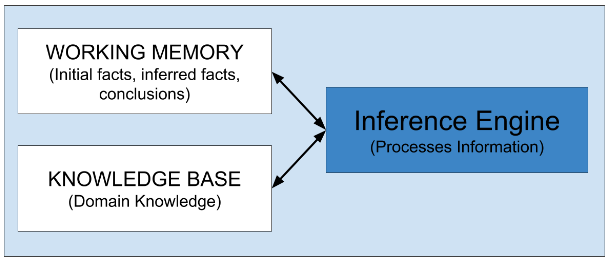


Figure 2.3: Core Components of a Knowledge Based System [36]

The formal model developed by Gamble et al. is based on the architecture seen in figure 2.3. It is composed of a set of formal models that define the Schema of each component. The Schema[7] is the “blueprint” that defines the component with attributes that it must have.

The model is “Object-State-Step” based. This means that the model defines the static properties of the components at each step - the model is therefore state based and can be viewed as a form of state machine. Therefore, alongside component Schemas, State Schemas also need to be defined formally. These collectively form transitions from one legal state to another.

The model has been defined using Z notation [75, 36]. A set of the Z notation

[7]For intuition, think of this as a class as perceived in an object oriented language such as Java.

conventions used in this dissertation has been provided in Appendix A. It is assumed that the reader is comfortable with basic logic and notation. A formal model for all three components has been provides, we begin with the definition of the schema for the component, followed by formal definitions for their state and step-changes in state.

A formal model for the controller that that acts as the data interface across all three components has not been provided. This is because we are interested in the high-level representation of knowledge and reasoning rather than the lower-level control flow. Gamble et al. however provide a formal model for the controller(s) and can be studied in greater detail in [36].

2.2.4.1 Preamble

The premise of a knowledge base system (KBS) is to capture expertise in the domain it is employed towards. Rules form the building blocks of KBS systems. In essence, Rules possess the form:

$$LHS \rightarrow RHS \tag{2.1}$$

The LHS (Left-Hand-Side) refers to the list of conditions that need to be fulfilled. The RHS (Right-Hand-Side) refers to the actions that need to be carried out, given that the LHS conditions are met.

Type Declarations

Consider the following types:

1. FACT

   A FACT type is one of the basic types, these are statements that are perceived to be true
2. RID

   A RID (Rule ID) is a unique identifier (e.g. a hash) for a rule component

3. ACTION

   An ACTION type refers to the changes from the rules

4. TEMPLATE

   A TEMPLATE is a type synonym for the left hand side of the rule

5. PRECONDITION

   A PRECONDTION type is a relation between a rule and the TEMPLATE of conditions for of the rule. The TEMPLATE refers to the “IF” part of the RULE. More precisely a PRECONDITION is:

   $$\text{RID} \leftrightarrow \text{TEMPLATE}$$

6. RULE-INSTS

   The RULE-INSTS type is a relational type that matches a rule with the FACTs in the WORKING MEMORY. Which results in the firing of a rule through rule instantiation. It is defined as:

   $$\text{RID} \leftrightarrow \textbf{set}\ \text{FACT}$$

User Defined Functions

Consider the following user defined functions, their purpose and definitions:

1. **update** function

   The purpose of the update function is to “update” the WORKING MEMORY. It takes and ACTION-FACT relation[8] (that is obtained from a rule), applies it to the WORKING MEMORY and returns an updated WORKING MEMORY. It is defined as:

   $$\textbf{update} ::= ((ACTION \leftrightarrow FACT) \times \textbf{set}\ FACT) \nrightarrow \textbf{set}\ FACT \qquad (2.2)$$

2. **unifedset** function

   This purpose of the “unifiedset” function is to produce new changes for the Inference Engine. The unified function matches a Rule Instantiaition with

[8]An ACTION-FACT pair simply refers to an activated rule

the PRECONDITION that may cause it to fire. It then " activates" the rule by returning an ACTION-FACT pair.

$$\textbf{unifiedset} ::= (RULE-INSTS \times \textbf{set}\, PRECONDITION) \nrightarrow (ACTION \leftrightarrow FACT) \tag{2.3}$$

3. **select** function

   The purpose of the select function is to choose a a rule instance from a set of rule instances that all satisfy the conditions in the WORKING MEMORY. It does this based on a STRATEGY[9]. The select function selects RULE-INSTS based on the given STRATEGY(therefore by implication selects which rule to fire when used alongside **unifiedset**).

$$\textbf{select} ::= (RULE-INSTS \times STRATEGY) \nrightarrow RULE-INSTS \tag{2.4}$$

4. **match** function

   The purpose of the match function is to pattern match a PRECONDITION to the set of FACTS (that collectively form the WORKING MEMORY). This in turn instantiates new Rules (i.e. RULE-INSTS), based on the match predicate.

$$\textbf{match} ::= (\textbf{set}\, FACT \times \textbf{set}\, PRECONDITION) \nrightarrow RULE-INSTS \tag{2.5}$$

The subsequent sub-chapters present the definition for each component in figure 2.3. Each component is composed of 3 specifications. The first is a schema specification for the component; it provides the "blueprint" for the component that defines its attributes. The second is a state specification that provides the schema for the modelling state. The third is the schema for the step change in state.

[9]For simplicity, we don't provide a model for STRATEGY in this discussion. It does not have to be a complex strategy it could, for instance, be based on salience which would entail providing a flagged number that signifies rule precedence.

2.2.4.2 Inference Engine

Component Schema

For intuition, consider the Inference Engine (IE) as a broker that exchanges inputs and outputs; it performs matches between the working memory and knowledge base (where the rule base resides). It collects the incoming rule *changes* from a rule (when fired) and processes them based on the current state of the working memory. These incoming inputs can be characterised as action-fact pairs as seen in the definition table 2.4. Action-fact pairs are a cartesian cross product between actions and facts, meaning that they represent the combination of all values of facts and actions. Similarly, the outgoing data is characterised through *rule_insts*. These are explicitly defined variables that store matched preconditions.

Table 2.4: Inference Engine Schema Specification

actions: **set** ACTION	*All actions*
fact: **set** FACT	*All facts*
changes: ACTION $\leftrightarrow$ FACT	*All incoming changes; action-fact relations*
rule_insts: RID $\leftrightarrow$ **set** FACT	*All outgoing rule instances*
states: **set** STATE	*Valid states*
start: STATE	*Starting state*
transition: (STATE $\times$ (ACTION $\leftrightarrow$ FACT)) $\leftrightarrow$(STATE $\times$ RULE-INSTS)	*Valid transition*
start $\in$ states	*Start state is valid*
changes $\subseteq$ actions $\times$ facts	*Incoming changes are valid*
($\cup$ **ran** rule_insts) $\subseteq$ facts	*Outgoing data is valid*
$\forall$ s_1, s_2: STATE; ch:ACTION $\leftrightarrow$ FACT; ri: RULE-INSTS | ((s_1,ch)(s_2, ri)) $\in$ transitions $\bullet$ s_1, s_2 $\wedge$ch $\subseteq$ changes $\wedge$ ri $\subseteq$ rule_inst	*Given the current state and incoming changes, a new state and the rule instances are produced*

The legal behaviour for the inference engine state machine is characterised through three variables: *state*, *start* state and *transitions*. The states consider all possible states that the Inference Engine can encounter. The start state provides the bearing for where to begin. The transition is essentially a function that takes an action-fact pair and the internal state. A new state and new rules are instantiated based on these. It should be noted that as per Z notation, these are declared above the midline as observed in definition table 2.4. This is because they represent all the variables and identifiers pertinent to the IE component.

The items below the midline are the properties that must hold for the entire definition to hold. Thus from the definition table 2.4 it can be observed that the following needs to hold true:

1. The start state should be valid
2. The incoming and outgoing data should be valid
3. The transition function needs to perform the matching and selection such that it is relevant to the rule instances, current state and incoming changes.

State Schema

The purpose of the state schema is to keep track of the IE's current state (*curstate*), the incoming state (*instate*) and the outgoing state (*outstate*). As seen from the portion below the midline in the table definition 2.5, the *curstate* is composed of a tri-tuple of:

1. The local variable represented by *state_vars*
2. The Knowledge-Base working memory represented by *work_mem*
3. The precondition of the rule represented by *precond*

In fact the IE state is just a cartesian cross product of the above three types.

Table 2.5: Inference Engine State Specification

ie: IE_COMPONENT	*Given instance of the inference engine*
curstate: STATE	*The current state of the inference engine*
state_vars: IESTATE	*Local variables of the state, is actually part of the current state*
work_mem: **set** FACT	*Current working memory*
precond: **set** PRECONDITION	*The rule preconditions*
instate: ACTION $\leftrightarrow$ FACT	*Current input data*
outstate: RULE-INSTS	*The data on the output*
curstate=(state_vars, work_mem, precond)	
curstate $\in$ ie.states	*The current state is valid*
work_mem $\subseteq$ ie.facts	*The working memory is valid*
instate $\subseteq$ ie.changes	*The changes provide the input data*
outstate $\subseteq$ ie.rule_inst	*Outgoing data is in the outgoing collection variable*
($\cup$ **ran** outstate) $\subseteq$ work_mem	*Outgoing data is a susbset of the working memory*
instate $\neq \emptyset \Rightarrow$ outstate $= \emptyset$	
outstate $\neq \emptyset \Rightarrow$ instate $= \emptyset$	*At most, only one of instate or outstate can hold data*

Now consider the constraints that need to be held for the state schema to hold [10]. A

[10]Recall that this is the portion below the midline of table definition 2.5, as per the convention in Z notation.

valid *curstate* must be present within the state instance of the IE instance [11]. The constraints for the input data that needs to be processed is represented by *instate*. Similarly, the output data is constrained through the *outstate* constraints seen in the table definition 2.5.

Given that a rule-based system computes in discrete cycles [12], there can only be data either in the the *instate* or *outstate*. This constraint is reflected in the last definition of table 2.5.

Inference Engine State Step Change Schema

The step schema (i.e. the IE state transition schema), is defined in the table definition 2.6. We are only considering the change in the IEState (hence the use of $\triangle$ symbol). Therefore, for a valid change to hold, all the constraints pertinent to the IEState should hold before and after the transition. These are the state invariants. However, we have undergone a computation therefore something must change. This change is characterised through the removal of the action-fact tuple from the instate and the addition of new information into the outstate.

Table 2.6: Inference Engine Step Specification

$\triangle$ IEState	*Given a step state change in the Inference Engine*
ie' = ie	*Constraint information does not change*
$\exists$ out: RULE-INSTS $\bullet$ ((curstate, instate), (curstate', out)) $\in$ ie.transitions $\wedge$ instate $\neq \emptyset \wedge$ work_mem' = update(instate,work_mem) $\wedge$ out = match(work_mem', precond) $\wedge$ instate' $= \emptyset \wedge$ outstate' = outstate $\cup$ out	*Given a valid current state and valid input, produce a valid output with the update and match functions. Remove the input data form the instate, and add the new data to the output*

2.2.4.3 Knowledge Base (Rule Repository)

Component Schema

Given that rules are fired based upon a given decision, we require unique identifiers in the form of *rulename* as observe in the table definition 2.7. Likewise, *facts* refer to all the facts refer to all the facts that appear as inputs [13] and cause

[11]The dot notation *ie.states* binds the state definition to the component definition

[12]e.g. using the Rete Algorithm [35]

[13]These come from the working memory

rules to to instantiate. Recall that the purpose of of rules is to make changes if the given conditions are satisfied. These conditions are denoted on the Right-Hand-Side (RHS) portion of the rule 2.1.

These changes are defined in the *changes* variables. As before, changes are essentially just a cartesian cross product between actions & facts. The rule transitions through a mapping of the rule's internal state and the value of *insts*, into a new internal state which becomes the outgoing data. The transition is constrained by the incoming insts and the outgoing changes as seen in the definition table 2.7.

Table 2.7: Rule Component Schema Specification

rulename: RID	*Rule Name*
insts: **set** FACT	*All facts triggered from the instantiation of rule(i.e. incoming data)*
actions: **set** ACTION	*Changes (i.e. outgoing data)*
changes:ACTION $\leftrightarrow$ FACT	*All actions pertinent to the given rule*
facts: **set** FACT	*All facts pertinent to the given rule*
states: **set** STATE	*Vaild states*
start: STATE	*Starting state*
transition: (STATE $\times$ **set** FACT) $\leftrightarrow$ (STATE $\times$(ACTION $\leftrightarrow$ FACT))	*Valid transitions*
start $\in$ states	*Start state is valid*
changes $\subseteq$ actions $\times$ facts	*Incoming changes are valid*
insts $\subseteq$ facts	*Outgoing data is valid*
$\forall$ s_1, s_2: STATE; ri: **set** FACT; ch: ACTION $\leftrightarrow$ FACT | ((s_1,ri) (s_2, ch)) $\in$ transitions $\bullet$ s_1, s_2 $\in$ states $\wedge$ ri $\subseteq$ insts $\wedge$ ch $\subseteq$ changes	*Given the current state, rule instantiation, and the set of preconditions a new state with action facts(changes) are produced*

State Schema

Just like the Inference Engine, an instance of the component needs to first be created as it is this that must possess the state. In this case we create a rule component instance. The purpose of a rule state schema is to assert, at any given point in time, that the rule is fully consistent with its internal state, incoming data and outgoing data.

The variable *curstate* (i.e. the state of the rule) is constrained as a cartesian cross-product of the current state and precondition[14].

[14]For simplicity it is presented as tuple in the definition table 2.8

Table 2.8: Rule State Schema Specification

r:Rule_Component	*Given the instance of the rule*
curstate: STATE	*Current internal state*
state_vars: RSTATE	*Local variable of the state*
precond: PRECONDITION	*The rule precondition*
instate: **set** FACT	*Current input data*
outstate: ACTION $\leftrightarrow$ FACT	*The data on the output*
curstate = (state_vars, precond)	*The current state is the current state info.*
precond.1 = r.rulename	*The precondition is valid*
curstate $\in$ r.states	*The current state is valid w.r.t the rule*
instate $\subseteq$ r.insts	*Incoming data is valid*
outstate $\subseteq$ r.changes	*Outgoing data is valid*
instate $\neq \emptyset \Rightarrow$ outstate $= \emptyset$	
outstate $\neq \emptyset \Rightarrow$ instate $= \emptyset$	*At most one can have data at any time*

The second constraint in the table 2.8 ensures that the rule id that maps to the template of the working memory of preconditions is consistent with the instantiated rule's state. Incoming data is constrained through *instate* and outgoing data is constrained through the *outstate* constraint. Once again, as rule-based systems operate in discrete cycles, there can only be either one input or output but never a simultaneous occurrence. The last constraint in table 2.8 ensures this.

State Step Change Schema

Table 2.9: Rule Step Schema Specification

$\triangle$ RuleState	*Step change in a rule*
r' = r	*Rule information does not change*
$\exists$ out: ACTION $\leftrightarrow$ FACT | ((curstate, in), (curstate', out)) $\in$ r.transitions $\bullet$ out = unified-set(in, precond) $\wedge$ instate' = $\emptyset\wedge$ outstate' = outstate $\cup$ out	*Given a valid curstate and a valid input, valid output is produced through set*

At every step the rule component transforms incoming *insts* to outgoing *changes* using the unified function[15]. This function writes the incoming insts with preconditions that in turn produce changes. It is these *changes* that are delivered to the I.E. As a result, constrains involve the removal of a rule instantiation from the incoming data and the addition of new changes.

[15]Equation 2.3

2.2.4.4 Working Memory

Component Schema

Table 2.10: Working Memory Schema Specification

facts: **set** FACT	*All facts*
initfacts: **set** FACT	*All initialising facts*
changes: **set** FACT	*All possible changes*
states: **set** STATE	*Legal states*
start: STATE	*Starting state*
transition: (STATE $\times$ **set** FACT) $\leftrightarrow$ (STATE)	*Valid transitions*
start $\in$ states	*Start state is valid*
changes $\subseteq$ facts	*Incoming changes are valid*
$\forall$ s_1, s_2: STATE; ch: **set** ACTION | ((s_1,ch) (s_2)) $\in$ transitions $\bullet$ s_1, s_2 $\in$ states $\wedge$ ch $\subseteq$ changes	*Given the current state and the set of changes, a new state is produced*

The Working Memory Component is a simple component in the system. It can be thought of as an unordered list of facts that can never be empty. It should be composed of initialising facts in the very least. Other *facts*[16] can be inserted into the working memory via *changes*[17]. It transitions through new facts that come from these changes.

Start states need to be part of valid states. Changes can only be facts. A valid transition is one that produces a new state given the current state and valid changes as seen from the last constrain in table 2.10.

State Schema

An instance of the working memory needs to be created to enable it to possess state. In this case we create *wm* as seen in definition 2.11. The purpose of the Working Memory state schema is to assert, at any given point in time, that the working memory is consistent with its internal state, incoming data and outgoing data.

[16]Defined in definition table 2.10

[17]Also defined in table 2.10

Table 2.11: Working Memory State Schema Specification

wm: Working_Memory_Component	*Instance of the Working Memory*
curstate: STATE	*Current internal state*
state_vars: WMSTATE	*The local variable of instance*
instate: **set** FACT	*Input data*
outstate: **set** FACT	*Output data*
curstate = (state_vars),	
curstate $\in$ wm.state	*The current state is valid and equal to all the state info*
instate $\subseteq$ wm.facts	*Input data is valid*
outstate $\subseteq$ wm.changes	*Output data is valid*
instate $\neq \emptyset \Rightarrow$ outstate $= \emptyset$	
outstate $\neq \emptyset \Rightarrow$ instate $= \emptyset$	*At most one can have data at any time*
curstate $\neq \emptyset$,	
curstate $\in$ wm.initfacts	*The current state can never be empty; it should be composed of initialising facts in the very least*

The current state is as a composition of itself, via its state_vars (unlike the Inference Engine and Rules). Incoming data is constrained through the *instate* and outgoing data is constrained through *outstate* as noted in table 2.11. The current state can never be empty, it has to be composed of the intialising facts in the very least. The last constraint in table 2.11 portrays this.

State Step Change Schema

Table 2.12: Working Memory Step Schema Specification

$\triangle$ WorkingMemoryState	*Step change in the working memory*
wm' = wm	*Working Memory information does not change*
$\exists$ out: **set** FACT $\mid$ ((curstate, in), (curstate', out)) $\in$ wm.transitions $\bullet$ out = update(in, changes) $\wedge$ instate' = $\emptyset \wedge$ outstate' = outstate $\cup$ out	*Given a valid curstate and a valid input, valid output is produced through with the update function*

At every step, the working memory component transforms incoming input to output. It does this via the update function (equation 2.2).

2.2.5 Distributed Ledger Technology

Distributed Ledger Technology makes use of a distributed (i.e. shared), replicated ledger (state) across a network of peer-to-peer nodes that keep consistency of the shared state through consensus. This consensus is often Hardened Byzantinian Fault tolerant as the peers may not trust each other. The first use of this technology was in Bitcoin [59]. A number of other projects include Ethereum [20] which is a permissionless ledger and Quorum [58] and Hyperledger Fabric [3] which are private and permissioned[18]. This chapter presents Ethereum and it's permissioned derivative: Quorum. Quorum has been used in the second investigation [19].

2.2.5.1 Ethereum

Ethereum is the biggest network for decentralised computation when measured by market capitalisation [23, 2]. Launched in 2015, Ethereum is a decentralised platform, offering the ability to run decentralised applications where code execution is distributed across the Ethereum network [20, 85].

Decentralised computation works by running code (smart contracts) on the Ethereum Virtual Machine (EVM), which can be thought of a distributed global computer. Since the EVM does not exist in any one physical location, but rather in all locations where there are nodes, code execution inherits the benefits of decentralisation [85]. Of these benefits, one of the most significant is a guarantee of immutability, meaning that it is impossible for published code to be modified. The nature of the network further guarantees zero downtime since there exists no single node and thus no single point of failure [20].

A fundamental entity in Ethereum is the account [20]. Accounts can be owned by either a human (external accounts, controlled by a private key) or autonomous (contract accounts, controlled by the logic of their contract code). They are identified on the network by a unique address, and are comprised of four fields: a nonce, ether balance (ether is discussed later in this section), contract code (often referred to as a smart contract) and storage. Unlike the other fields, the contract code is immutable and cannot be altered after the account is created [20, 85].

[18]The degree of privacy and permissioning can vary depending upon how the network is configured

[19]In reality an HL Fabric implementation was also made by the collaborators, as one of their objectives was to assess a Quorum vs. HL Fabric. This thesis presents the Quorum design and implementation.

Entities external to Ethereum that wish to interact with an account, typically users, do so by creating a transaction. Transactions either initiate the execution of an accounts contract code, or create a new account. Valid transactions are collected into blocks, and during the block validation, code at the accounts specified by each transaction is executed, or, if no account exists at the given address then one is created. Contract code can send messages to other contracts, causing their code to run, and it is in this way that contract communicate with each other autonomously [20, 85].

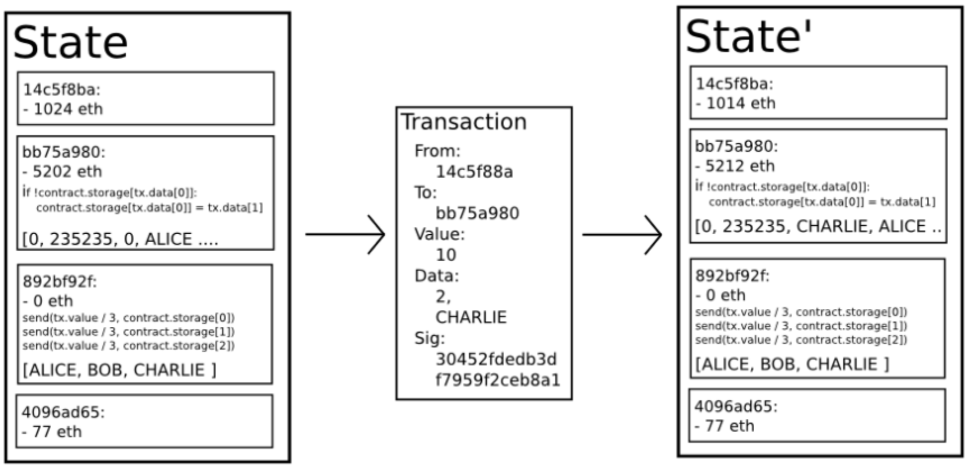

State
14c5f8ba:
- 1024 eth
bb75a980:
- 5202 eth
if !contract.storage[tx.data[0]]:
contract.storage[tx.data[0]] = tx.data[1]
[0, 235235, 0, ALICE ....
892bf92f:
- 0 eth
send(tx.value / 3, contract.storage[0])
send(tx.value / 3, contract.storage[1])
send(tx.value / 3, contract.storage[2])
[ALICE, BOB, CHARLIE]
4096ad65:
- 77 eth
Transaction
From:
14c5f88a
To:
bb75a980
Value:
10
Data:
2,
CHARLIE
Sig:
30452fdedb3d
f7959f2ceb8a1
State'
14c5f8ba:
- 1014 eth
bb75a980:
- 5212 eth
if !contract.storage[tx.data[0]]:
contract.storage[tx.data[0]] = tx.data[1]
[0, 235235, CHARLIE, ALICE ..
892bf92f:
- 0 eth
send(tx.value / 3, contract.storage[0])
send(tx.value / 3, contract.storage[1])
send(tx.value / 3, contract.storage[2])
[ALICE, BOB, CHARLIE]
4096ad65:
- 77 eth

Figure 2.4: Ethereum State Transition, adapted from [20]

Each transaction that updates state on the network invokes the Ethereum state transition function. This is a simple, deterministic function δ that takes the current state S and a transaction and returns a new state S':

$$\delta : (S, T_x) \rightarrow S'$$

δ performs several important tasks, including the validation of T_x and adding the transaction that pays the miner. More interestingly, however, it is in this function that T_x's contract code is executed. As such, the result of T_x's contract code is encoded within S' meaning that all the nodes that validate T_x's block from this point onwards will execute T_x's code. The parameters found in S and T_x can be seen from figure 2.4

2.2.5.2 Quorum

In commercial contexts, one of the biggest drawbacks to Ethereum is that every transaction on the network is public. Anyone with access to the network can look at the data of any transaction or account [58, 55]. Clearly, this is unsuitable for sensitive information, which firms have significant interest in protecting [55].

Quorum aims to address this issue. It is a version of Ethereum (specifically a fork of the go-ethereum library, commonly referred to as Geth) developed at J.P. Morgan which incorporates an additional private state tree visible only to certain users [58]. As such, private transactions can be made on a public Quorum network without divulging details about that transaction. As asserted on their website, Quorum offers "high speed and high throughput processing of private transactions within a permissioned group of known participants[58].

As in Ethereum, a Quorum network is comprised of a set of connected nodes that work together to maintain consensus. However, each node is subtly modified to support Quorums private state, among other things [58]. Additionally, each node is paired with a constellation node comprised of a transaction manager and an enclave, as seen from figure 2.5.

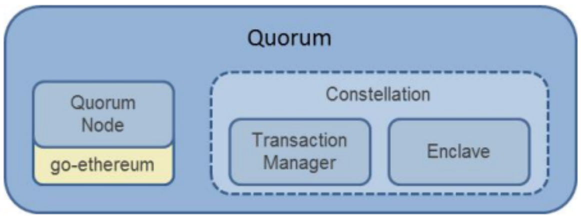


Figure 2.5: Quorum Node, adapted from [58]

In a Quorum network, the state database of each node is split into a public state and a private state. "All nodes are in perfect state consensus based on their public state". However, the private state is different. This permissioned private feature makes Quorum suitable for developing blockchain systems in which financial institutions and regulators want to share sensitive financial data using private channels[58].

Transactions can be made private for any number of nodes on the network by setting the transactions `privateFor` field. This instantiates a sequence of computation when that transaction is received.

For example, in a network of three nodes: A, B and C, if A wished to send a private transaction to B it could list Bs public key in `privateFor`, omitting C. Whilst public transactions are handled in the normal way, private transactions go through an additional step where the transaction payload is replaced with a hash. This hash is generated in such a way that only nodes explicitly listed in `privateFor` can replace the hash with the original payload[58].

As depicted in figure 2.5, Quorum contains 4 components that are further explained below[58]:

1. Transaction Manager manages the encrypted private communication channels and data store
2. Crypto Enclave manages the private keys of the network
3. QuorumChain is part of the QuorumNode, it represents a voting consensus that check and propagate votes on the network
4. Network Manager is part of the QuorumNode, it enables the creation of permissioned networks

2.2.6 Federated Learning

The traditional machine learning strategy is to gather raw data together (e.g. in a central repository hosted in the cloud) for training. This is characterised as taking the data to the algorithm. In contrast, federated learning involves taking the algorithm to the data. The typical federated learning paradigm involves two stages: a) clients train models with their local datasets independently, and b) the data centre gathers the locally trained models and aggregates them to obtain a shared global model.

Federated machine learning by definition aims to build a joint model based on data located at multiple sites. It involves two processes: i) model training and ii) model inference. In the process of model training, information [20] can be exchanged between parties but not the underlying sensitive raw data. The exchange does not reveal any protected private portions of the data at each site. The trained model can reside at one party or be shared among multiple parties. At model the inference stage, the model is applied to a new data for the purposes of prediction and analytics(e.g. a federated anti money laundering detection system may receive a monetary transaction from a bank. Banks collaborate in classifying this transaction as legitimate or fraudulent based on previous transactional patterns of historically similar type of transactions).

Broadly, federated learning ecosystems address:

- On-device (i.e. cross-device) infrastructures for mobile devices, IoT, Edge and other connected devices. For example, Google and Apple used it for keyboard (next-word prediction models) [53]. The data ecosystem is characterised by a very large number of devices (tens or hundreds of millions), with intermittency and low bandwidth connections.
- Inter-organisation (i.e. cross-silo) infrastructures allowing collaborating organisations to contribute to the training with their local datasets. An example is the European MELLODDY project [19], where ten pharmaceutical companies collaborate in training a machine learning for drug discovery based on private, highly sensitive datasets. This data ecosystem is characterised by a smaller number of participants with good bandwidth and connectivity.

[20]Usually model weights and other statistical patterns

Chapter 3

Using AI to Automate the Regulatory Handbook

The chapter explores the use of traditional AI- knowledge base systems for capturing machine readable and executable semantics of regulation. The chapter begins with a deeper discussion on automating regulation & compliance, through the use of machine readable semantics of regulation. The chapter then delves into the theory of expert systems, inferences engines and other key concepts with the aim of laying the technical foundation necessary for subsequent sections, including an analysis on Gamble et al.'s [36] work on expressing expert system shells through the use of formal methods. The chapter then moves on to exploring the use of these formalisms in expressing regulatory rules. The chapter ends with a discussion on representing regulation from the FCA handbook with the semi-formal model and provides the design an implementation of a system based on this. The chapter concludes with a discussion on the shortfalls of the system and the possible solutions to address them.

3.1 Introduction

In [76] Treleaven et al. have modelled the concept of algorithmic regulation on algorithmic trading systems. It involves the consolidation of compliance data, social networks data and other sources of information to a platform that encodes compliance reports using distributed ledger technology, where regulation is enforced as executable programs in the form of smart contracts [76]. The platform in question would be composed of 5 operational aspects [1]:

1. Intelligent Regulatory Advice
2. Automated Monitoring
3. Automated Reporting
4. Regulatory Policy Monitoring
5. Automated Regulation

Each one of these operational aspects relies on the rules and principles presented within regulatory Handbooks, such as the FCA handbook [14]. All reporting and compliance related operations revolve around this handbook. However, the FCA handbook is computationally stolid - it has no formal semantics and all regulation is written in text based markup that cannot be used easily by machines to reason about regulation. This makes compliance, registration, monitoring, reporting and other operational tasks difficult to automate. The ability to interpret regulations from the Handbook by a machine is necessary before any other algorithmic regulation task can be carried out.

The motivation for the work within this chapter stems from this problem. The aim of the project is to develop a system that is capable of reasoning with regulation. This reasoning engine has then been integrated into a platform that allows financial companies to navigate through the handbook to carry out tasks such as license registration. The platform in turn serves as the Intelligent Regulatory Advisor component noted in figure 1.1 and [76].

An important point to note is this project has been carried out in close collaboration with the Financial Conduct Authority (FCA) and the RegTech company RegulAItion. The motivation for this collaboration is based on the need for developing a system that actually meets real world needs of the regulator and firms alike within the industry and is validated by industry.

[1]Discussed in greater detail, chapter 2

3.1.1 Objectives

The objectives of this project are to:

1. Develop a semi-formal model for an inference engine and regulatory knowledge base that is capable of reasoning with regulation

2. Once a model has been developed, it will be implemented and its ability to reason with regulation will be tested. This involves a number of sub-objectives:

   (a) Implementation of the regulatory expert system

   (b) Analysing its ability to reason with regulation based on examples provided by the regulator and legal services

   (c) Adjusting the knowledge representation model based on the analysis

3. Develop a system that integrates the reasoning engine into an advisory platform. This also involves a number of sub-objectives

   (a) Designing a systems architecture, for the engine, to enable RESTful services

   (b) Developing a rich interactive UI for testing by the project collaborators

   (c) Deploying the system to a cloud based environment

   (d) Creating documentation for the entire platform

4. The shortcomings of the Intelligent Regulatory Advisory system as well as possible correction and improvements are then discussed

Thus the end goal is interpret to develop a system capable of providing an intelligent front-end to a regulatory handbook (through the interpretation of the rules and regulations found within it) that guides a registrant through the financial registration process and also provides basic automated regulatory advice for applicants with limited knowledge of the regulatory application process (e.g. FinTech bank.)

3.2 Background

3.2.1 A semi-formal model for a regulatory expert system

Representing regulation as rules is not a trivial task. This is because:

1. There exists no clear definition of what needs to be represented or how regulatory texts can be represented as a collection of regulatory rules. Even if there exists such a schema, there exist no clear way in which we can approach it or from where one should begin reasoning.

2. There is no clear distinction that separates rule based regulation from principle based regulation.

A formal model partially helps us address these issues. The first problem can be solved through the use of a Rule Component Schema definition such as the one found in table 2.7. This is because it provides a clear structure and definition of the parameters that need to be extracted, enabling us to create a valid rule whose behaviour is guaranteed. A schema also forces us to specify a start state, this provides a means through which it can be approached by the I.E [2].

Consider the Rule Schema in table 2.7, based on the schema it can be seen that a rule is composed of the following:

1. *rulename*
2. *facts*
3. *start*
4. *actions*
5. *changes*

Now consider the example found in figure B.1 of appendix B. The figure is part of the "Perimeter Guidance Manual" chapter of the FCA Handbook [9]. It deals with authorisations of firms. We need to identify the above parameters for each node in the diagram. The rulename is a unique id that we can arbitrarily allocate although

[2]It also provides us with a means to make a rule dependency tree in real time.

it is best if we let a hashing algorithm handle this to avoid collisions as this has to be unique [3]. We need our facts to be those that are asserted in the working memory such that the rule will fire. Thus we can either have “y” or “n” to represent this. Our start states are simply those the nodes that point to our node. Actions are the new rules that need to be fired given the response (i.e. the transitions). Our changes are the new rules that are fired as well as any other data such as recommendations made during inference. Based on this we can easily “translate” figure B.1 to the table B.1.

We can therefore see that it is possible to “translate” rule-based regulation into formally specified rules. However, there are times that the recommendations may not be presentable due to the their arbitrary nature. To address this we introduce a new type into the Rule Component Schema referred to as EXTSTATE. Its type however is concretised to STATE [4]. Its purpose is to address the arbitrariness of some portions of the rule. The rule component definition is redefined as:

Table 3.1: Modified Rule Component Schema Specification

rulename: RID	*Rule Name*
tags: **set** TAG	*All semantic tags pertinent to the rule*
insts: **set** FACT	*All facts triggered from the instantiation of rule(i.e. incoming data)*
actions: **set** ACTION	*Changes (i.e. outgoing data)*
changes:ACTION $\leftrightarrow$ FACT	*All actions pertinent to the given rule*
facts: **set** FACT	*All facts pertinent to the given rule*
states: **set** STATE	*Vaild states*
exttstate: EXTSTATE	*Variable represents context*
start: STATE	*Starting state*
transition: (STATE $\times$ **set** FACT) $\leftrightarrow$ (STATE $\times$(ACTION $\leftrightarrow$ FACT))	*Valid transitions*
start $\in$ states	*Start state is valid*
changes $\subseteq$ actions $\times$ facts	*Incoming changes are valid*
insts $\subseteq$ facts	*Outgoing data is valid*
$\forall$ s_1, s_2: STATE; ri: **set** FACT; ch: ACTION $\leftrightarrow$ FACT $\mid$ ((s_1,ri) (s_2, ch)) $\in$ transitions $\bullet$ s_1, s_2 $\in$ states $\wedge$ ri $\subseteq$ insts $\wedge$ ch $\subseteq$ changes	*Given the current state, rule instantiation, and the set of preconditions a new state with action facts(changes) are produced*

Similarly the Rule’s state schema needs to be redefined as:

[3]An annotated version of the diagram is represented in figure B.2, it shows which node corresponds to which unique rulename

[4]It can, to some extent be regarded, as a type synonym

Table 3.2: Modified Rule State Schema Specification

r:Rule_Component	*Given the instance of the rule*
curstate: STATE	*Current internal state*
state_vars: RSTATE	*Local variable of the state*
estate:r.extstate precond: PRECONDITION	*The rule precondition*
instate: **set** FACT	*Current input data*
outstate: ACTION $\leftrightarrow$ FACT	*The data on the output*
curstate = (state_vars, precond,estate)	*The current state is the current state info.*
precond.l = r.rulename	*The precondition is valid*
curstate $\in$ r.states	*The current state is valid w.r.t the rule*
instate $\subseteq$ r.insts	*Incoming data is valid*
outstate $\subseteq$ r.changes	*Outgoing data is valid*
instate $\neq \emptyset \Rightarrow$ outstate $= \emptyset$	
outstate $\neq \emptyset \Rightarrow$ instate $= \emptyset$	*At most one can have data at any time*

The current state needs to now also be composed of the *extstate* variable from the component schema. This is done by taking the cartesian cross product of the *stat_vars*, *precond* and *estate* variables. One thing to note is that the step schema does not change because our definition constraints on *curstate* are of the same type [5].

What we still can't do is isolate rule based regulation from principles based regulation; rules need to be derived from principles and principles are meaningless without rules. One way to address this is to semantically tag our rules in some way such that an Inference Engine can identify the underlying principles that govern a given rule. Therefore, the Rule Component is revised to hold tags. In order to do this, a TAG type is introduced. It holds the tagging metadata, such as the family of regulation the rule belongs to. For instance, in the FCA Handbook example of figure B.1, we can possibly add the tag "authorisation" to indicate that the rule is pertinent to authorisations. This allows us to enrich the rule semantics because we can, by implication, reason that the principles applicable to "authorisation" are applicable to the rule. The revised rule schema is provided in table 3.1.

Thus, we can to some extent, represent the arbitrariness and the principles pertinent to the rule. Given that we have introduced an arbitrarily defined state

[5] As we have said that EXTSTATE is a type synonym for STATE

variable, we forgo the ability of the model to behave deterministically. This, however, gives us greater flexibility in the representation of rules- the model is no longer formal it is “semi-formal”. A key attribute of the model is that it only manipulates the rule based schema. The other components do not require modification and we can adopt Gamble et al.’s definition [36] for simplicity.

3.3 System Design

3.3.1 System Requirements

The purpose of the system is to provide an intelligent front-end to a regulatory handbook that guides a registrant through the financial registration process and also provides basic automated regulatory advice for applicants with limited knowledge of the regulatory application process (e.g. FinTech banks). It has two categories of users: an admin user and a normal user. The admin user will be able to create/delete/edit/test regulatory rules. A normal user will be able to use the system for seeking regulatory advice. Thus the system should:

1. Be capable of reasoning with regulation
2. Provide an intuitive means to encode regulations as rules
3. Provide a store for regulatory knowledge
4. Be easy to use and interact with on any device

Based on the above requirements the following were surmised, respectively:

1. The formal model from the previous chapter can be implemented into an Expert System capable of reasoning with regulation
2. A graphical means can be used to intuitively encode rules
3. A database can store the regulatory rules and knowledge
4. A rich web-based UI can be used to promote cross-platform usability

3.3.2 System Overview

The design for the system has been decomposed into four units that accommodate the above requirements.

1. The reasoning engine

   This is responsible for reasoning with regulation.

2. Database

   Rather than store knowledge directly into the reasoning engine, it is stored in a standalone database. At each session, knowledge is exported into the reasoning engine. This promotes scalability.

3. Front-End

   The system will be used by regulators and lawyers therefore designing an intuitive cross-platform UI is important.

4. Supporting Infrastructure

   Integrating the reasoning engine within API level infrastructure promotes extensibility.

3.3.3 Feature Driven Design

Rather than solely tackle the design of the system with a set of abstract requirements, a feature driven design methodology has been adopted. This provides a better benchmark for developing software that actually meets real-world needs, because development is inspired directly from the problem.

The problem considered was that of license registration where a client asks the question: “I would like to assist clients in buying equities. What licenses do I need?”. To answer this a lawyer/ regulatory advisor must ask the following questions[6]

1. Will you be buying the equities for them? (if yes, then dealing as agent or dealing as principal license is required)
2. Will you buy the equities in your name, then sell them to your client? (dealing as principal. If no, then only dealing as agent is required)
3. Where will the purchased equities be held? With custodian? With bank? (generally the answer will be yes to both or either one)

[6]The accompanying answers are also provided

4. Will you put clients in touch with other brokers? (if yes, then we have to continue with below)
5. Will you assist your clients in opening accounts with custodians and banks? (if yes, then arranging license may be required but not necessarily)
6. Will you assist clients by taking their purchase orders and sending them to brokers, banks or custodians? (if yes, then arranging license is required)
7. Will you put clients in touch with custodians? Or banks who provide custody? (if yes, then arranging custody license is required)
8. Will you advise your clients on which equities to buy? (if yes, then advising license is required)
9. Will you receive money from your clients when they open an account? (if yes, then system has to send them to us for advice. If no, then continue)
10. Will you receive money from your client before they purchase equities? Or will the money go directly to banks and other brokers? (if yes, then system has to send them to us for advice. If no, then finish)

The set of answers are logical and can be represented as a decision tree. Figure C.1 in Appendix C illustrates the decision tree for the above problem. Hence the system should have the following leading features:

1. The system has to enable lawyers/regulators (we term them as admin users) to create decision trees
2. These decision trees should be parsed, by the system, into rules and facts within the reasoning engine
3. The system should provide normal users with an interactive environment to query the inference engine to obtain regulatory advice.

3.3.4 Regulatory Reasoning Engine

The previous chapter discussed a semi-formal model for representing rule based regulation and a means of reasoning with it. An important property of the model is that a large part of it is generalised and applicable to most expert system shells, making it possible to bootstrap development and implementation onto an existing

Expert System Shell. This is because the underlying Inference Engine, Working Memory and their respective controllers conform to the model and are largely the same.

The Java Expert System Shell (Jess) was identified as the most suitable Expert System Shell. Jess makes use of a DSL known as the Java Expert System Shell Markup Language (JessML). It is syntactically similar to Lisp. For instance it has a fully parenthesised prefix notation and makes use of functional like notational structure [7]. A large part of JessML has also been influenced by another DSL used for Expert Systems - the CLIPS language. CLIPS (C Language Integrated Production System) is one of the first Expert System DSL and has been used to develop a large number of commercial systems. JessML borrows a large number of functions and other expert system programming conventions from CLIPS.

A key characteristic of Jess is that it is treated as a first-class citizen within Java and therefore embeds comfortably within the Java environment. This is possible because Jess Engine can be instantiated within a Java class. The Jess API (Application Programming Interface) provides a seamless means of interaction with the engine; it provides a direct mapping to JessML types and functions. For instance, consider the example in figure 3.1, both code fragments create a new template that holds a fact [8]. The first is in JessML and the second is written using the Java API. Although verbose, it is possible to interact with the reasoning engine fairly easily. Figure 3.2 provides an insight into this isomorphic relationship. Another important feature of Jess is that it can handle Java Bean Objects [26]. It is therefore possible to pass the Rule Component Schema into Jess as a Java Bean. It is important to note that a Java Bean holds state this makes it all the more suitable for the purpose of defining our rules based on the schema found in table 3.1. A detailed discussion has been carried out in the next chapter.

[7] Similar to Alonzo Chruch's Lambda Calculus

[8] In the example we consider a basic license type with a name and description

```
;;Create a new template to hold license type
(deftemplate license (slot name)(slot description))
```

```
//Instantiate a Jess Engine
Rete engine = new Engine();
//Instantiate a fact template
Deftemplate d = new Deftemplate ("license", engine);
//Define the slots in the engine with a number of initialising parameters
d.addSlot("name", Funcall.NIL, "STRING");
d.addSlot("type", Funcall.NIL, "STRING");
//Pass the fact to the Engine
engine.AddDeftemplate(d);
```

Figure 3.1: Fact template in JessML vs. Java API

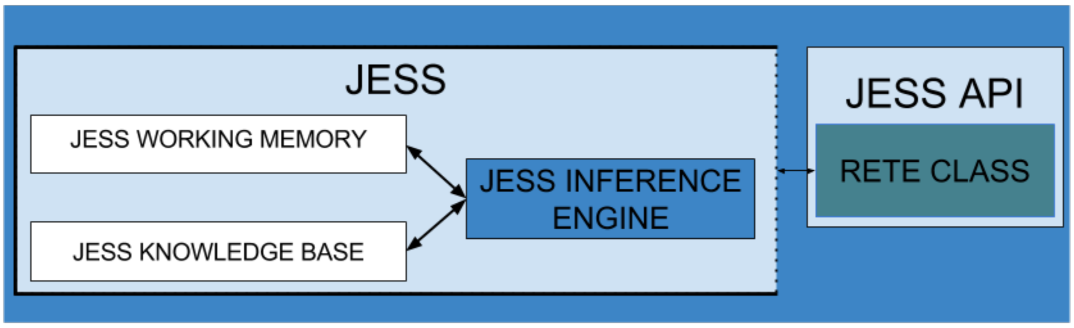


Figure 3.2: Jess Architecture and API access

The reasoning engine is the system's core. From a design perspective, this is the focal point. Therefore, all other design decisions have been made in cognisance of this.

3.3.5 Database

Integrating the reasoning engine with a standalone database is important for a number of reasons:

1. The database holds all the knowledge (regulatory facts and rules) collected during a knowledge engineering process. This is a very laborious task that takes a very long time. It also requires input from domain experts, which can be expensive. Hence it is crucial to isolate and back-up this knowledge as it provides redundancies against system failures - e.g. in the event that the reasoning engine fails we still have all the knowledge intact and well preserved in a database.

2. The entire repository of knowledge is not used in most cases. A subset of it is used most times. It therefore does not make sense for the reasoning engine to hold the entire store of knowledge at all times because it needlessly adds a memory overhead that impacts performance and speed of the system. A solution is to only import relevant knowledge from the entire repository of knowledge. For instance, if the system is required to give licensing recommendations, it should only import knowledge relevant to licensing from the database.
3. Jess's memory is mutable. The inference engine constantly changes the state of memory as it transitions through each rule. This means that multiple instances of the reasoning engine can't run simultaneously, unless they all have access to a global repository of knowledge. Therefore, integrating a database enables parallelisation of the reasoning engine.
4. Different types of users require different levels of access to the knowledge. Access restrictions and permissions are better managed directly from a database, freeing up the need for the reasoning engine to handle this. This reduces complexity.

A choice between an SQL database or a NoSQL database has to be made given the above database requirements and motivation. Table 3.3 summarises the limitations and merits of both. Our purpose requires a database capable of handling constantly

Table 3.3: Database Comparisons SQL vs NoSQL, adapted from [18]

SQL	**NoSQL**
Logical related discrete data requirements, possibly known beforehand	Unrelated and evolving data requirements
Data integrity is essential	Looser integrity requirements
Standards based - readily used in enterprise grade systems that need consistency and availability	Speed and scalabilty are more imperative that availability

evolving data-type definitions. Even though rules will be constrained to the formal definition, other aspects may require more loose definitions. Moreover, NoSQL databases have become more robust and reliable. They are increasingly being used for enterprise-grade systems. NoSQL databases hold data in the form of key-value pairs using JSON[9]. JSON representation provides an easier interface for programming rich and readily-scalable User Interfaces through frameworks. Thus a NoSQL database has been identified as fitting. MongoDB is the most widely used NoSQL database, it has a wide library of developer tools and documentation [56]. Therefore MongoDB has been selected as the system's database.

3.3.6 Front-End

The target users for the system are regulators, lawyers and people seeking regulatory advice. These users fall under two classes: admin users and normal users. The UI should enable admin users to:

- Create new rules and facts
- Modify existing rules resulting from changes in regulation
- Test the flow and logic of rules

The UI should enable normal users (e.g. clients) to:

- Seek advice by answering a number of questions
- Assist them with the registration process (through intelligent form completion)

These are just the basic UI requirements for the proof-of-concept. A large number of features can be integrated into the system in the future. Hence the UI needs to be extensible. This can be achieved through a component based modular design where each feature is boxed into an independent component that sits on a global dashboard. The target platform for the UI needs to be also considered. A browser based UI has been selected for the following reasons:

1. Browser based UIs promote cross-platform usage as most devices (mobile or desktop) have browsers
2. There exist a large number of frameworks that promote component based modular design, improving extensibility

[9] JavaScript Object Notation

3. There exist a large collection of libraries such the node package manager that could be used to provide more features in the future

The next sub-chapter discusses the choices made in selecting a framework for the UI.

3.3.7 Supporting Infrastructure

A service-oriented architecture has been adopted for the design and deployment of the system. This decision has been undertaken to promote a “Regulation As A Service” philosophy, where users can use the inference engine to reason with regulation that is relevant to them.

To facilitate this a Representational State Transfer (RESTful) architecture has been used. A RESTful architecture involves the design of RESTful APIs that use HTTP requests to GET, PUT, POST and DELETE data to/from a client from/to a server. RESTful architectures are a subset of client-server architectures that are widely used in industry as their design promotes scalability [10].

3.3.8 Frameworks

3.3.8.1 UI Frameworks

Frameworks provide browser-based application with a model view controller capability that promotes modularity and curtails spaghetti code. It also reduces the amount of JavaScript code needed through the use of single-page-applications. Frameworks also provide testing environments and tools. They promote rapid front-end development. Most are designed to accommodate RESTful architectures as well.

Two popular front-end frameworks are Google’s Angular 4 and Facebook’s ReactJS. Angular was chosen due to the more compositional design pattern it adopts.

[10] There are many reasons for this, [66] discusses them in detail. For instance the RESTful architecture enables dynamic load balancing across servers, especially within cloud computing environments.

Angular also makes use of TypeScript as its programming language. TypeScript is a strongly-typed superset of JavaScript. A strongly typed system promotes less error-prone development. The relative familiarity with Angular 4 over ReactJS was another factor that led to its selection.

3.3.8.2 RESTful Frameworks

The Reasoning Engine has been incorporated into Spring. Spring is a light-weight, enterprise grade framework that facilitates the creation of high performance, easily testable and reusable code. Moreover Spring is composed of a large number of modules that provide a wide number of features such as the creation of RESTful services.

Other RESTful frameworks such as Jersey and Spark exist. However, these are not as comprehensive as Spring. Therefore, Spring was selected.

3.3.9 Architecture

Thus, based on the discussions above, the final system architecture is presented in the figure 3.3 below.

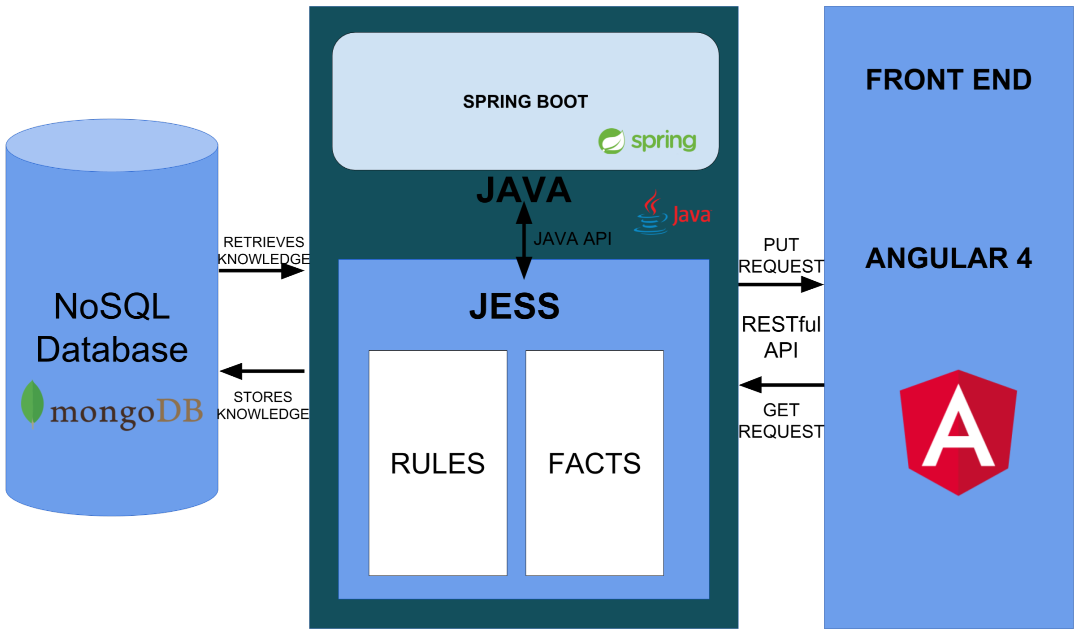


Figure 3.3: System Architecture

3.4 Implementation

This section provides the implementation details for the system. It begins with an overview of the central data structure, referred to as the “flow”, its structure and purpose. A discussion about integrating this data structure with a database has been provided, followed by a means of creating RESTful services. The chapter then moves onto a discussion about integrating the flow with the reasoning engine. Some familiarity with the Java Spring Framework has been assumed. The chapter finally ends with a brief discussion on the implementation of an Angular 4 User Interface.

3.4.1 The Flow Data Structure

The flow data structure is a directed acyclic graph for representing rules. In essence it is just a collection of relevant rules that are pertinent to one branch of regulation.

Figure 3.4 provides a class diagram for a flow. A flow could be thought of as a collection of Nodes and Edges. Nodes can be of different types. The parent type is simply referred to as a “Node” type. It consists of a unique identifier and comment describing its purpose. An OutboundNode type is one that has an additional component; an outgoing edge. OutboundNodes can either be MultiNodes, that have more than one outgoing edge, or purpose specific edges such as QuestionNodes. QuestionNodes hold the questions that are asked to a client. Nodes can also be terminal. Terminal nodes such as LicenseNode and LaywerNode represent logical conclusions that may be reached after transitioning through multiple QuestionNodes. Flows store Nodes based on their type - i.e. different nodes are stored as different collections. Nodes are connected by Edges. Edges are unique and are represented with identifiers. They contain information about their origin and destination Nodes and a comment that describes their purpose. Flows also have a unique identifier, a name, comments that describe their purpose and a flag for the Node that initiates the transitions across a flow. For simplicity the unique identifiers (id: Integer) are incrementing integers. These, however, can be substituted for digests from a suitable hash function to avoid collisions. It should be noted that the implementation of supporting objects such as

Nodes and Edges has not been provided for conciseness [11].

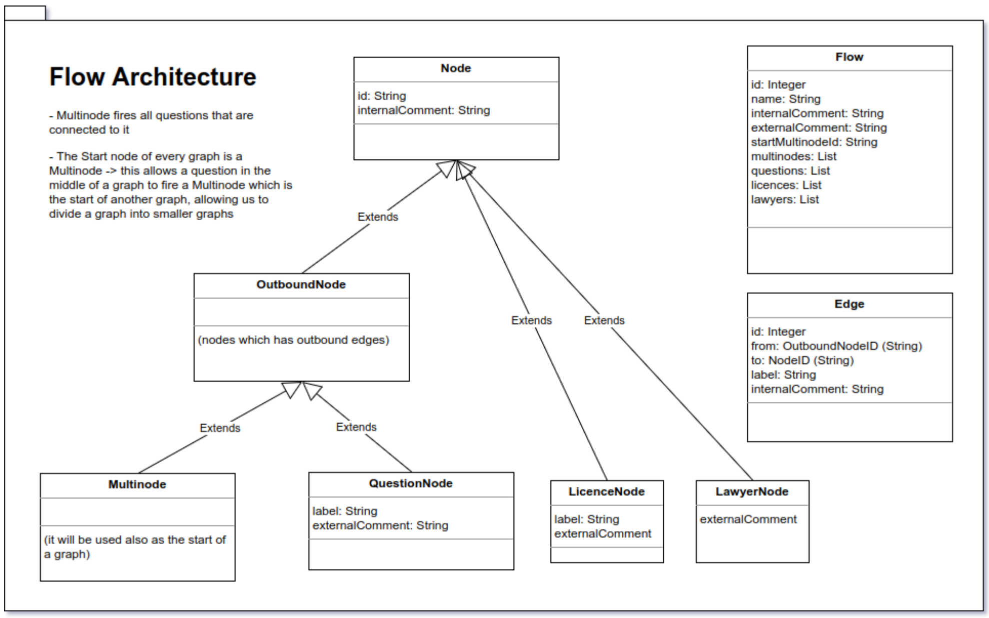


Figure 3.4: Flow Class Diagram

3.4.2 RESTful services and API creation

Figure 3.5 illustrates the structure of how RESTful services are created. Each object is confined to its own package with a controller and service. The controller is responsible for "transferring" HTTP requests (GET/POST/PUSH/DELETE) to the service. The service is responsible for holding the methods definition body onto whom the HTTP requests are mapped.

The Spring framework makes use of the `@RequestMapping` annotation above controller class methods to map the given HTTP request to its method. RequestMappings by path have been used. This means that an incoming URL is pattern matched with the specified path. If the pattern matches, the method body annotated directly below will be executed.

[11]However, an implementation of their use with RESTful methods is discussed in a later part of this chapter

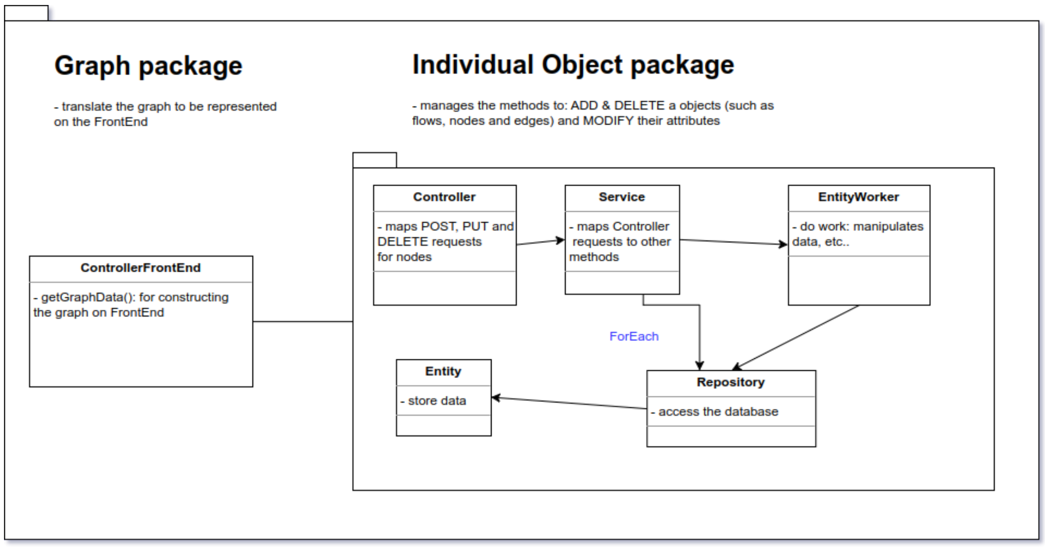

Graph package
- translate the graph to be represented on the FrontEnd
Individual Object package
- manages the methods to: ADD & DELETE a objects (such as flows, nodes and edges) and MODIFY their attributes
ControllerFrontEnd
- getGraphData(): for constructing the graph on FrontEnd
Controller
- maps POST, PUT and DELETE requests for nodes
Service
- maps Controller requests to other methods
EntityWorker
- do work: manipulates data, etc..
ForEach
Entity
- store data
Repository
- access the database

Figure 3.5: RESTful Service Diagram

3.4.2.1 Implementing a Service

Consider a given flow, users may want to either add, delete, retrieve or update a flow. This means that our flow service class will need to handle these methods. Flows can be added by creating a start node and specifying the all the nodes and their edges in the collection. The body of the `addFlow` illustrates this. Similarly, flows can be updated, retrieved or deleted, based upon their unique identifier, via the `updateFlow`, `getFlow` and `deleteFlow` methods.

3.4.2.2 Implementing a Controller

As soon as the controller is instantiated, we autowire the service onto it. This resolves injection dependencies in Spring. The `@RequestMapping` annotation is used to map the HTTP request with the method definition. We define the `GET` RequestMethod above our `getFlow` method, based upon the unique ID. The `getFlow` method then calls on its service to fulfill this task.

The RESTful services for other objects such as Nodes and Edges have been implemented in a similar, modular fashion.

3.4.3 Database

For the purpose of consistency, all knowledge that is engineered during a knowledge engineering process is stored as a flow. Therefore, the database schema is such that it is able to store different flows.

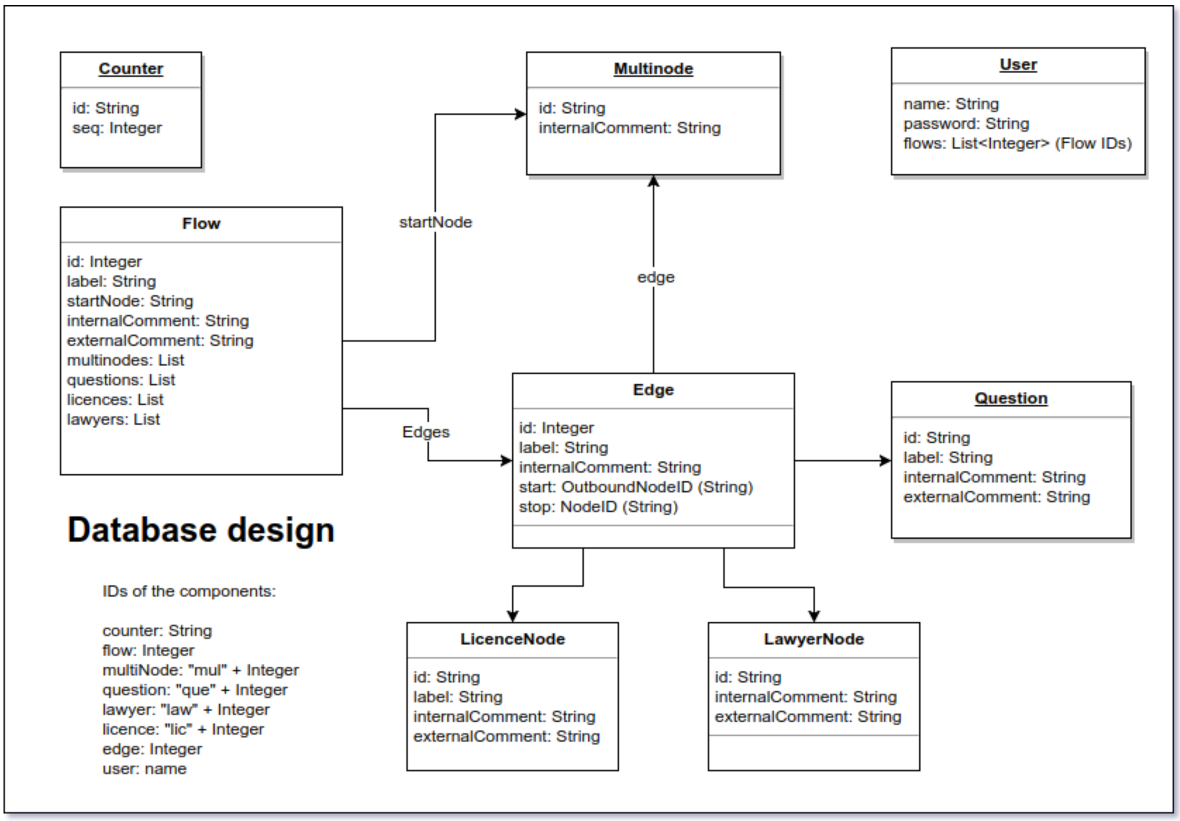


Figure 3.6: Database Schema

Intuitively, the database could be regarded as a list of flows with a unique identifier, name and a set of comments. It also stores a nested collection of multi-nodes, question-nodes, license-nodes, lawyer-nodes and edges. Each of these objects in turn hold their data, such as their unique identifiers and labels.

Before being inserted into the database the flow needs to be manipulated. Spring maps the class into a JSON collection, however in order to do this it must be provided with a template. This template must begin with the `@Document` annotation. This notifies Spring that the given class accesses the database. This

EntityWorker[12]is used to store and receive a QuestionNode from the database. However, before the database can be accessed, we require an interface that can cycle through the data already stored within the database. The Repository interface enables us to do this.

3.4.4 Reasoning Engine

The Reasoning Engine is the core of the system as it handles all the logic and queries. Development of this has been bootstrapped onto the Jess Expert System Shell. Java treats Jess as a first-class citizen and can be accessed from Java classes through the instantiation of a `rete` object. A `rete` object is just an instance of the Jess Engine. The Jess-Java API can then be used to interact with the engine, add rules and facts or query the engine. The `start` method in Jess enables us to do this. Other methods such as `stop` and `retrieveState` control the engine. The `defineClasses` method in the snippet injects rules into the engine session. Similarly, the `defineFacts` method injects initial facts into the working memory.

Figure 3.7 provides the class diagram for the engine.

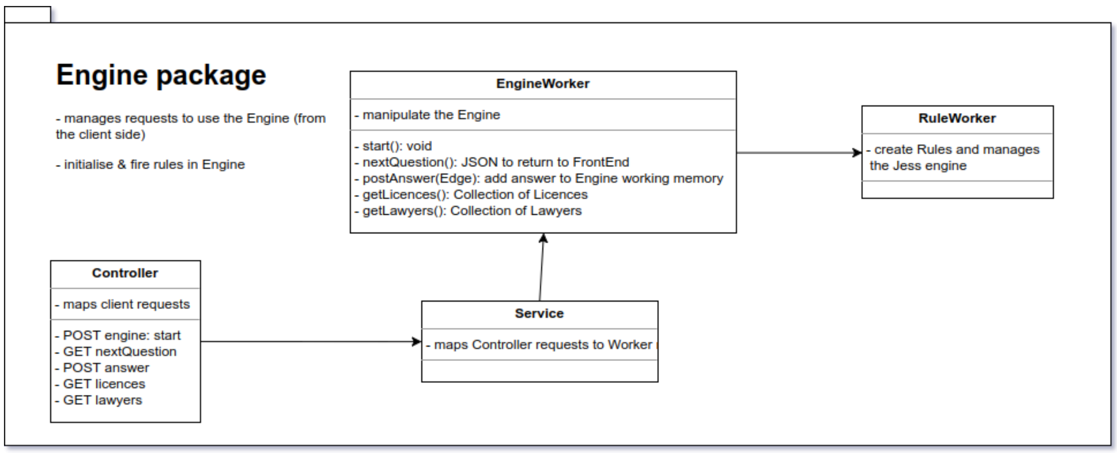

Engine package
- manages requests to use the Engine (from the client side)
- initialise & fire rules in Engine
EngineWorker
- manipulate the Engine
- start(): void
- nextQuestion(): JSON to return to FrontEnd
- postAnswer(Edge): add answer to Engine working memory
- getLicences(): Collection of Licences
- getLawyers(): Collection of Lawyers
RuleWorker
- create Rules and manages the Jess engine
Controller
- maps client requests
- POST engine: start
- GET nextQuestion
- POST answer
- GET licences
- GET lawyers
Service
- maps Controller requests to Worker

Figure 3.7: Engine Schema

Before being injected into the reasoning engine, the rules and facts need to be parsed from a flow. In order to do this, a number of parsing classes have been

[12]This has been written by Andrei Margiki

written. The method of interest within this class is the `transformQuestionNode`. It tokenises the metadata of the QuestionNode and creates an ArrayList of tokenised arguments that are used to inject rules into the reasoning engine. It is these set of arguments that are based on the semi-formal model define in the previous section of this chapter.

It should be noted that the above technique of parsing individual objects is sub-optimal as it does not scale well when new types of Nodes are used. For this reason, a standalone JSON to JessML parser is under development[13].

3.4.5 UI Implementation

Angular4 has been used for the implementation of the UI. In Angular4 each UI feature is compartmentalised into standalone components. These components rest on a parent component e.g. a dashboard. Each component is has 3 files: an HTML file that describes the structure of the component; a CSS file that provides a set of styles for the component and a TypeScript file that acts as the controller for the component and handles all the business logic.

Consider the example to illustrate the UI's control logic, we require that a flow of a given I.D needs to be displayed. For this, the TypeScript file must send an HTTP request to the server to collect the information about the flow. Once this is done, it filters the array of flows that were received, based on the ID of interest. Angular now has access to this and can inject it into the HTML Document Object Model (DOM) via string interpolation. Other components may require different services, some may require that we send a payload of data to the server. POST methods come in use for this. Different components require different levels of interaction with the server. The isolation of each of these through individual TypeScript files makes this extensible. Each component has a structure that is nested within a separate HTML file.

[13]It is not yet complete and therefore not presented

With regards to styling, Twitter BootStrap's CSS grid system has been used for responsiveness. The other aspects of the style have come from a boiler-plate theme.

3.5 Testing & Results

3.5.1 Testing

Testing is a crucial part of the development lifecycle, particularly when it comes to systems like this that provide guidance. As such it acts as a safety net, helping avoid the unwitting deployment of broken code. Given that the system was to be tested in a production like environment by the collaborators, the system had to go thorough testing requirements imposed by the collaborators, this included a penetration test.

3.5.1.1 Unit Tests

Unit tests check for the validity and correctness of an individual unit. A unit can be arbitrarily defined, but is typically a function or method. Well tested applications employ many unit tests to assert various properties across the codebase.

The entire SpringBoot application has been tested using the JUnit framework. The frontend codebase uses Jasmine, a test generation framework, and Karma to run the test suite.

3.5.1.2 Deployment and Manual Testing

Whilst testing local versions of the codebase captures a wide range of errors, it is has been equally important to test a deployed version of the application. Particularly because deployment environments vary from the development environment. Additionally, new constraints, such as CORS and SSL certificates, come into play each of which can cause unforeseen integration problems.

The project also made use of continuous integration pipelines (initially using CircleCI and then using GitLab [14]) to manage deployment of code into the production environment, and manage deployment to the master branch from dev. This formed

[14]Cost was the motivation for this

part of our continuous development process, and meant that deployment bugs could be addressed as they rose. Part of the build pipeline made use of a code scanner tool (SonarQube) to assess for any code vulnerabilities.

3.5.1.3 Pen Testing

As part of the collaborators' requirements, the system was put through a black box penetration test before use. There were a number of minor vulnerabilities found particularly with Cross-Site Scripting that were fixed before use with the collaborators [15].

3.5.2 Results

Due to the commercial sensitivity of the testing- the tests were carried out in a real environment by the collaborators exact details cannot be provided. However, twelve different flows within consumer credit were created by the collaborators from the handbook [16]. From these about 600 individual set of queries, based on if and what licenses (i.e. authorisations) were needed, were passed through the system. The responses and guidance from these queries were manually compared with the responses (of the same set of queries) from the legal experts. The following empirical results were obtained:

1. Processing a set of queries was about 12 times quicker than the human legal expert equivalent

2. The system provided a 94% true positive rate with respect to the accuracy of the guidance it provided, this was better than the human counterpart that averaged 85%

3. Where lacking, the system required additional information and flows- i.e. a larger knowledge base and therefore could not provide an answer

4. The user experience in receiving was ranked as 4/5 compared to the current 2/5 for human guidance

[15]Exact details cannot be provided due to the sensitivity.
[16]Similar to the figure B

The system has been considered a success (in a business context) in its application to provide guidance for licensing however the collaborators need to augment its knowledge base and thoroughly test the guidance it provides before deploying it as the tool for providing regulatory guidance.

3.6 Conclusions

3.6.1 Summary

The project looks into the problem of machine interpretability of regulation and provides a "white-box" technique to address machine reasoning with regulation through the use of a knowledge base system. The project provides a means to capture machine readable and executable semantics of regulatory rules that could be used to automate financial services regulation. It demonstrates how the techniques proposed in the project can be applied to the FCA Handbook. The use of a "white-box" solution (compared to a "black-box" Machine Learning solution) has been validated by the regulator's approval in using it internally.

3.6.2 System Shortfalls

The system however has a number of shortcomings that need to be addressed:

1. There is a scalability bottle-neck due to lack of a JessML-JSON two way parser -facts and rules need to be converted from JSON strings to JessML. In order to do this currently, we rely on creating mapping classes that map each object (e.g. a flow node or an edge) in the JSON string to JessML and vice-versa. These classes are unique to each object. Their creation cannot scale when the number of objects grows. We therefore require a more automated means to achieve this. A 2-way parser can help resolve this.

2. Inadequate real-world regulatory testing -even though the system has been show to be able to reason with regulation, its design and development has been based on a limited number of examples. Further testing, based on diverse real world examples, is required to better assess the limitations of the system (and also by implication the semi-formal model).

3. Lack of diverse knowledge representation -the flow is a crucial and central data-structure within the system. Even though it possesses a fairly simple structure its graph structure can encode a large number of rules and facts. The flow however makes use of customised nodes to hold specialised knowledge that is particular to one kind of regulation. For instance, for the license and registration case, it makes use of a hard coded license node structures. These cannot be used to represent another type of regulation. It is this lack of knowledge diversity that needs to be addressed, it can be achieved through the use of more generalised nodes.

4. UI design and implementation -even though the UI is functional, it still requires more usable features. The knowledge engineering user interaction is still unwieldy and requires more work. This however is more manageable given the component based modular design choice that has been made as it facilitates rapid UI component development.

3.6.3 Evaluation of the Objectives

This section lists all the research goals of the project and highlights how they have been achieved.

> Develop a semi-formal model for an inference engine and regulatory knowledge base that is capable of reasoning with regulation

A semi-formal model capable has been developed. It is based on and extends Gamble et al.'s work on formalising Expert System models. The model has been used to represent regulation taken from the FCA handbook [17].

> Once a model has been developed, it will be implemented and its ability to reason with regulation will be tested. This will involve a number of sub-objectives:
>
> 1. Implementation of the regulatory expert system

[17] Refer to Chapters B and B.1

2. Analysing its ability to reason with regulation based on examples provided by the regulator
3. Adjusting the knowledge representation model based on the analysis

Given that the semi-formal model only makes changes to the Rules Schema and no other component of an Expert System, implementation has been boostrapped onto an existing Expert System Shell - Jess.

The license registration example has been used to throughout implementation. Based on an iterative design process "the flow" data-structure has been developed to better represent regulation.

> Develop a system that integrates the reasoning engine into an advisory platform. This will also involve a number of sub-objectives
>
> 1. Designing a systems architecture, for the engine, to enable RESTful services
> 2. Developing a rich interactive UI for testing by the project collaborators
> 3. Deploying the system to a cloud based environment
> 4. Creating documentation for the entire platform

The reasoning engine has been integrated into a platform capable of serving RESTful service. This has been done through the use of the enterprise-grade Java Spring Framework. The RESTful API endpoint is a rich, cross-platform interactive dashboard built with Angular4. It however needs to be used a lot more by regulators and lawyers to help identify and revise its shortcomings.

The back-end of the system has been deployed onto the Heroku Cloud service whereas the front-end of the system has been deployed onto Google's Firebase.

The front end is being ported to Heroku. Even though the code has been commented, suitable documentation for the system has not done. This sub-objective will be addressed in the near future.

> The shortcomings of the Intelligent Regulatory Advisory system as well as possible correction and improvements will then be discussed.

The need for the JessML-JSON two way parser, the inadequacy of real-world testing and the need for representing more diverse forms of regulatory knowledge have been discussed. The motivation and need for better user interaction has also been discussed.

Chapter 4

SmartReg: Using Blockchain for Regulatory Reporting

The chapter explores the use of permissioned blockchains for regulatory reporting. It also explores how smart contracts can be used to encode regulatory reports. The chapter begins with the problem that financial institutions have with scaling their regulatory reporting needs and then discusses the design goals of a system capable of addressing these problems. The chapter then presents, in detail, the solution architecture and its implementation and the results of the regulatory reporting experiment. The chapter ends with a discussion on the results and the impact of this study. The work in this chapter has been carried out with Santander UK PLC and the Financial Conduct Authority.

4.1 Introduction

Regulators in financial services, in the UK, have 2 main roles: to manage Prudential Risk and maintain Financial Conduct. Although 2 separate bodies carry out these functions in the UK, the means through which they carry this out is the same. They guide, supervise and indict firms based on their practice. The previous chapter described in detail how regulators could improve their guidance function. This chapter discusses how they can carry out their supervisory role more efficiently.

4.1.1 The problem with regulatory supervision

In order to supervise firms, regulators need to examine the practices of institutions they regulate. All regulated firms are required to submit data of their practice through regulatory reports [14]. Whereas handbooks, such as the FCA Handbook, provide the instructions of how regulatory reports should be built and delivered, the reports are explicit instructions of what fields of data need to be submitted for supervision. The level of regulation and regulatory reporting has significantly increased since the financial crisis [5, 6, 79].

Given this increase in reporting requirements, the complexity and time it takes for financial institutions to manage compliance reporting has also grown. Moreover regulators make ad-hoc data requests over and above routine reports submitted by regulated firms [79]. The only way most firms have managed to scale this increase in requirements and complexity is by increasing the headcount within their compliance departments [79].

There are a number of reasons why the process of supplying regulatory reports is increasingly complex [12, 79]. The process of building reports from the FCA Handbook is difficult and open to interpretation by legal departments in firms . Moreover the entire set of instructions for compiling a report can be spread across many different areas of interlinking regulation. At times, the wording found in the Handbook is insufficient or unclear for firms to understand. On the other side, regulators struggle to provide precise reporting instructions for about 50,000 firms that operate across financial services [12]. Often times firms need to make judgements based on their practice that makes it difficult to provide unambiguous, definitive requirements. Additionally firms could be operating across several jurisdictions, forcing them to repeat the reporting process in multiple regimes and jurisdictions. This repetition is often across the same data sets with similar requirements. Figure 4.1 demonstrates how reporting is done.

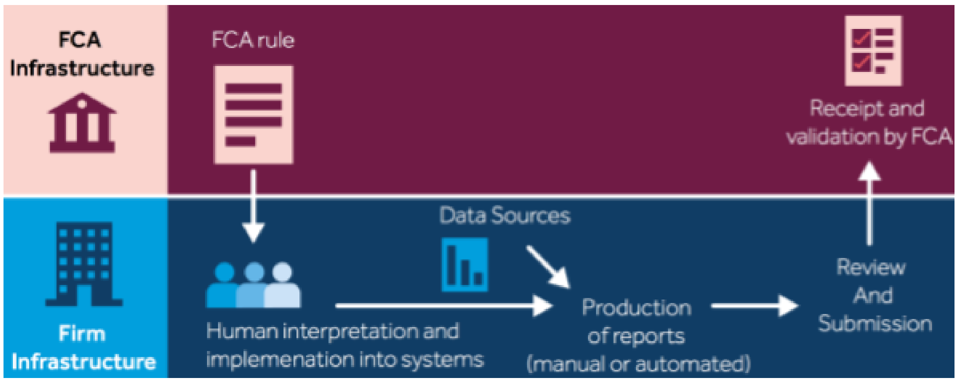

FCA Infrastructure
FCA rule
Receipt and validation by FCA
Data Sources
Review And Submission
Firm Infrastructure
Human interpretation and implemenation into systems
Production of reports (manual or automated)

Figure 4.1: Regulatory Reporting workflow [11]

Fundamentally, we argue that the problem is one of data. How can the regulator efficiently collect, store and use this data when need be. Moreover, can it use this data to monitor financial health in real-time? This an extremely arduous task because the regulator deals with thousands of firms of varied sizes, different capabilities and lines of business. Additionally, they all have different levels of technology to carry out compliance from containerised micro-services infrastructure in tech savvy funds to monolithic banking applications on mainframes, to spreadsheets in small family owned businesses. The one common thread they share is regulatory compliance and supervision.

4.1.2 Objectives

The aim of this project is to design and build a prototype solution that demonstrates a viable alternative to the current system for regulatory reporting; one which could be realistically explored further by Santander and regulators such as the FCA. The findings of this study will be used to inform discussions on the development of digital regulatory reporting infrastructure by the regulator.

The following objectives were established in order to realise this aim:

1. Identify problems with regulatory reporting currently

2. Establish systems design constraints based on these

3. Propose a solution architecture that addresses these design constraints

4. Implement, analyse and iterate the proposed solution

5. Test the solution in the "real world"

6. Analyse and compare the proposed solution to the current system

In order achieve the objectives, the following technical tasks were established:

- Standardise of reporting firms' data

- Develop machine-readable and executable reporting rules from the Handbook

- Develop a standardised automated reporting mechanism

4.2 Background

The FCA and Santander were consulted to identify the problems with reporting data they currently face, these responses were analysed collectively and the a set of requirements/ design constraints were established. These requirements/ design constraints are based on deeper underlying issues identified by the analysis. These were then used to identify broader solutions to the problem (all outlined in table 4.1).

Problems P1-P8 (table 4.1) affect both the regulator and the firms it regulates and in order to design an infrastructure that addresses both their needs, it is important to establish which entity should be responsible for solving each of the problems. It is also important to understand which entity develops and maintains such an infrastructure.

Table 4.1: Problems, design constraints and solutions for regulatory data submission

	Problem	Design Constraint/Requirement	solution
P1	Firms have to constantly interpret natural language regulation, and operationalise it through machine-readable code	Reduce the need for firms to interpret regulatory rules via natural language	Create machine-readable and executable reporting rules (or in a way such that they can be unambiguously transpiled to this), ab initio.
P2	Regulatory reporting is cumbersome, there are too many intermediate processes leading to errors in data submission	Improve the regulatory reporting workflow by removing intermediation as seen from figure 4.1	Automate the entire reporting process; rather than using multiple disparate systems, consolidate all processes and workflows into one platform
P3	New regulations and changes to existing regulation is time consuming	Enable new regulations and changes to existing regulations to be implemented more quickly and cheaply	By encoding reporting rules directly as machine-readable/ executable code, it is possible to treat the rules as a piece of software. New rules or amendments can then be treated like any other software rollouts that have inbuilt version control and continuous integration and delivery pipelines that facilitate a much quicker and more agile process
P4	There exist no single set of reporting standards or a uniform reporting framework, making it difficult for the regulator to analyse the submitted data	Allow regulators to collect more structured data which can allow them to identify and monitor risks more efficiently through analytics	Machine executable rules would produce reporting data that has a consistent data model.
P5	There exist no single set of reporting standards or a uniform reporting framework, firms have tried to tackle this multiple times through the introduction of uniform reporting data models but the adoption is minimal as firms find it difficult to refactor their existing systems to these new data models	The solution should promote the consistency and data quality across reporting firms	By enforcing the use of machine executable rules as the only acceptable mechanism to report data, it would be possible to maintain consistency.
P6	Both the regulator and the firms need to maintain a record of the reporting data	Remove needless data replication between the regulator and reporting firms	Regulators don't need store the data, all they need is a unique identifier that can be used to access the data if/when they need to from the bank
P7	Regulators cannot share reporting data of the firms it regulates, with each other. Conversely firms have to submit the same reporting data to different regulators	Provide a means for the regulator to share reporting data with other regulators. Also provide a means for firms to submit to multiple regulators.	Regulators could share these identifiers with other regulators. For intuition, the identifier could be thought of as a key to the underlying data. As such, regulators would then trade these keys (along with their inherent data). Similarly, firms could share the same identifier with different regulators.
P8	Data reporting happens infrequently as it is laborious	Improve the reporting frequency by firms	The consolidation of the entire reporting process into one platform will improve the efficiency of the reporting process- firms would be able to run more reports within a given time, making reporting more frequent and push towards real-time reporting

The obvious suggestion is that the regulator should be this entity as they are perceived neutral by firms and they authorise all regulation. However, this cannot be the case for a number of reasons:

- It is not within the regulators mandate to actively maintain infrastructure; their role is to facilitate and provide regulatory oversight not implement [14][1]

- As a policy, the regulator maintains technology agnosticism as much as possible [14, 25] [2].

- Regulators need to share reporting data with each other (e.g. the FCA sharing data with the Bank of England), what would the trust model look like if only one regulator maintained and developed such a system. Should all the regulators trust the one regulator in charge of the system. The trust problem is amplified by the fact that some of the regulators could be in different jurisdictions [3].

- Who manages the governance (technical, commercial and to a greater extent political governance) of the platform, should there be a regulator's regulator that manages this governance?

An alternative solution could be an industry-led platform similar to the way in which SWIFT works [69]. However, this also has a number of problems:

- The regulator regulates thousands of firms varying in size and technical competence; no one regulated firm or group of firms can be left in charge of developing and maintaining such a system

- It still does not solve the problem of regulators wanting to share data with each other, often within different jurisdictions

Given that one entity cannot be responsible for the maintenance and development of the platform. it is therefore important to understand the bounds of trust and the trust architectures available before considering the technical architecture of the system. The chapter below explores this, in the scope of networking and computing.

[1]Refer to section PRIN 1.1 Application and purpose
[2]Section REC 3.16 Information technology systems of the Handbook
[3]e.g. The FCA sharing data with the SEC

4.2.1 Trust models in Computing Networks

4.2.1.1 Centralised Trust

When it comes to centralised trust, client server architectures are the most widely used networked service architectures prevalent on the web [65]. In networking terms, the server is regarded as the "producer" as it provides the service [4]. Conversely, the client (i.e. the consumer) consumes this service and responds accordingly [65]. The responsibility of trust mainly lies with the producer [5] i.e. the server and all service related transactions are intermediated by the server [6].

4.2.1.2 Decentralised Trust in Computing Networks

Peer-to-peer (P2P) architectures are one of the most common decentralised compute architectures. The web is one such example. A computer is regarded both a 'producer' and a 'consumer' of a service [65]. Distributed ledgers, more commonly referred to as blockchains, provide a trusted decentralised architecture. This is done through the use of Byzantinian fault tolerant consensus that manages a replicated state of the entire network across the peers [59]. Effectively, trust is maintained via an honest majority principle on this ledger [85].

4.2.1.3 Trust in SmartReg

In SmartReg, trust does not fall within either extremities. It falls in between- Regulators do not trust each other (particularly if they operate cross jurisdiction). Firms however trust the regulator that regulates them. A hybrid model is therefore required. However, in the cases where trust is required it does not have to be based on hardened consensus. This is because regulators generally know each other and have their reputations staked and international legislation prohibits them from acting destructively, meaning that they do not have strong incentives to behave maliciously against their counterparts. Regulators are also aware of all the firms that regulate and actively police them, legal frameworks are also present within the regulator-

[4]A service is an abstraction of computer resources and a client does not have to be concerned with how the server performs while fulfilling the request and delivering the response.

[5]Protocols such as TLS maintain the integrity of this trusted relationship

[6]Distributed systems can still be centralised if the computation/ storage is managed/ orchestrated and carried out by a single entity such as an organisation

firm relationship that prevent them from acting dishonestly.
The problem of trust breaks down into "Federated Trust" that is based on the authority of each regulator, the firms they regulate and the wider legal implication that may apply to them if they behave nefariously. Effectively, SmartReg would have two components:

1. The network infrastructure that facilitates regulatory reporting
2. The machine-readable semantics that enable automated reporting

Where the latter uses the former to share executed reports[7].

4.2.2 Experiment

Based on the requirements discussed above the following experiment was designed to validate SmartReg[8]:

1. Contain three types of actors or participants: Regulators, banks and code providers where:
    - Regulators are responsible for receiving and validating the reporting data
    - Banks are the entities that need to report their data to the regulator
    - Code providers provide machine executable regulations to the banks
2. Manage six nodes: two regulators, two banks and two code provider nodes (as seen from figure 4.2; the nodes in the figure represent the participants and links represent the sharing of reporting data or machine executable regulations).
3. Represent a marketplace where code can:
    - be submitted by the code providers
    - be traded by the banks and approved, rejected or queried by the regulators

[7]Including the underlying data that is part of the report
[8]With input from the FCA and Santander

4. Represent a Submission Management Tool where the banks can submit reports to the regulators by using approved code and regulators can approve, reject or query them

5. Contain a screen to visualize the Code Inventory (i.e. submitted machine executable regulations), where:

   - the code providers can submit new code to the inventory
   - the regulators can change the status of the submitted code by using Smart Contracts

7. Contain a screen to visualize the Submission Management Tool, where:

   - the banks can submit new regulation reports through the Submission Tool
   - the regulators can change the status of the submitted reports by using Smart Contracts

8. Comply with the following set of permissions and privacy rules:

   - the code and Submission Inventories should be public to every participant node
   - Code can only be added by the code providers
   - Submission reports can only be submitted by the banks
   - The status of the code and submissions can be updated only by the regulators
   - Banks and regulators can only see records related to them

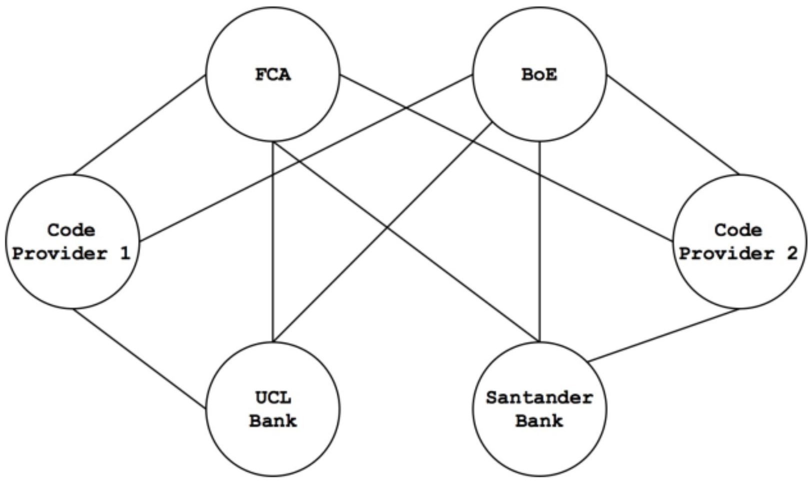


Figure 4.2: Network topology for the experiment

4.2.3 Use Cases

Two use cases were explored in the experiment. Both these use cases make use of a firms' compliance reports. The regulatory rules for these are defined in the FCA and PRA Handbooks [14].

4.2.3.1 PSD001 Loan to Income Ratio reporting check

This is a quarterly report that shares the data of all the mortgages a firm has sold. Regulators require this information to ensure that lenders are lending in a responsible manner. According to the Handbooks, no more that 15% of all mortgages sold should exceed a Loan to Income ratio of 4.5. The firms have to make this calculation and provide this to the regulator.
The reason for using this as a use case has been because PSD reporting usually falls within the remit of the PRA, but it is often the case that the other regulator (FCA) want this data. They usually get this data from the each other but it is a lengthy and error prone process (as outlined in the sections above).

4.2.3.2 CET1 Capital Equity Tier 1 compliance check

This is a report that shares the measure of the minimum capital a firms has against its assets. By requirement, this ratio must be greater than 4.5% of the risk weighted assets a firm has at all times.

The reason for using this as a use case has been because CET1 reporting also falls within the remit of the PRA, but it is often the case that the other regulator (FCA) want this data. Additionally, it falls under different regulatory reporting standards- the Basel III Standards.

4.3 Systems Design

This section examines, at a high-level, how SmartReg is designed, exploring the reasoning behind the choices made and the trade-offs encountered.

4.3.1 Systems Architecture

Before exploring how the disparate components of the system relate to each other, it is helpful to start with a list of the services that comprise SmartReg and their roles:

Table 4.2: SmartReg System Components

Service	Description
Frontend	Displays information to the user, and exposes a visual interface for interacting with that data.
Backend	Server which communicates with the Blockchain to create contracts on the network.
API	A RESTful API server that enables communication between the frontend and backend.
Authentication Server	Server which holds authentication details for the users of the system.
Database	Hold arbitrary business data. Cache blockchain data.
Blockchain	Decentralised network which stores an immutable growing chain of data.
Code Explorer	Frontend site which is essentially a wrapper around the GitHub API. It enables users to search for approved repositories, and for users to log in and submit their own repositories for approval.
Regulatory Report	A machine-readable and executable version of regulation- “codified” regulation.

Extraneous services such as load balancers and caches have been omitted here for the sake of brevity and because their exclusion does not meaningfully impact the core components of the system.

In addition to the business requirements discussed in table 4.1, it is crucial to consider the requirements of a system when it comes to system design. Every choice brings with it a set of trade-offs, and successful systems will use its requirements to inform which trade-offs are acceptable and which are not.

In the case of financial regulation, two of the most significant technical requirements are security and data privacy. As such the system was designed with modular components in a manner that gives each entity complete control over their data. The system should also avoid imposing any single opinionated way of delivering this, as different banks may have completely different approaches to data-privacy. Another consideration is that firms may want to connect SmartReg to their existing corporate authentication system, and SmartReg shouldn't enforce any particular constraints. Put succinctly, SmartReg should be built as a platform for supporting decentralised financial regulation, irrespective of the environment.

Our response to these constraints is to categorise each component as either a peer component or a reporting network component. The peer components include:

1. Frontend
2. Backend
3. API
4. Authentication Server
5. Database
6. Reporting Module
7. Quorum node

The reporting network on the other hand also includes the Quorum node but also the Quorum blockchain. Peer components will be replicated at each institution and thereby be under that institution's control, whilst a blockchain component is shared across all institutions, thereby remaining uncontrolled by any one actor. The rest of this section will discuss the design of each of these sections.

4.3.1.1 Peer Components

Each node in the network runs a traditional web stack, comprised of a frontend, backend, authentication server and database. There is nothing inherently special about this stack, in fact it is designed to be as generic as possible so that it can be deployed to whichever set of environments the client is running.

The frontend application is the primary way users will interact with the underlying application and blockchain. The application is decoupled interacting with the blockchain through the server via a RESTful API service [66, 71] and secure only ever receiving information that the current user has permissions to view, as defined by an access token.

One benefit of this approach is that the server and database can be switched out as needed from institution to institution, so long as the API specification is upheld. This makes for a highly flexible integration experience for institutions joining SmartReg, who often have a large amount of technical debt and legacy infrastructure.

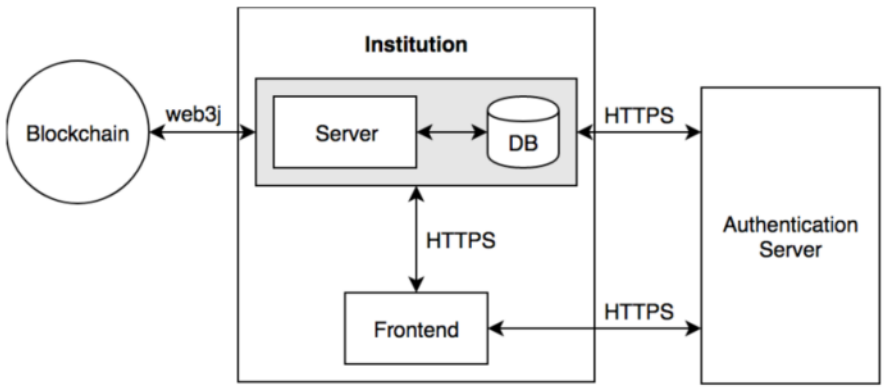


Figure 4.3: Architecture of an individual SmartReg peer

4.3.1.2 Frontend

The frontend is responsible for visually displaying data to SmartReg users. The UI is particularly important as it's the primary method of interaction with the underlying network. Since the primary audience of our solution are compliance professionals, not software engineers, a clean, intuitive UI will be key in selling the solution as a viable alternative to the tried and tested spreadsheet and email workflow used today.

There are some important design and security constraints to consider. Firstly, the frontend should make no assumptions about the backend, other than that it provides an API that meets a certain specification. This allows a loose coupling between the front and backend, for flexibility. No server specific or institution details are hard coded into the frontend codebase. Owing to this loose coupling, there only needs to exist a single code base to serve all entities on the network, with each entity simply deploying a copy of the frontend code and setting environment variables pointing to their API. Indeed, because the frontend is so stand-alone, it can be open sourced where it can benefit with input from the developer community.

Secondly, the frontend code is stateless and modular. API keys, URLs, authentication tokens, institution data, regulation data, submission data and other such information is never be hard-coded into the system, as doing so would greatly reduce the codebase's portability. Instead, data is requested from the API and used to populate an instance of the frontend. Meanwhile, API keys, endpoint and other per-institution data are defined in environment variables. In this way, each deployed instance of the frontend codebase can use a custom configuration, maximising flexibility across a range of deployment environments. This also delivers significant security benefits, as private keys and other custom configuration never need leave its institution.

Finally, the application accepts and stores an opaque, stateless access token

(SmartReg uses JSON Web Tokens -JWT) from the authorisation server upon a user's successful login [8]. This token is included as an authorisation header in all requests made to the server, and encodes details about the user making the request. In this way, the server can respond with only information that user has permissions to view.

4.3.1.3 Backend, Authentication and Database

The backend has several key responsibilities. Firstly, verifying authentication. Each incoming HTTP request from the frontend contains an access token in the request's authorisation header. Before processing the request, the backend should verify the integrity of the token. This exact verification process depends on whether the token was signed using public-private key cryptography or a shared secret. If any part of the information contained in the token has been changed then this verification step will fail and the request should be rejected. If the verification succeeds, and the token is within its expiry time, then the request is accepted and processed [8, 34].

Secondly, the server is responsible for reading and writing to the database. The fact that each institution connects to their own internal databases is significant as it enables users to authenticate with their institutional credentials. Additionally, the database can act as a cache, storing requests so the system doesn't always need to access the blockchain, thereby speeding up common requests.

Thirdly, the server needs to make connections to the Quorum network so that data can be read and written to the blockchain. We use Web3J, a library for interacting with nodes on an Ethereum-based blockchain. It is across this connection that nodes can view, modify and create submissions on the shared network, and retrieve data to send as API responses.

4.3.1.4 API

The sole method of communication between the frontend application and backend server is via a RESTful HTTP API [71]. The frontend requests data from the backend via this API, passing an authentication token with each request. According to the permissions encoded in the token, the backend responds with a payload, hydrating the client with data.

Writing the API specification is not something that could be done once and left alone. It needed to adapt as the project grew and changed, and as such we adopted a pragmatic, flexible approach to the API specification, one where changes were not just allowed, but encouraged as we rapidly iterated in SmartReg.

4.3.1.5 Reporting Module

The reporting module has 2 sub components:

1. Reports Explorer
2. Machine Executable Reports (also referred to as reporting functions)

The regulation function is a machine executable version of the regulatory rules. Its purpose is to deterministically map an input to an output according to the logic inscribed by a particular piece of regulation. The regulation function could be a single rule or a set of rules. The resulting output encodes whether or not the input has passed the function's regulation, among other things. In order to support automation in running the function, each function should conform to a standardised interface. This allows any function to be programmatically downloaded and run from within a virtual or containerised environment (e.g. Docker) on any server offering a significant speed and reliability boost when compared to a manual approach.

Because these functions are deterministic, the running of a function can be described by its input, function ID (comprised of the repository URL and git commit SHA), and its output. It is this determinism that makes verification so computationally easy; a regulator simply runs the specified function on the given inputs and checks that the output matches that given by the firm. Such steps are trivial to automate. Furthermore the determinism at play gives rise to greater transparency, as all steps a regulator take to verify compliance can be recorded and replayed at any time with identical results. This could benefit auditors, who might wish to replicate the exact steps taken at a particular time in history.

One important property of regulatory functions is their referrential transparency; they apply the same logic every time, regardless of the input. This represents one of the most significant benefits of using codified regulation, as the interpretation is fully defined with no subjectivity or ambiguity.

The code explorer is an online directory which enables users to search for and discover regulatory function repositories. Code providers can log in to the code explorer using their GitHub accounts and submit one of their repositories to a regulator. Each submission is represented by a code token smart contract which is stored on the blockchain. Regulators assess submitted repositories and hence approve or reject code tokens. Each accepted repository is associated with a set of approved commit SHAs, or versions, so approvals are tagged to a specific version of the code. This prevents code providers from pushing new, unapproved logic to previously approved repositories.

For the purposes of this project, a very simple UI wrapper around the GitHub API was built along with an additional form page for making code submissions. In the future, we envisage the creation of a fully featured online code directory with complex search, filtering and submission functionality.

4.3.1.6 Quorum Peer

The SmartReg Reporting Network consists of the decentralised network consists of a network of nodes which produce smart contracts, connected by Quorum. Each node performs a variety of functions, including the generation of non-deterministic data (such as the current time) and running computationally expensive tasks that do not need to be done on-chain. It has two main sub components: a peer identity wallet and a repository of smart contracts relevant to the peer. Each of the peers validate their identities, sign transactions and manage other peer responsibilities (e.g. consensus) through this identity. The wallet, referenced by an address number, manages this identity, it essentially is a secure artefact that manages secrets (i.e. private keys associated with the peer), based on the SECP26k1 ECDSA standard [85]. Section 4.3.2 goes into more detail of how smart contracts are designed to be used in SmartReg.

4.3.2 Reporting through Smart Contracts

To prove a bank has complied with regulation, it runs its inputs (financial data) on a regulatory function to deterministically generate an output. The entire process is orchestrated through the use of Smart Contracts. Smart Contracts are also used to persist the details of each regulatory report on the blockchain in the form of a token that represents a "proof of compliance". The contracts specification has been designed with the aim of being implemented in Solidity, a high-level, contract-oriented language similar to JavaScript [31], as seen from figure 4.5. The language compiles down to Ethereum virtual machine (EVM) instructions which are executed by the Ethereum virtual machine [31]. There exist four concrete contracts, and one abstract contract. These are:

1. Submission Factory
2. Submission Token
3. Code Factory
4. Code Token
5. Regulated (abstract)

4.3.2.1 Patterns

One pattern used heavily in the contract design is the factory pattern (figure 4.5). Each token, be it submission or code, has an associated factory contract which is responsible for instantiating the token.

A factory describes an architecture where an object, or in this case a contract, is created without calling that object's constructor. Instead, a factory method is called, which internally invokes the constructor, abstracting away the need to know about the details of the underlying object's instantiation. This is useful because it decouples the codebase from any one specific implementation. If, at a future date, the object be replaced with one that has an entirely different signature, the only place where changes would need to be made are in the factory method.

A further benefit, and the main reason for employing the factory pattern in our case, is that default values can be specified in the factory method and thereby applied to all new contracts created by that method. For the submission factory, this pattern is used to provide the value for the submission's regulator, with the values for the regulator being provided when the factory class is instantiated. In this way, every bank-regulator pair has a corresponding factory class, and by calling that class the regulator value is prefilled. The code factory contract behaves similarly.

The data that forms a token on the network is stored in the submission token and code token contracts. Whilst submission data is specific to each, both store the same regulation information, namely who the regulators are and the current submission status.

To promote code reuse, and to adhere to the don't repeat yourself (DRY) principle [44], this common regulation code has been abstracted into the Regulated contract. This contract is not designed to ever be instantiated on its own, but rather exists solely to be inherited (analogous to abstract classes in Java). By sub-classing the regulated contract both token contracts inherit its logic. Now, any updates to this shared piece of code only ever need take place in one file, improving code maintainability. Further to this, structuring the contracts in this way encourages a modular application design where specific pieces of logic are represented in one authoritative place, rather than the monolithic practice of combining distinct logical components together in one class.

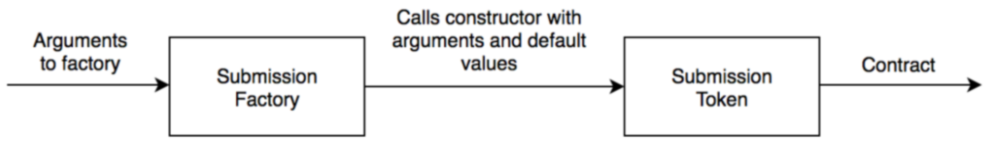


Figure 4.4: Flow of arguments to a factory method which instantiates a contract.

The submission and code factory contracts implement a factory design pattern. Each factory has a create method which performs token construction and injects certain defaults to that constructor, such as the regulator's address. Meanwhile, the regulated contract encapsulates functionality common to both token contracts in one place. These patterns are explored in detail in the subsections below.

4.3.2.2 Submission

Submission contracts hold submission data and encapsulate permission checking logic which, for example, sets the address of the sender making the contract as the owner of the submission (to prevent an entity submitting on behalf of another). A hash of the inputs, link to the function and output is sent to the network, where it is stored as an immutable item on the distributed ledger. Table 4.3 shows the information stored in each submission, along with a description of how the value is verified.

The address of the submission token serves as the token's ID. This reduces the space used by each token, and enforces uniqueness at the protocol level, reducing the amount of custom logic needed to read and write from the smart-contract.

At its core, the submission token consists of the following fields: the bank's address on the network, the address of the regulator this submission was submitted to, the submission status (pending, approved, rejected, queried), the address and SHA256 hash of the code token used to verify compliance, the input hash string and the proof string (which is the output of the function). The piece of regulation being complied with is implied by the code token (and the regulatory function it points to).

Table 4.3: Contents of a bank's submission

Data stored	**How to verify**
ID	Is a unique identifier at the protocol level (unique to each smart contract), verification can be done through comparing with block data
Submission timestamp.	Can be trivially verified by comparing it with the timestamp of the block the submission was "mined" in.
Address of the bank making the submission.	he address of the sender is set as the bank address, and this logic is written into the smart contracts. Thus, bank A cannot submit on behalf of bank B without bank B's private key.
Address of regulator being submitted to.	This is up to the bank, but entering an incorrect value here would be self defeating and not in the bank's interest.
The address of the code token that represent the regulatory function used	Together these three items form a proof of compliance. Changing any one of these values would cause a verification error if a regulator decided to check that token. This is made possible by the determinism of the regulatory function
The hash of the input data used.	
The output of the regulatory function.	

4.3.2.3 Code contract

Code contracts are smart contracts similar to submission contracts, figure 4.5. The code token contract represents a submitted code repository. They are submitted by code providers and approved, rejected or queried by regulators. Each code submission, also referred to as a code token, links to specific versions of a specific repository containing a regulatory function.

A code token is responsible for storing the author of the regulatory function (represented by an organisation name and optional URL), a repository (represented by a name, URL and array of allowed SHAs), a regulation ID corresponding to the piece of regulation that the function was written to verify, the address of the regulator the submission is for and the submission status (pending, approved, rejected, queried). When a code token has an approved status, it is available to banks for verifying their submissions.

In SmartReg a repository is specified by a GitHub URL and the versions are specified by git commit SHAs. Once a code token has been approved by a regulator, is on the bank's server and is available for use by banks, SmartReg uses the token's GitHub URL to download and run the regulatory function for verification.

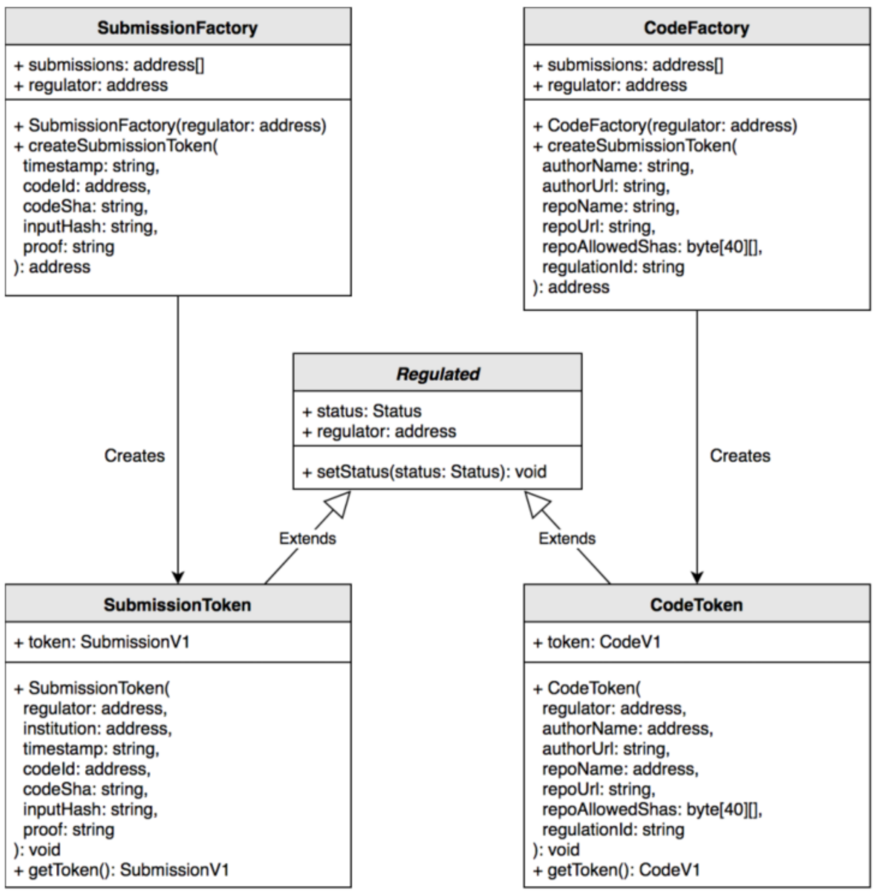

SubmissionFactory
+ submissions: address[]
+ regulator: address
+ SubmissionFactory(regulator: address)
+ createSubmissionToken(timestamp: string, codeId: address, codeSha: string, inputHash: string, proof: string): address
CodeFactory
+ submissions: address[]
+ regulator: address
+ CodeFactory(regulator: address)
+ createSubmissionToken(authorName: string, authorUrl: string, repoName: string, repoUrl: string, repoAllowedShas: byte[40][], regulationId: string): address
Regulated
+ status: Status
+ regulator: address
+ setStatus(status: Status): void
Creates
Creates
Extends
Extends
SubmissionToken
+ token: SubmissionV1
+ SubmissionToken(regulator: address, institution: address, timestamp: string, codeId: address, codeSha: string, inputHash: string, proof: string): void
+ getToken(): SubmissionV1
CodeToken
+ token: CodeV1
+ CodeToken(regulator: address, authorName: address, authorUrl: string, repoName: address, repoUrl: string, repoAllowedShas: byte[40][], regulationId: string): void
+ getToken(): CodeV1

Figure 4.5: UML diagram of reporting smart contracts

4.3.2.4 Limitations

At the time of conducting the SmartReg experiment, the latest version of Solidity was 0.4.21. As might be expected for such an early version there exist limitations in the language which need to be worked around and have, in some cases, constrained the design of the contracts.

One of the main limitations was Solidity's non-support for two-dimensional or higher arrays in parameters to external functions. So if a contract has an external function foo, it cannot have array parameters with dimensionality 2 or above (e.g. `uint[][]` is unsupported). This limitation becomes more significant when considering that strings in Solidity are internally represented as arrays of bytes. Hence, the type `string[]` is internally represented as `bytes[][]` a two dimensional array. This means that contracts cannot accept string arrays from external calls.

This presents a problem in the code token contract, which needs to accept an array of SHA strings. To address this issue, I exploited the consistent format of git SHAs, which are hashes of 40 characters. This consistent length could be exploited in the type of the SHA array type, by moving from `string[]`, an array of dynamically sized byte arrays, to `byte[40][]`, an array of statically-sized byte arrays. Because the byte array's length is known at compile time, Solidity can collapse this type to a single dimensional array, thereby allowing it in the parameter of the externally facing constructor.

However, there exists a second limitation which affects the `byte[40][]` workaround, and that is Solidity's lack of support for nested arrays in constructors. This means that even with the `byte[40][]` workaround, arrays cannot still be passed directly to contracts during construction. There exist several workarounds for this limitation, each with their own trade-offs.

1. Drop array support this is the simplest option, but means that each submitted SHA would need its own separate submission. For simplicity, this is the approach taken in the contract source code

2. Implement a public function, which supports being passed `byte[40][]`, that sets the allowed SHAs this function could be called directly after contract construction. It implements the functionality, but comes at a significant complexity cost

4.3.3 Consensus

SmartReg uses Quorum as its blockchain protocol. Part of the design decisions made also rely on defining the choice of consensus mechanism to validate transactions onto the distributed ledger. A consensus mechanism is needed because such a distributed system has peers that act independently when processing transactions and updating state. As a result, a non-disputable agreement amongst the peers is required to achieve synchronisation.

Consensus mechanisms need to be fault-tolerant, meaning that they need to provide a reliable mechanism to achieve synchronisation and network resiliency. Here two kinds of fault tolerance have been considered:

1. Crash Fault Tolerance - Fault tolerance that provides resiliency against peers failing (as a result of technical mishaps such as communications failure, peer outage etc). Common examples include Raft and Paxos [68, 49].
2. Byzantinian Fault Tolerance - Fault tolerance that provides resiliency against dishonest peers that may act adversarially. Common examples include Proof-of-Work, Proof-of-Stake and Proof-of-Authority [59, 85]

Open blockchain systems, such as Bitcoin and Ethereum, need Byzantinian Fault Tolerance as the identities and intentions of the peers is not known. The trust model in SmartReg is different. It is a private consortium blockchain that characterised by three distinctive trust properties:

1. The identities of all the peers within SmartReg are known, if they act dishonestly this can be identified and they can be punished by the regulator
2. Peer permissioning is required- all peers need to be explicitly allowed to join the network so the chances of adding ”dishonest” peers is reduced
3. Peers (i.e. the organisations/ firms) have a legal obligation to report honestly, failure to do this can result in being punished by the regulator [14]

As a result, Crash Fault Tolerance has only been considered when identifying a consensus mechanism rather than Byzantinian Fault Tolerance. Raft has been chosen as the consensus mechanism for SmartReg for a number of reasons:

- It has a simple leader-follower model and implementation compared to its Paxos counterpart [68]

- Raft only synchronises peers if transactions exist and if they need to be processed. Proof of Work and other Byzantinian Fault Tolerant mechanisms need to create blocks regardless of transaction processing. This can unnecessarily bloat the ever growing blockchain [58]

- Raft has been used in large scale enterprise-grade distributed applications such as Kubernetes [58]

In SmartReg experiment, both the regulator and reporting firms can take part in consensus through Raft. This means that both can propose new blocks and validate transactions. Initially, it was proposed that the regulator be the only entity that takes part in consensus. This was changed to provide better Crash Fault Tolerance within the system. To better understand this, we need to consider that regulators in the reporting ecosystem are a handful compared to the thousands of firms they regulate. This would mean that only a handful of peers would provide fault tolerancy for the entire network of potentially consisting of thousands of peers, reducing the resiliency of the network [9]. Therefore both the regulator and the firm has been allowed to take part in consensus.

If firms are to act dishonestly, this can easily be identified by the regulator as discussed above meaning that firms don't have strong incentives to act dishonestly within the network.

4.4 Implementation

This section presents the implementation details of SmartReg. It starts with a discussion of the tools, environments and frameworks used, followed by a discussion on how a role-based identity system has been implemented. It also discusses the important APIs implemented for Regulatory Reporting. It ends with a discussion with the implementation details of an automated deployment mechanism for each of the peers in the reporting network.

[9]What if the all the handful regulator peers go down?

4.4.1 Tools & Environments

The main motivation behind the choice of tools, languages and frameworks for the components in chapter 4.3.1 is to enable "frictionless" integration between SmartReg and internal banking infrastructure.

4.4.1.1 Backend

The entire backend has been implemented on the Java Virtual Machine (JVM) using Java 8 and the SpringBoot framework. The JVM and SpringBoot are widely used in the financial services industry for their backend systems [10]- writing the backend on the JVM would allow quicker integration of SmartReg with internal banking infrastructure.

4.4.1.2 Frontend

The Frontend component has been implemented using the ReactJS Framework. The intention of using ReactJS is as follows:

1. Its virtual DOM paradigm enables a higher level abstraction for quicker development and quicker rendering of frontends [33]

2. It enables code reusability and efficiency through reusable React components [33]

3. A uni-directional data flow model provides code stability, particularly when refactoring code [30]

4. It is being increasingly used by banks for their front end applications

4.4.1.3 API

The APIs have been written with the OpenAPI specifications to facilitate quicker integration between the internal banking systems and SmartReg.

4.4.1.4 Authentication Server

A third party tool has been used as an authentication server. This tool conforms to the 0Auth 2.0 protocol (and makes use of JSON Web Tokens, JWT) that is com-

[10]Validated by Santander and FCA

monly used for Single Sign On (SSO) based authentication within banks and enterprise [8].

4.4.1.5 Database

For the purposes of the SmartReg experiment, a NoSQL database (MongoDB) has been used. There is no particular reason for this (other than implementation ease), the interface that consumes the database has been implemented to consume either a SQL or NoSQL database.

4.4.1.6 Reporting Module and Reporting Functions

The entire reporting infrastructure has been implemented using Haskell. Haskell is a high-level functional programming language that is referrentially transparent by design- a technical requirement for the reporting module and functions.

4.4.2 Identity

In order to manage the identity of the clients that interact with the Spring Boot web server, we use Auth0, a platform that employs the OAuth 2.0 authorisation framework. This framework, which replaces the previous OAuth 1.0 protocol, authorises a third-party application to access data. From a high-level perspective, the OAuth framework empowers clients to gain access to specific resources by acquiring an access token, such as a JSON Web Token, instead of directly using the credentials of the resource owner. The authorisation server generates tokens after the resource owner approves it. The client may use the token in order to access the resource server. The access to the resource server is made possible through HTTP requests using the Transport Layer Security (TLS) protocol.

Particularly, in the context of SmartReg application, the client-side obtains a JWT from the Auth0 server and uses the token to access resources hosted on the Spring Boot web server, which in our context plays the role of both resource owner and resource server. It accomplishes this by incorporating the token into the header of the HTTP request (Figure 4.6).

```
GET contracts/id HTTP/1.1
Host: https://smartreg-ucl.herokuapp.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

Figure 4.6: HTTP request incorporating the token into the Authorisation header

Once the web server receives the JWT, it uses a number of different methods to verify the validity of the token by interacting with the Auth0 authorisation server. Thus, the token represents an abstraction that comes as an advantage for the resource owner, due to the fact that it removes the resource server's need to implement its own identity management system. By using the OAuth 2.0 framework, the resource server has to deal only with granting access to the authorisation server and verifying the validity of the tokens received from the clients. Figure 4.7 illustrates the workflow of this framework protocol and presents the communication flow between the client, resource owner, authorisation server and resource server (the resource owner and the resource server are one entity).

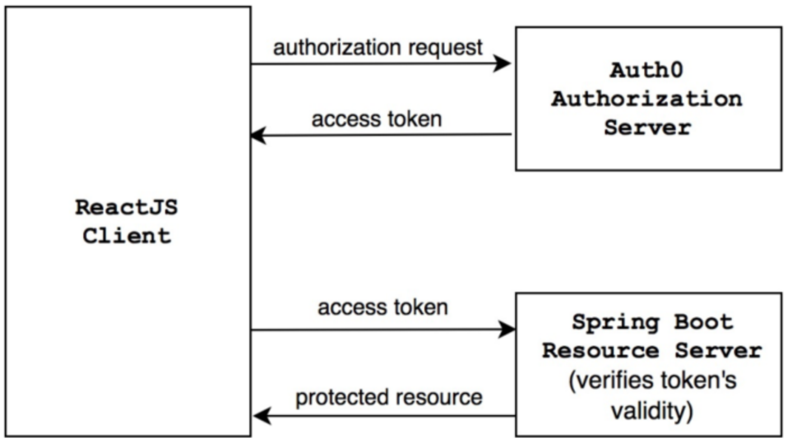

authorization request
access token
Auth0 Authorization Server
ReactJS Client
access token
protected resource
Spring Boot Resource Server (verifies token's validity)

Figure 4.7: The authorisation workflow used by the SmartReg application

The workflow between the three entities depicted in Figure 4.7 includes these stages:

1. The client requests an access token from the Auth0 server by sending credentials

2. The Auth0 server checks the credentials, and if valid, sends back the access token

3. The client uses the token to request the protected resources from the web server

4. The web server verifies the token, and if valid, sends back the requested resources

4.4.3 Experiment Implementation

The experiment outlined in section 4.2.2 has been implemented as in table 4.4. Figure 4.8 illustrates how the Solidity Application Binary Interface (.abi) has been generated from the Solidity Smart Contract code and "wrapped" as a corresponding Java class.

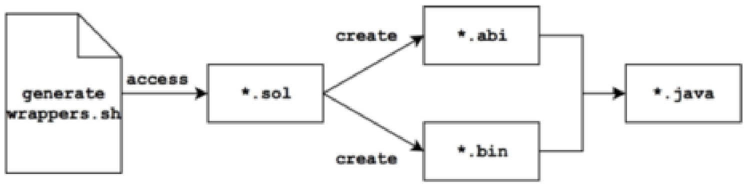


Figure 4.8: Generation of Smart Contract API wrappers

Figure 4.9 provides the inheritance logic for the institutions within SmartReg.

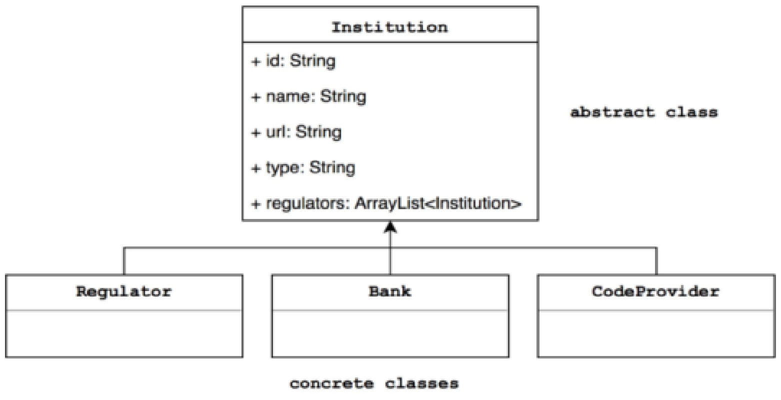


Figure 4.9: Inheritance structure of the Institution Java classes

Table 4.4: SmartReg Implementation details

Requirement	Implementation Details
1. Achieve data privacy by interacting with a private blockchain network	Implement the blockchain network using the Quorum private protocol
2. Contain three types of actors or participants Regulators, Banks and Code Providers	Implement Regulator.java, Bank.java and CodeProvider.java classes that inherit from the Institution.java parent class (Figure 4.9)
3. Manage six nodes: two Regulators, two Banks and two Code Providers nodes	Implement InstitutionController.java, a controller class that creates 6 Java objects by instantiating the Regulator.java, Bank.java and CodeProvider.java classes
4. Create a marketplace and a submission management tool, where code can be submitted by the Code Providers, traded by the Banks and approved, rejected or queried by the Regulators. Banks can only trade code that has already been approved by the regulatory institutions. In addition, before they make the submission, Banks must run the code.	Create a Spring Boot RESTful API that handles HTTP requests from the client server and uses Web3j to connect to the blockchain network Store the data on the Quorum blockchain and implement a caching system using MongoDB Each node must have its own web server and MongoDB instance Manage the identity of the participants by using the Auth0 Authentication system At least one Bank node must run code in a virtual deterministic environment using Docker Containers Implement the RunController.java class (Figure 4.11) that calls a BASH script to run code
5. Contain a screen to visualize the Code Inventory (submitted code) and the Submission Management Tool, where: • the Code Providers can submit new code to the inventory the Regulators can change the status of the code by using Smart Contracts • the Banks can submit new regulation reports through the Submission Tool • the Regulators can change the status of the reports by using Smart Contracts	Develop the Graphical User Interface (GUI) using React, a JavaScript library Implement two Java classes Submission.java and Code.java and two controllers SubmissionController.java and CodeController.java (Figure 4.10) Implement smart contracts using Solidity and automatically convert them into Java Wrapper classes using the Web3j library and BASH scripting (Figure 4.8) Manage the nodes by finding a feasible solution that manages the keys and permissions of the network Automatically deploy the Spring Boot web servers on a Cloud Computing platform
6. Comply to the following set of rules: • Code and Submission Inventories should be public to every participant node • Code can only be added by the Code Providers and the submission reports can only be submitted by the Banks • The status of the code and submissions can be updated only by the Regulators • Banks and Regulators can only see records related to them	The smart contracts and the way the Quorum protocol works will guarantee that the application respects these set of rules. Implement QuorumBlockchain.java (Figure 4.11) class that uses methods from the generated Java Wrapper classes and: • Connects to the Quorum network • Deploys smart contracts • Loads the deployed smart contracts

Figure 4.10 presents the class diagrams for the code and submission components, including their controllers.

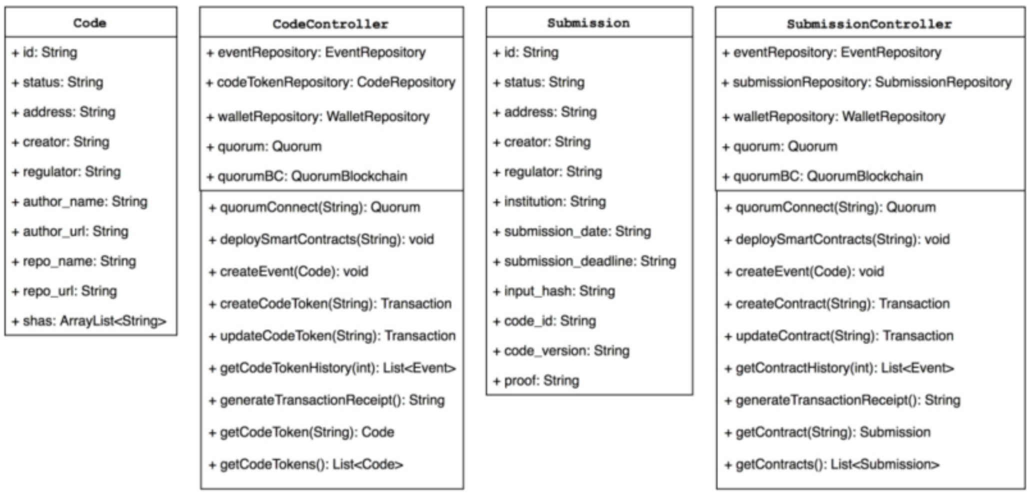

Code
+ id: String
+ status: String
+ address: String
+ creator: String
+ regulator: String
+ author_name: String
+ author_url: String
+ repo_name: String
+ repo_url: String
+ shas: ArrayList<String>
CodeController
+ eventRepository: EventRepository
+ codeTokenRepository: CodeRepository
+ walletRepository: WalletRepository
+ quorum: Quorum
+ quorumBC: QuorumBlockchain
+ quorumConnect(String): Quorum
+ deploySmartContracts(String): void
+ createEvent(Code): void
+ createCodeToken(String): Transaction
+ updateCodeToken(String): Transaction
+ getCodeTokenHistory(int): List<Event>
+ generateTransactionReceipt(): String
+ getCodeToken(String): Code
+ getCodeTokens(): List<Code>
Submission
+ id: String
+ status: String
+ address: String
+ creator: String
+ regulator: String
+ institution: String
+ submission_date: String
+ submission_deadline: String
+ input_hash: String
+ code_id: String
+ code_version: String
+ proof: String
SubmissionController
+ eventRepository: EventRepository
+ submissionRepository: SubmissionRepository
+ walletRepository: WalletRepository
+ quorum: Quorum
+ quorumBC: QuorumBlockchain
+ quorumConnect(String): Quorum
+ deploySmartContracts(String): void
+ createEvent(Code): void
+ createContract(String): Transaction
+ updateContract(String): Transaction
+ getContractHistory(int): List<Event>
+ generateTransactionReceipt(): String
+ getContract(String): Submission
+ getContracts(): List<Submission>

Figure 4.10: The UML diagrams of Code, CodeController, Submission and Submission-Controller

Figure 4.10 presents the class diagrams for the Quorum component.

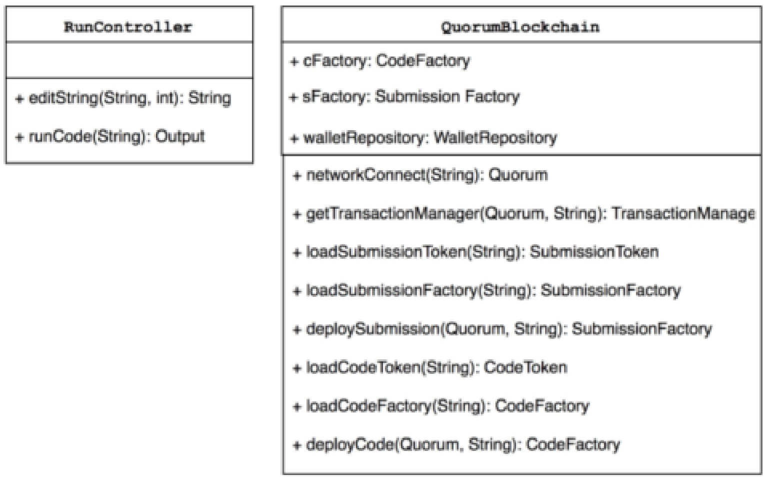

RunController
+ editString(String, int): String
+ runCode(String): Output
QuorumBlockchain
+ cFactory: CodeFactory
+ sFactory: Submission Factory
+ walletRepository: WalletRepository
+ networkConnect(String): Quorum
+ getTransactionManager(Quorum, String): TransactionManage
+ loadSubmissionToken(String): SubmissionToken
+ loadSubmissionFactory(String): SubmissionFactory
+ deploySubmission(Quorum, String): SubmissionFactory
+ loadCodeToken(String): CodeToken
+ loadCodeFactory(String): CodeFactory
+ deployCode(Quorum, String): CodeFactory

Figure 4.11: The UML diagrams of RunController and QuorumBlockchain

4.4.4 Deployment

4.4.4.1 Heroku Cloud Environment

Each of the peers in the SmartReg Experiment have been deployed on the Salesforce Heroku cloud environment. There is no particular motivation behind this other than cost. The deployment process on the Heroku platform is automated using BASH scripting. Automated deployment made the whole deployment process faster and more efficient, offering ease during testing and replication.

In order to understand how Heroku works, one must understand what a dyno is. Dyno is a specific name given by the Heroku community to represent a lightweight Dockerised Linux container. Every time an application is deployed, it runs inside a dyno container. Dynos are maintained and managed by an entity called the dyno manager. Each dyno has its own ephemeral file system that contains the latest version of the deployed code. The processes that run inside a dyno can use this file system, but any files written are erased when the dyno is restarted or stopped.

The architecture of the SmartReg application imposes that each institution has its own web server deployed at a different URL. Figure 4.12 illustrates how the automated deployment is achieved in the context of the SmartReg application. Each Dyno is a separate lightweight Linux environment that runs a SmartReg peer to replicate the network topology found in figure 4.2

Although Heroku makes the deployment process simple and efficient through the CLI, dynos are limited and they do not allow the applications running on top of them to make changes to the file system. Given that each of the bank entities must be able to run reporting functions that are Haskell applications, a separate environment is needed to run these. In order to address this limitation, the bank entities had reporting functions deployed on the Digital Ocean Cloud Platform.

4.4.4.2 Digital Ocean Cloud Environment

The bank entities additionally have their reporting functions deployed on Digital Ocean. It runs onto an Apache Tomcat 8 server that is deployed on a Linux virtual machine, which uses the Ubuntu 16.04 x64 distribution. The virtual machine is configured so that the process created by the Spring Boot web server, when the client runs code, has root privileges. Thus, the process is allowed by the virtual machine's operating system to call a BASH script. The script performs a computation and returns its result to the web server, which ultimately sends it back to the client server (Figure 4.13). Figure 4.12 presents the automated deployment on the Heroku Cloud Platform via a single heroku-deploy.sh shell script.

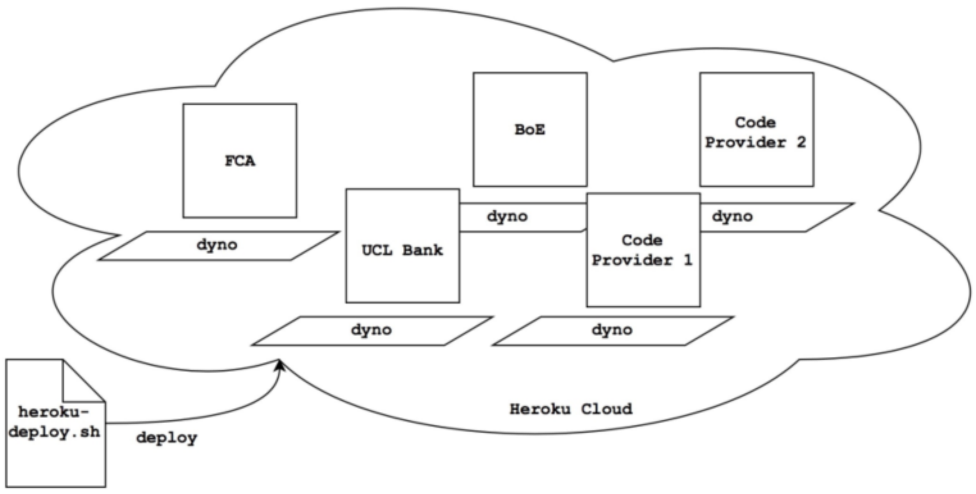


Figure 4.12: Automated deployment on the Heroku Cloud Platform

Figure 4.13 presents the deployment of reporting functions on Digital Ocean Droplets.

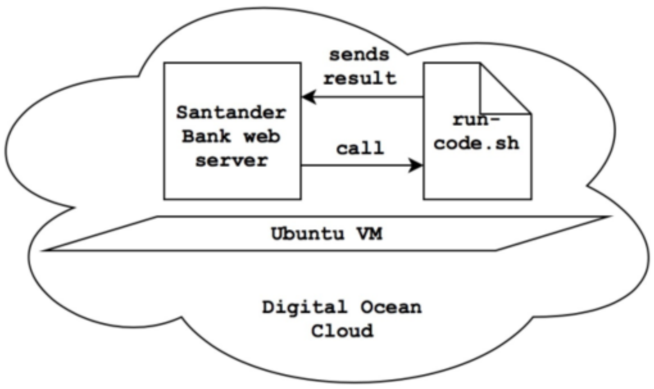


Figure 4.13: Reporting Functions on Digital Ocean Cloud

4.5 Testing & Results

4.5.1 Testing

Testing is a crucial part of the development lifecycle, particularly when it comes to applications like SmartReg which touch sensitive data. As such it acts as a safety net, helping avoid the unwitting deployment of broken code. In SmartReg a number of tests were carried out.

4.5.1.1 Unit Tests

Unit tests check for the validity and correctness of an individual unit. A unit can be arbitrarily defined, but is typically a function or method. Well tested applications employ many unit tests to assert various properties across the codebase.

The smart contract code has been developed using the Truffle framework, which supports the testing of contracts locally. This means unit tests can be run on the developer's machine, without having to deploy contracts to the network and then run tests on those deployed contracts. Meanwhile, the frontend codebase uses Jest, a testing platform, to run the test suite.

4.5.1.2 Deployment and Manual Testing

Whilst testing local versions of the codebase captures a wide range of errors, it has been equally important to test a deployed version of the application. Particularly because deployment environments vary from the development environment. Additionally, new constraints, such as CORS and SSL certificates, come into play each of which can cause unforeseen integration problems.

Early on in the development of SmartReg made use of continuous integration pipelines to manage deployment of code into the production environment, and regularly updated that code to match the local copies. This formed part of our continuous development process, and meant that deployment bugs could be addressed as they rose.

4.5.1.3 Iteration

SmartReg has evolved substantially throughout the course of its development, adapting to new constraints, feature requests and the continually shifting blockchain ecosystem. This section describes how the final solution was reached through an iterative process of continuous feedback from the collaborators.

Initially, it was planned for banks to upload a full, plain-text record of their financial metadata in their submissions, relying on the quorum network for orchestration and audit only. This had several issues, however. First, after initial talks with Santander it became clear that, from a legal perspective, Santander would not be prepared to store copies of sensitive metadata on a shared network outside of its firewall, even when that data was private to itself and its regulators. Secondly, the storage requirements necessary to facilitate all banks uploading full financial metadata for every submission did not scale well, and would have led to undue storage costs through a bloated blockchain.

To solve these problems, the property of transaction replay by way of deterministic functions was used to devise the system which has been described in the chapter. Instead of storing plain-text metadata, the SHA256 hash of that data is stored as a persistent identifier. This identifier can be used by the permissible entity (ideally the relevant regulator) to view the underlying data from the bank. This solves both the privacy and storage problems, since all hashes are 128 characters long and are boundedly intractable to reverse, meaning that the plain-text source cannot be recovered from the hash alone. This new approach is better for users of the system, as they don't upload plain-text metadata to an external network, and offers improved scalability for the system as a whole.

4.5.2 Results

The experiment describes in section 4.2.2 was carried out [11]. In order to identify whether the system is capable of almost real-time data reporting each of the reports (PSD and CET) were run 1000 times on the infrastructure. The runs were randomly allocated between a bank and a regulator or a pair of regulators. The results have been presented in figures 4.14 and 4.15.

[11] Some screenshots of the UI have been provided in Appendix E.1

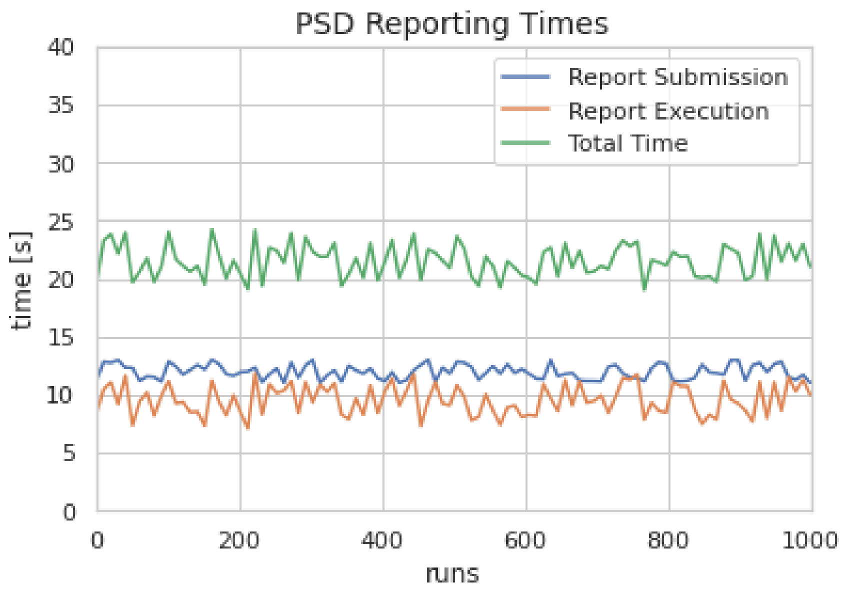


Figure 4.14: Test times for the PSD Report Execution and Submission

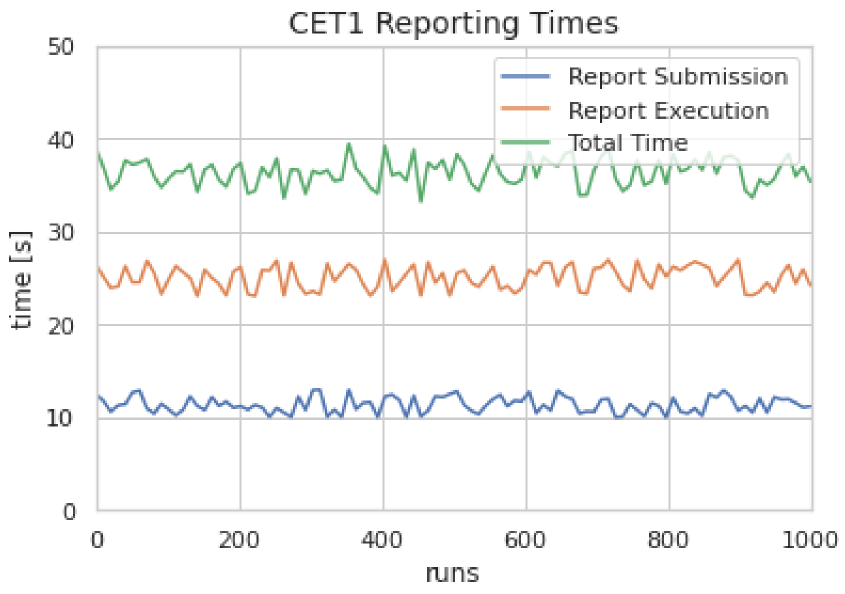


Figure 4.15: Test times for the CET Report Execution and Submission

The report execution times are the time (in seconds) it took to run a machine executable regulatory report (i.e. the reporting function). The report submission time is the time it took to submit this report to the regulator after execution of machine executable report. The total is the sum of these two times- the total time it takes to run a given report i.e. submit data to the regulator from the time of request by the regulator.

Table 4.5 provides the average of these times from all 1000 runs. In both cases the submission times are about 12 seconds. This is expected as the average block time on Quorum is set to 10 seconds. This factored with some additional latency due to communications across peers results in the average. The execution times are dependent on the reporting functions, the execution time of CET is greater as the number of data fields it has to run against is far greater than its PSD counterpart. Although not quite "real-time" regulatory reporting SmartReg provides a much quicker reporting time (in seconds) compared to the average 1 month it takes currently [12]- a significant improvement that is closer to the goal of real-time reporting.

Table 4.5: Contents of a bank's submission

Report Type	Execution Time [s]	Submission Time [s]	Total Time [s]
PSD	9.80	12.11	21.91
CET	25.04	11.35	36.40

4.6 Conclusions

SmartReg has explored the design of a new approach to financial regulation. It combines traditional banking infrastructure with a shared distributed ledger to offer a hybrid system that combines the benefits of both technologies. By leveraging Quorum, a version of Ethereum that supports private state, data can move within private channels on a shared network, offering privacy and permissioning between competing entities on the same network.

Our implementation sees each submission storing a hash of the financial data input, the output and the link to the regulatory function. This approach has the advantage of storing minimal data, and thereby keeping the network's file size small, but

[12]Determined by the collaborators

means that raw financial input is not stored on the blockchain, necessitating the request of this data from the banks by regulators. Future work may see an alternative approach used, depending on the desired features of the system. An implementation where some raw financial data is stored on chain may be preferable, and worth higher storage costs, to leverage a greater level of automation, for instance.

Codified regulatory functions (i.e. machine executable regulations) shift the onus of regulatory interpretation to an objective, pre-approved function. This offers non-subjective interpretations of regulation, enabling automated submissions by banks and verification by regulators which delivers speed and cost improvements. The work in SmartReg has wide reaching impact[13] through the following benefits of its approach:

1. Easier regulatory compliance
2. More precise and agile regulation
3. Accurate and real-time information
4. Standardisation
5. Improved systemic risk control

Whilst we proposed the core concepts of SmartReg, there are still areas for extension. It is our hope that future work, will take the ideas described in the project and yield a compelling, production ready solution for modern financial compliance. First, there exists the issue of standardisation. The regulatory functions rely on the assumption that certain standards exist to enable their successful execution across a variety of environments. These assumptions include the existence of a standardised interfacing for running regulatory functions that enables execution support across all supporting nodes. Also, the output of the functions should conform to some standard format, such that the interpretation of the function result can be automated. These standardisation efforts should possibly exist as an open source specification,

[13]Confirmed by the collaborators, FCA and Santander

so that members of the community can propose changes and raise concerns.

The design decisions made in SmartReg have informed a wider project carried out by the regulator and a number of banks. This is the Digital Regulatory Reporting Project (DRR). DRR has become a priority for the regulator, banks and the UK financial services industry [12, 13]. This has been the greatest impact of the work presented in this chapter.

Chapter 5

RegNet: Using Federated Learning and Blockchain for Privacy Preserving Data Access

The chapter discusses new techniques in Federated Machine Learning, Data Privacy and Trusted Computing that can be used for the purposes of compliance with data regulation. It also explores the use of Differential Privacy for privacy preserving machine learning. The work in this chapter has been carried out as part of an InnovateUK Grant, ReguIAItion and a consortium of financial institutions, law and accountancy firms, The Open Data Institute as well as the regulator. The chapter begins with the growing problem of regulating data and algorithms. It then provides the design and implementation of a framework for this-RegNet. The chapter ends with a discussion on the results and impact of the application of this framework to a use case. Due to the commercial sensitivity of the work undertaken, the chapter provides the relevant scientific details of the work undertaken and does not explore finer technical details of the work undertaken[1].

[1]These can be clarified in the examination viva of this thesis.

5.1 Introduction

The debate around Data Regulation is amplifying and the need for policies and mechanisms to manage data compliance, governance and regulation is growing [89, 81, 38, 82]. There are growing political, commercial and social challenges concerning data that have resulted in the need for the Algorithmic Regulation of data [89, 46, 4, 60]:

- Data harvesting - the process of extracting and analysing (personal) data on users from online interactions. Leading companies include Google, Facebook, Amazon, Tencent, and ByteDance TikTok. They collect comprehensive personal and interaction data on users and their network of contacts; then use sophisticated machine learning algorithms for 'deep' behavioural and predictive analytics.

- Data ownership - who own the data and the right to exploit it. It covers issues of ownership, stewardship and custodianship; responsibility for data content, context, safe custody and usage.

- Data privacy - ensuring that the data shared by clients is only used for its intended purpose; and the right of individuals to have control over how their personal information is collected and used.

- Data collaboration - spans: a) internal companies in a group or departments in government; b) consortia groups of companies partnering in analytics and business; and c) international organisations such as financial regulators (e.g. AML), law enforcement (e.g. drug cartels) or security organisation (e.g. terrorist groups).

- Data security - means protecting digital data from destructive forces and from the unwanted actions of unauthorised users, such as a cyberattack or a data breach, that may compromise the integrity, availability, confidentiality or privacy of the data

- Data legislation - controls how personal or customer information is used by organisations or government bodies. A prominent example being the EU General Data Protection Regulation (GDPR) covering data protection and privacy, plus transfer of personal data outside the EU.

- Data sovereignty - the idea that data is subject to the laws and governance structures within the nation it is collected; central to competition, taxation, security and economic supremacy.

Whereas collaboration, ownership and harvesting are all challenges that deal with Data Management (including analytics, application and administration of data). Legislation and sovereignty are challenges deal with Data Governance. Security and privacy challenges deal with the assurance and safekeeping of data. All three factors: Data Management, Governance and Security need to be considered when regulating data [74]. The fundamental research question of this chapter is based on the premise of all these three factors: How can we better manage and govern data in a secure, privacy preserving manner for algorithmic purposes?

5.1.1 The Problem of Analytics, Data Sharing and Compliance

With huge and ever-growing amounts of data for analysis, organisations are faced with three major challenges [74]:

1. Data ecosystems comprise of distributed and isolated data sets that need to be moved around for analytics

2. Analytics requires models to be trained across these independent data sets

3. Data sovereignty/privacy legislation is making the collection, sharing and analysing of data increasingly difficult

Organisations and industry recognise the utility in collaborating with each other for the purposes of analytics but also recognise that data regulation and compliance, when sharing this data, makes it challenging to do so [74]. Fundamentally the problem is one of sharing data, we argue that data does not need to be shared for the purposes of analytics and that trusted data access can achieve the same goal whilst maintaining compliance. Whilst sharing explicitly involves the trading of data between owners/custodians and counterparties, data access involves making data available for algorithmic purposes. Trusted data access facilitates this in a secure and privacy preserving manner for a singular-algorithmic purpose only. The aim of this chapter is to provide an infrastructural framework for this trusted data access. This is significant as data policies such as GDPR can be complied with as there is no sharing of data and individuals' privacy within the dataset is preserved [87].

5.1.2 Objectives

The following objectives were established with the collaborators to realise this aim:

1. Identify relevant technologies and fields of research in data science and information security to support trusted data access for a singular-algorithmic purposes

2. Architect a framework based on these technologies that facilitates data access for industry applications

3. Implement, analyse and iterate over the design of this framework and its emerging infrastructure

4. Test and validate this infrastructure with the collaborators

5. Analyse the proposed solution and its efficacy in governing data in a secure, compliant manner for algorithmic purposes

We establish 2 core aspects for the development of such an infrastructure:

1. **Federated Data Infrastructure**- privacy-preserving data infrastructure; a framework for collaboration, allowing secure communication with collaborating parties, such that 'raw' data does not leave the owner.

2. **Federate Machine Learning**- decentralised training of a machine learning model which enables collaborative learning while keeping data sources in their original location[2].

The subsequent chapters explore the technologies and techniques used to achieve this.

5.2 Background

This section provides the relevant information on the use of Federated Learning (FL) for RegNet, it provides an overview of FL, type of FL and a taxonomy of this emerging sub field of Machine Learning. The taxonomy is one of the contributions of this thesis.

The second half of this section provides an overview of some of the Privacy Enhancing Techniques used in PPML.

5.2.1 A Taxonomy for Federated Learning

To provide a perspective we next look at categorising federated learning (see in figure 5.1):

- Communication - communication or control of federated analysis: a) centralised learning - a central server orchestrates the different steps of the algorithms and coordinate participating nodes during the learning process; b) decentralised Learning - the participating nodes coordinate themselves to obtain the global model.

- Data - by data partition: a) horizontal federated learning - homogeneous data sets have the same feature space but distinct sample spaces; b) vertical federated learning - heterogeneous data sets with different feature spaces but the same sample space; and c) Federated transfer learning - here data sets differ not only in samples but also in feature space.

[2] For example, Google's mobile phone users benefit from obtaining a well-trained model without sending their personal data to the Cloud.

- Federation - the scale of federation of nodes: a) multiple nodes - a large number of On-device nodes, such as smart phones, each with a relatively small amount of data and processing power; b) major nodes - a small number of major inter-organisation nodes, such as data centres, each with a large amount of data and processing power.

- Security - the data privacy preserving techniques employed. Popular for federated learning are: a) Secure Multi-Party Computation SMPC - a subfield of cryptography with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private; b) homomorphic encryption - the conversion of data into an encrypted form that can be analysed and worked with as if it were still in its original form; and c) differential privacy - a system for sharing information describing the group patterns within a data set while withholding identifiable 'raw data' about individuals in the dataset. These are discussed further in the section below.

- Machine learning - the emerging federated machine learning models are variants of traditional models. Examples include: a) deep neural networks- networks multiple layers between the input and output layers; and b) gradient boosted decision trees - involves three elements: a loss function optimisation, a weak learner for predictions, and an additive model for minimizing the loss function.

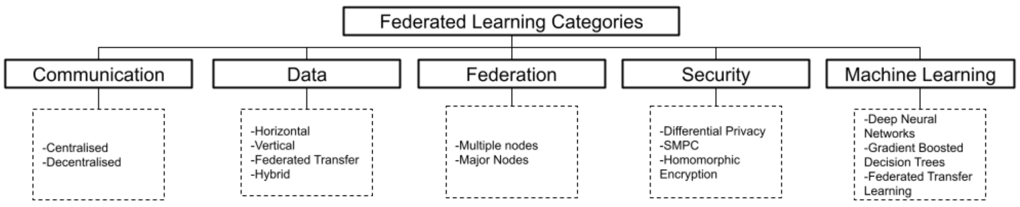


Figure 5.1: Federated Learning Taxonomy

5.2.2 Communications and Control Architecture

Federated Learning is, in part, a problem of orchestration- how do we orchestrate the training of a model across multiple federated learning nodes and their datasets.

By abstracting away orchestration from training/ learning it is possible scale and modularise FL infrastructure from the Machine Learning problem [3]- we could have 2 ML problems, each using a different type of model architecture (e.g. Neural Nets vs. Decision Trees) but they can both use Trusted Aggregation, meaning that their orchestration is coordinated out by a trusted aggregator node. Given that orchestration relies on communcations across peers it is important to understand common communication architectures such as the use of a trusted centralised node (i.e. server) to orchestrate learning; or a collection of decentralised nodes coordinating themselves to obtain a global model. Examples include FedAvg and SimFL [87].

5.2.2.1 Centralised Federated Averaging (e.g. FedAvg)

With centralised learning the trusted node aggregates the information from the other nodes and sends back training results (e.g. gradients or model parameters) to the participating nodes. Communication or control between the nodes can be synchronous or asynchronous.

5.2.2.2 Decentralised GBDT FL (e.g. SimFL)

With decentralised learning communications are performed amongst the nodes and every node is able to update the global model parameters directly.

5.2.3 Data Partition

Federated learning systems are frequently classified by their data partition into how the data sets are distributed across the nodes [87]; namely the sample and feature spaces:

- Horizontal federated learning - homogeneous data sets share the same feature space but have different in samples;
- Vertical federated learning - with feature-based learning multiple heterogeneous data sets share the same data space but differ in feature space;
- Federated transfer learning - here data sets differ not only in samples but

[3]It could potentially also help address open problems such as FL crash tolerancy as tolerancy becomes a problem of nodes and communication across them

also in feature space with only a small portion of the feature space from both parties overlaps; and

- Hybrid federated learning - used a combination of horizontal and vertical data partitions.

Figure 5.2 provides a graphical representation of this.

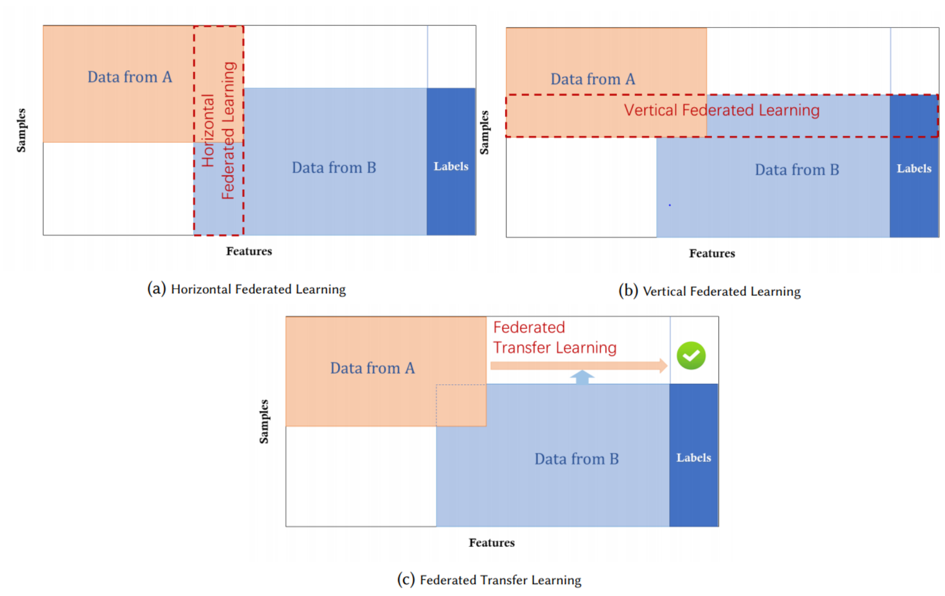

Data from A
Horizontal Federated Learning
Data from B
Labels
Samples
Features
(a) Horizontal Federated Learning
Data from A
Vertical Federated Learning
Data from B
Labels
Samples
Features
(b) Vertical Federated Learning
Federated Transfer Learning
Data from A
Data from B
Labels
Samples
Features
(c) Federated Transfer Learning

Figure 5.2: Federated Learning Characterised by Data Partition [86]

Data can also be categorised by the ‘unbalancedness’ (i.e. non- Independent and Identically Distributed) of local data samples: a) covariate shift - local samples have different statistical distributions; b) prior probability shift - local nodes may store labels that have different statistical distributions; c) concept shift dividing into: i) local nodes share the same labels but have different features, and ii) local nodes the same features but different labels; and d) unbalancedness - the data available at the local nodes may vary significantly in size.

5.2.4 Federation of Nodes

Classification by federation ranges from a) multiple nodes - a large number nodes each with a relatively small amount of data (e.g. smart phones, IoT devices); to b) major nodes - a small number of powerful nodes, each with a large amount of data (e.g. data centres).

Federations of multiple or major nodes can span a single product line or company, or a small number of major organisations collaborating.

5.2.5 Security & Privacy

Federated Learning is a form of Privacy Preservation Machine Learning Techniques (PPML). This is not to be confused with secure Machine Learning (secure ML). The fundamental difference that Secure ML assumes that the adversary breaches the integrity and availability of the system whereas PPML assumes that the adversary breaches the confidentiality and the privacy of the system [87]. Where:

1. Integrity- an adversarial attack on the integrity means the veracity of the system is compromised e.g. the system may false negative outputs by the system as normal

2. Availability- a systemic adversarial attack that renders the system unusable and can lead to classification errors

3. Confidentiality- an adversarial attack results in the leakage of sensitive information e.g. training data

4. Privacy- an adversarial attack results in the leakage of identifiable and attributable information e.g. identifiable features such as person name or company name in a data set

A number of Privacy Enhancing Techniques (PETs) can be used individually and in ensemble (Differential Privacy and Secure Multi Party Computation, SMPC). This section explains some of the techniques, for the case of RegNet only Differential Privacy has been employed[4]. The next subsection discusses these in detail.

5.2.6 Secure Multiparty Computation

Also known as Secure Function Evaluation [88], it involves jointly computing a function from the private input by each party without revealing the value of these

[4]It is expected that future work will involve the development of SMPC and the two in ensemble

private inputs to other parties. In ML terms this function could be a model's loss function during training or the model itself (during inference). Cryptographic schemes such as Oblivious Transfer [47] and Threshold Homomorphic Encryption [27] schemes can be used to facilitate SMPC. However, the most common scheme currently used in PPML is Secret Sharing schemes [72, 29].

Secret sharing schemes involve the hiding of a secret value by splitting it into parts and randomly distributing it to the parties involved in the multi-party compute such that each part has only one share of this secret. A threshold number of these individual shares is needed to fully reconstruct the entire secret. Arithmetic secret sharing is the most commonly used in existing SMPC PPML systems.

Most SMPC based PPML consists of two parts: an online and offline phase. The offline phase involves the bulk of the cryptographic operations such as the generation of triples. The online phase involves the training of the ML Model (e.g. using the triples generated in the offline phase). SMPC based PPML is commonly used for a number of reasons [62]:

- It is less computationally expensive than Fully Homomorphic Encryption
- It isn't vulnerable to computationally powerful adversaries
- It can be used to perform inference directly on encrypted data i.e. without allowing the model owner to see the private data of the owner

It however has a number of limitations:

- There is a networking and communications overhead
- It assumes that individual parties are not colluding or at most the n parties are colluding such that $n <$ threshold distributed secret shares, assuming each party has at most one share.

5.2.7 Homomorphic Encryption

Homomorphic Encryption (HE, first proposed by Rivest et al. [67]) involves the direct computation over a ciphertext, without decrypting the ciphertext. There are a number of HE schemes, often bucketed into three: Partially HE schemes, Somewhat HE schemes and Fully HE schemes that have been used for PPML (their computational complexity grows as the HE functionality grows). These include the use of Paillier's Scheme in the training of logistic regression models through secure gradient descent [41] or the use of secure inference of encrypted queries over trained neural networks [37]. The inference by clients is classified securely by the neural network without inferring information from the query itself.

The biggest advantage of HE is that it can be used to perform inference directly on encrypted data without revealing any information

It however has the following limitations:

- HE schemes are extremely slow ad require large computational and memory overheads compared to other techniques
- HE schemes are restrictive as a limited set of computational operations can be performed, these might not restrict the degree to which they can be used for ML

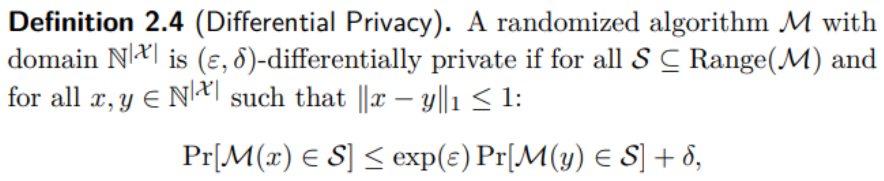
5.2.8 Differential Privacy

The premise of Differential Privacy (DP) is to confuse an adversary (in the case of FL, an algorithm) that may be trying to access individual information from a database such that they cannot distinguish any individual-level sensitivity from it. In the PPML context DP can be used at the local level (Local Differential Privacy, LDP). Each party perturbs their dataset and releases this perturbed data for model training during FL. DP schemes are usually considered to have a relatively smaller computational overhead compared to SMPC/ HE based schemes and a number of different techniques have been explored [64, 53].

In [32], DP is qualitatively described as:

" Differential privacy describes a promise, made by a data holder, or curator, to a data subject: You will not be affected, adversely or otherwise, by allowing your data to be used in any study or analysis, no matter what other studies, data sets, or information sources, are available."

Formally Differential Privacy is defined as:


Definition 2.4 (Differential Privacy). A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta,$$

Figure 5.3: Formal Definition of Differential Privacy [32]

Where $\mathscr{M}$ is the randomised algorithm consuming the dataset for a given query; $\mathscr{S}$ all the outputs of $\mathscr{M}$ that can be predicted; x are the entries in the dataset and y are the entries in the parallel dataset.

The terms ε and δ are the most significant terms in the definition. ε refers to the maximum distance between a query on one dataset and the same query on its parallel dataset and δ refers to the probability of any information being leaked. It is often referred to as the "privacy budget"- the amount of "privacy" the owner of the dataset wishes to "spend" on the query. The smaller the values of ε the more privacy (noise) that is added to the response of the query making it difficult to construct the true value from the query. The trade-off with using a smaller value of ε is that the results from the query could be distorted (less reflective of the true value).

δ is dependent on the size of the dataset and caters for the privacy of "significant few" outliers in the dataset. For simplicity in the experiment within RegNet, we consider this value to be 0 and only consider ε differential privacy.

As the algorithm "asks" the dataset more queries, it spends "privacy" of the dataset. Therefore it is important for the dataholder to put an upper bound on the allowable limit to the number of queries w.r.t maximum ε spend that can be offered to the algorithm. RegNet provides a means to keep track of this privacy budget loss spend per dataset- offering a means to quantitatively keep track of privacy of the dataset.

5.3 Systems Design

Whereas the previous two sections explained techniques and purpose of FL and PETs (the first core component of RegNet) in detail, this section will present the design of a Federated Data, Audit and Orchestration infrastructure. Familiarity with blockchain concepts (from the previous chapters) is assumed.

5.3.1 Concepts

In this subsection we introduce the main concepts underlying RegNet. Conceptually RegNet is a framework to orchestrate computations within a peer to peer network of different nodes that could be data holders, algorithm providers or consumers of a machine learning model. These peers communicate within Private Channels over a number of Assets [5] under the constraint of the Channels Governance regime- a concepts central to RegNet.

5.3.1.1 Network

The Network refers to a collection of individual nodes that communicate with each other over one or a number of private channels. The term RegNet is used to refer to the network, along with its peers, channels and their governance. It has a peer-to-peer architecture orchestrated through a distributed ledger.

5.3.1.2 Nodes

Nodes are standalone compute and storage resources that run a RegNet peer. Reg-Net peers could be composed of organisations that are data holders, algorithm

[5] Channels and Assets are defined in the subsections below

providers or consumers of a machine learning model. By design, it is the responsibility of independent organisations to maintain and control their respective nodes [6]. Individual users are authenticated by way of their organisational nodes. The nodes in turn are authenticated through a node level CA (certificate authority) architecture [1]; individual users are not personally identifiable within the node (and by implication the Network/Channel level). To that end, for the remainder of this chapter, we will refer to users, organisations, institutions as Nodes distinctly within RegNet. The term peer will also be used interchangeably. All peers/ nodes are part of the global channel.

5.3.1.3 Channels

Channels are a monadic construct for the platform; all computations and governance happens by way of them. They are the basis for everything that happens within the platform, by design any sort of interaction that happens within the platform has to carried out via a channel. In fact, RegNet itself could be thought of as one single network level channel, all the other channels are a level of abstraction below these channels that implement their own governance over the network.

Channels can be composed of:

1. Unary channel- a single node is in a private channel with itself
2. Binary channels- channels composed of 2 nodes
3. Multitenant channels- channels composed of a number of nodes n, where n¿2.

5.3.1.4 Assets

Assets are tokenized artefacts that operated within RegNet. Tokenization is a key part of how RegNet deals with managing algorithms, datasets etc. on the platform. Each Asset Type is a tokenized [7] artefact with its own deterministic finite state machine that runs on the distributed ledger. They are implemented using smart

[6]RegNet is a P2P system
[7]We present Tokenization in greater detail in Appendix D

contracts within the platform. Assets manage the governance, orchestration and access of their underlying artefact. Assets include:

1. Datasets- These are to the individual collections of data for federated learning or inference. They may contain sensitive information. By design datasets within a common channel are required to have a standard format. Datasets need to be registered as an Asset before being enabled for use.

2. Algorithms- These are federated learning algorithms that train a model on a given Dataset for a given task. It could specify a number of parameters including the model type, architecture loss functions, neural network hyper-parameters and the parameters that are tuned during a training.

3. A model is a collection of parameters from a training- they could be collection of weights. It is the product of the computation between an algorithm and a dataset or a collection of datasets. In the case of neural nets, it refers to a collection of weight parameters for the connections of the neural net.

5.3.2 Distributed Ledgers

5.3.2.1 Motivation

By design FL is a collaborative activity that has a number of open problems:

- How do you administrate the actions of the collaborators?

- How do you manage the interactions between the collaborators?

- How do you enforce acceptable behaviour (or unacceptable behaviour)?

- Who manages the governance of the training process?

- Which entity manages the trusted actions (such as trusted aggregation in FL) in the channel?

- How do you ensure integrity of the entire FL process?

Traditional systems have a trusted intermediary that carries out these roles. For instance, in the case of the FL use case by Google (Google keyboard), Google has been the trusted intermediary that manages these issues [53]. Trusted intermediary based systems such as these have stronger integrity but do not scale well; they continuously rely on individuals/organisations to assimilate the "trust risk" whilst relying on them for the administration and enforcement of an FL process. We use distributed ledgers to remove this need for trusted intermediation.

5.3.2.2 Purpose

In RegNet, a distributed ledger is used for three main functions:

1. Auditability- maintain a traceable record of all the activities by the nodes within the channel

2. Governance- the permissioning, administration and use of assets within the network with respect to channels

3. Orchestration - the execution of an FL task across a channel (this becomes more apparent when it comes to activities such as trusted aggregation)

RegNet has a P2P architecture, where peers keep in sync via a distributed crash-tolerant consensus. Hyperledger Fabric has been used to facilitate this [3]. A RegNet node is a HL fabric instance with a number of further design choices:

Consensus

Raft has been used as the consensus mechanism for a number of reasons [68]. These include its simplicity, pragmatism and scalability [43]. A Hardened Byzantinian Fault Tolerant consensus is not needed for RegNet as the network itself is a vetted network where the identities of the peers are known beforehand and a strict vetting process is carried out before allowing them to join the network (each peer additionally stakes his/her reputation) [43, 57]. As such all the limitations of Raft are subsumed here as well, for instance there need to be mechanisms in place to manage the collusion between two followers.

Identity/ authentication

The HL Fabric model of using TLS certificates is used to manage and maintain identities. The reasons for doing this are simple: all enterprise organisations already use trusted Certificate Authorities (and its underlying web of trust) to vet their digital infrastructure [3], we just bootstrap onto this. Certificates also offer a hierarchical abstraction of identities which can be at the organisational level as well as the individual/user level thus enabling Certificate based RBAC (roles based access control).

Roles

There is only one peer type that is created in RegNet. The responsibilities of this individual include:

- Ability to register assets
- Ability to maintain the governance of assets
- Take part in channel governance
- Take part in channel consensus

As the protocol matures, these responsibilities will be split and peer roles with more defined responsibilities will be created, according to the organisations needs.

5.3.3 Channels

Channels are the building block of RegNet, by design every allowable interaction with an Asset type on the network has to happen via a channel. Similarly all Asset types need to be designed with channel level interaction and channel level policies in mind. Channels should be used for purposes such as interaction with collaborators, to create audit trails, manage governance, manage access etc. The failure to use channels, for instance to maintain Assets, will mean that the Asset is not recognised by the protocol and is therefore invalid. Figure 5.4 presents the architecture of one such RegNet channel between three organisations.

If you cannot see your use case using a channel, you do not require RegNet; you will be adding unwanted complexity to your technical stack. The "REGNET-ServiceApp" tool is a front end abstraction of the concept of channels and provides an interface to manage Channels and Assets associated with it.

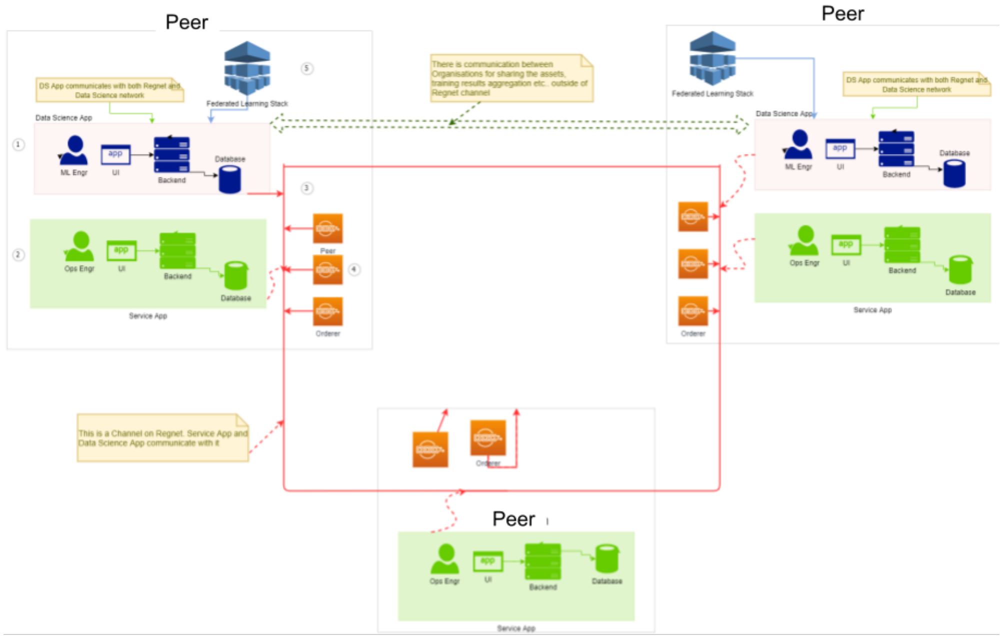

Peer
Federated Learning Stack
5
DS App communicates with both Regnet and Data Science network
Data Science App
1
ML Engr
app
UI
Backend
Database
2
Ops Engr
app
UI
Backend
Database
Service App
3
Peer
4
Orderer
There is communication between Organisations for sharing the assets, training results aggregation etc.. outside of Regnet channel
Peer
Federated Learning Stack
DS App communicates with both Regnet and Data Science network
Data Science App
ML Engr
app
UI
Backend
Database
Ops Engr
app
UI
Backend
Database
Service App
Orderer
This is a Channel on Regnet. Service App and Data Science App communicate with it
Orderer
Peer
Ops Engr
app
UI
Backend
Database
Service App

Figure 5.4: Channel level architecture of RegNet

There can be two kinds of channels:

1. Inter-organisational- These are channels that exist across one or more organisations that come together to form a consortium. They are used to manage Assets, access policies and governance for the consortium. In order to maintain simplicity and information security, separation of concerns should be considered in that a channel should only exist for one specific use case; another use case with the same collaborators should be managed on a different channel

2. Intra-Organisational- channels, although organisational, can be made at the individual level (assuming there is valid PKI/ CA infrastructure in place to facilitate this). This can be used to create channels across departments within

the same organisation that may have internal policies that may restrict the data access otherwise

RegNet also provides unary channels. Unary channels are an oddity in that they are channels with no one. These are allowed on RegNet for a number of reasons:

1. Testing Federated Learning Infrastructure e.g. by Data Science companies before being made available to others

2. Testing the management of Assets and Assets Types

3. Testing policies

5.3.3.1 Channel onboarding process

1. Organisations, wanting to provide data access, agree on the terms and policies outside the network

2. They nominate an organisation to create the channel

3. This entity that creates the channel, by default is the channel admin (and the ordering service)

4. They add the organisations' nodes; with the choice of making some (or all) the Orderers

5. The choice of adding/expelling organisations to/from the channel has been left to the channel admin and or the channel organisations

6. The channel is used for tasks such as FL training

5.4 Implementation

This section will discuss how the concept of RegNet has been implemented.[8]

[8]Due to the commercial sensitivity of the project only some aspects have been presented. The other parts of the implementation will be discussed in the examination viva of this thesis.

5.4.1 Distributed Ledger and RegNet Peer

Hyperledger Fabric (HLF) has been used to implement the Distributed Ledger component. This section explains how it works at a high level and how some of its parameters are configured for RegNet's purposes.

5.4.1.1 Transactions

The interactions between nodes across the network are transactional; they happen on a per transaction basis. Each node must submit a transaction to interact with any part of the distributed ledger, whether it is a simple smart contract read query or a smart contract write.

A transaction flow is as follows, this is partly similar to the transaction flow in HLF:

1. A Node submits a transaction e.g. to register an Asset

2. Endorsing nodes independently verify the validity of the transaction (is it sent by the right person (i.e. valid signatures)?, is the initiating entity permitted to carry out the transaction?, is he/she part of the channel?)

3. Endorsing nodes independently verify the transaction and run the smart contract/ code that is attached to the transaction to generate a set of computed outputs (read-write transaction set generation)

4. The node that initiated the transaction can choose to inspect the read-write sets from all the endorsing nodes

5. The endorsement policy of the channel is enforced by the node that submitted the transaction

6. If everything is valid, this is sent to the Orderer (i.e. the ordering service)

7. The ordering service packages the transaction into a block

8. The block of transactions is sent back to the committing peers on the channel that independently verify that transaction and endorsement policy one last time

9. The transaction is committed

5.4.1.2 Endorsement Policies

Endorsement policies are the orchestration mechanism for verifying and validating transactions within the channel and have implications to the overall governance of the channel [3]. They should be carefully created and agreed upon when deciding/ creating the channel.

Endorsement policies are the smallest subset of peers within a channel that are required to endorse a given transaction (recall that a transaction needs to be endorsed before being sent to the Orderer).

Policies could be:

1. Explicit- Peers A,B, C and E can only endorse a transaction

2. Categorical- A 70 % majority of peers in channel need to endorse the transaction

By default, all RegNet nodes are given the responsibility of endorsing transactions. This can be changed however depending on the need. Endorsement policies in RegNet are tightly coupled to the Channel Level Agreement that is pre-agreed before joining/ creating the channel.

5.4.1.3 Peers/ Nodes

There are a number of responsibilities that all nodes, wanting to join a RegNet channel, need to undertake:

1. Endorsement - by design (as a default; it can be changed if need be) all RegNet nodes will have to take part in endorsement, this means that they will

have to validate transactions in the channel. The action of endorsing involves independently running the smart contract code associated with a transaction and signing the outcome. This outcome along with the transaction and the endorsing signature by the peer is sent to the ordering service to be included into a block. If all the other peers have come to the same outcome, then and only then is this included within the block. By default, all RegNet nodes are given the responsibility of endorsing all channel transactions. This can be changed however depending on the need.

2. Commit - once the Orderer has ordered the block of endorsed transactions, committing peers make one last check to validate that the transactions will give the same outputs. They then "commit" this block to the distributed ledger- they are responsible for actually writing onto the ledger. Recall that the channel has a replicated distributed ledger, therefore each committing peer actually writes the block to its version of the ledger (which is consistent with all the other peers' version). By default, all RegNet nodes are given the responsibility of committing transactions. This can be changed however depending on the need. The responsibility of this involves maintaining a replicated copy of the distributed ledger.

3. Ordering - their job is to make the block of transactions once the endorsing peers have provided a list of endorsed transactions. Every channel needs at least one. This does not have to be someone specific to the channel's use case, it can also be a third party as their job is to simply cryptographically collate transactions and package them onto the block (for brevity the exact cryptographic mechanism of transaction ordering has not been discussed). Given that the Orderer works with cryptographic material only, he/she can learn nothing identifiable from the transaction.

5.4.2 Tokenization of the FL Process

Once organisations are within a private channel they can initiate a training process through FL. This is done through the use of tokenised artefacts (refer to appendix

D). In order to engage in one FL workflow, organisations must:

1. Add Datasets: each of the organisations add the datasets they want to provide for training. They "add" the dataset by allowing their RegNet peer to access it (this could be as simple as providing a URL to the AWS S3 bucket in which it resides)

2. Register Dataset: Once added to their peer nodes, organisations register this dataset onto the channel; they publish **details** (such as the name, description etc.) across the channel, including the privacy budget they want to provide for the access of the dataset.

3. Register an Algorithmic Purpose: The "Purpose" defines the intent and the use of an Algorithm and respective datasets for a training

4. Register an Algorithm: The peer/ organisation in the channel that wants to provide the Algorithm must register it in the channel and all the elements associated with its Algorithmic Purpose

5. Register an FL Task: Organisations/ peers describe all the details needed to execute a Federated Learning task implemented in an available Algorithm, including the names of the registered datasets (within the channel only) that the task might need. Owners of the selected datasets will be all invited to participate in the task, and its execution will be pending until they approve it.

6. Approve an FL Task: Organisations/ peers have to then approve the execution of the FL task against their datasets. This means they authorise the deployment of the infrastructure needed to execute the task within their infrastructure.

This entire process is orchestrated by way of the tokenized artefacts. Table 5.1 provides the definition of some of these tokenised artefacts. Each of these artefacts is a collection of smart contracts that have been implemented as a finite state machine within RegNet. It is the allowable interactions, permissions and behaviours that channel level members can have on these tokenised artefacts that provide governance. All interactions with these tokenised artefacts are logged by the DLT layer, providing strong auditability of the entire process.

Table 5.1: RegNet Tokenized Artefacts

Tokenised artefact	**Properties**
Dataset	• Name (ascii [50char, no spaces, just -]) • Hash (SHA-256) – key • Owner (MSP ID) • Nonce • Role (training/testing) • Permissions for use during training/evaluation/prediction • Metadata • Status: registered/deprecated/pending(initialised) • Privacy Budget
Algorithm	• Name • Description • Owner (MSP ID) • Hash of the cloned git repo • Nonce • [Purpose keys] (hash) • Quality metrics • Dataset format (schema) • Status: dev, prod
Training Task	• Algorithm key • Nonce • Keys of input data (datasets, other models) • Author • Permissions to see the training task, download and use the output • Aggregation subtask • Status: in_queue, provisioning, provisioning_failed, training, done, failed
Prediction Task	• Algorithm key • Nonce • Keys of input data (datasets, other models) • Author • Permissions to see the prediction task, download the output • Status: in_queue, provisioning, done, failed

5.5 Testing and Results

5.5.1 Testing

The aim of this project is to provide an infrastructure for trusted data access for singular-algorithmic purposes across organisations in a manner that is compliance with regulation. Once the infrastructure was implemented, deployment and testing was carried out with the industry collaborators (which consisted of a number of institutions from financial services). The peer nodes were deployed within their cloud infrastructure. They formed a private channel and each provided access to one of their datasets, all within the bounds of a sandboxed environment. An entire end to end workflow (as discussed in the previous section) was executed. The following subsection explains the datasets used and the use case.

5.5.1.1 Use Case

The FL training experiment was carried out on Anti-Money-Laundering datasets. The private channel consisted of 3 financial institutions (that provided datasets), one algorithm provider and one observer. A Horizontal-FL (figure 5.2) process was executed with the algorithm provider also managing the trusted aggregation (FedAvg). Although the datasets used were mock datasets (already labelled), they were suitable in validating the entire concept and framework. The FL task was to create a classifier for detecting fraudulent monetary transactions. The FL objective was to be train a model across all the financial services institutions' datasets within a federated space and then assess how effective the federated model was compared to each of the institutions own model (i.e. trained with their own data only).

5.5.2 Results

Two training cases were considered; FL with no Differential Privacy and FL with Differential Privacy. Each of the firms used a maximum allowable budget value of $\varepsilon = 60$. The training algorithm was designed to spend an $\varepsilon = 0.8$ [9]. An entirely

[9] These values were empirically found to provide the best accuracy-privacy trade-off when Differential Privacy was applied to each of the datasets. It is left to the owner/administrator of the dataset to determine what levels they want to set this to- they need to provide this value when registering their datasets on the channel.

new dataset was used for validation, and was found to have an average accuracy of 73.1% in a non-federated, non differentially private environment. The accuracy rose to 76.6% for the federated model. This increase was an expected result as the federated model is trained on more data compared to its non-federated counterpart.

In a non-federated, differentially private environment, the average accuracy of the model fell to 65.8%. This was expected as the model trades off accuracy for privacy, a higher ε value would've improved the score (at the expense of privacy). In a federated and differentially private space, the accuracy score was an improvement to 69.3%. This increase was an expected result as the federated model is also trained on more data. In both cases (with and without DP) the federated models each provided a 4% increase in accuracy.

It should be noted that the purpose of the testing was to validate an entire end-to-end FL workflow with the infrastructure rather than the use of FL for AML. Through their involvement, the collaborators validated the use and proposition of the infrastructure and are in the process of using the infrastructure for more complex use cases. A number of other tests are currently being carried out with more collaborators from the insurance, law and government sectors. A Data Protection Impact Assessment is being carried out with the regulator to validate that our approach is compliant with the regulator.

5.6 Conclusions

Through the use of Federated Learning and Tokenization, RegNet, removes the need to explicitly share data, for the purposes of analytics and provides access in an auditable, privacy preserving manner that is able comply with data policies such as GDPR. Tokenization also offers the ability to provide a means to administer data governance, particularly at the channel level. The work undertaken within this chapter proves the use of both technologies for trusted data access (also validated by its application on an applied use case within industry). Additionally, we also provide a taxonomy for Federated Learning- a new and growing form of Machine Learning. Our work also provides a list of open problems with Federated Learning (presented in chapter 6)

The need for such a trusted infrastructure that is compliant with data regulation has been further strengthened by a report released recently by the UK AI Council [77]. It recognises that AI advances with diverse data and stresses the need for trustworthy accessible data. It also recognises that need for making public sector data safely and securely available. To achieve this goal and further validate our approach, a use case is being explore with a UK Government body.

Whilst we proposed the core concepts of RegNet, there are still areas of extension. It is our hope that future work will take the ideas described in the project and apply them to a more complete set of industry-related use cases to further validate the need and approach of the RegNet. To that end work is still being done with a number of enterprise collaborators within the Legal, Insurance, Accountancy, Financial Services and Government sectors. A grant application for further funding on this work has been made to UK Research and Innovation (UKRI).

Chapter 6

Conclusions and Future Work

This thesis investigates the application of blockchain technology and artificial intelligence to the domain of Algorithmic Regulation. It explores the problems with regulatory compliance and supervision in financial services as well as an emerging form of regulation- the regulation of data. The main motivation behind the thesis is to architect Algorithmic Regulation solutions, that automate regulation and compliance, which could realistically be adopted by industry. All the work has been validated by industry partners and collaborators. To this end, the thesis explores 3 projects:

1. Using AI to Automate the Regulatory Handbook
2. Using Blockchain fro Regulatory reporting of data (SmartReg)
3. Using Federated Learning and Blockchain for Privacy Preserving Data Access (RegNet)

The first project looks into the problem of machine interpretability of regulation and provides a "white-box" technique to address machine reasoning with regulation through the use of a knowledge base system. The project provides a means to capture machine readable and executable semantics of regulatory rules that could be used to automate financial services regulation. It demonstrates how the techniques proposed in the project can be applied to the FCA Handbook. The use of a "white-box" solution (compared to a "black-box" Machine Learning solution) has been validated by the regulator's approval in using it internally.

The second project presents a solution to enable regulators and firms (within financial services) to share regulatory reporting data in a manner that is agile, almost real-time and promotes data standardisation across the industry, through the use of distributed ledger technology and smart contracts. Automating regulatory reporting is a growing problem [1]for the industry as regulation gets more complex and systems and processes within the industry [2] are increasingly being automated.

The third project explores the use of Federated Learning and DLT to provide trusted data access for singular algorithmic purposes (e.g. for an Anti-Money-Laundering use case). It provides a means to automate the orchestration of Federated learning through a trusted data infrastructure that relies on tokenization. Although more work is being carried out, it provides an approach that is validated through a use case with industrial collaborators.

6.1 Contributions & Future Work

6.1.1 Using AI to Automate the Regulatory Handbook

The work in capturing regulatory semantics has made the following contributions:

1. A semi-formal model capable of reasoning with financial regulation has been developed. The model has been used to represent regulation from the FCA Regulatory Handbook, proving that it is capable of representing and reasoning with financial regulation.

2. The model has been developed into a working reasoning engine

3. The reasoning engine has been incorporated into a platform that is capable integrating existing regulatory systems or be used as a standalone tool for regulatory advice

Although the project has been able to develop as system capable of reasoning with regulation. Its reasoning engine requires thorough testing with regulation in the

[1]The Bank of England recognise that is costing the UK Economy 4 Billion annually and growing
[2]Other than regulation

real-world. Future work could involve the revision and improvement of the parsing engine and its implementation based upon feedback and testing from this.

A large number of "business" oriented features still need to be implemented onto the platform. For instance, the platform requires a user authentication mechanism and a better user experience. Possible future work can also revolve around these.

This work presents the design for a system capable of reasoning and representing regulation as machine readable and executable semantics - it is the first component in Treleaven et al.'s Algorithmic Regulation system. It can be further used to encode Smart Contracts for Digital Regulatory Reporting.

The work in this project is part of a long-term goal and collaboration with the FCA to deliver a system capable of reasoning with regulation. The work provides the groundwork needed to realise this goal through the development of a system capable of reasoning with regulation. The proof-of-concept, based on this work, has also been used by the FCA for their internal regulatory guidance for providing licenses. The work presented in this chapter has also fed into a Master's dissertation on reasoning with regulation.

6.1.2 SmartReg

SmartReg has explored the design of a new approach to financial regulation. It combines traditional banking infrastructure with a shared distributed ledger to offer a hybrid system that combines the benefits of both technologies. By leveraging Quorum, a version of Ethereum that supports private state, data can move within private channels on a shared network, offering privacy and permissioning between competing entities on the same network.

Whilst we proposed the core concepts of SmartReg in this thesis, there are still areas for extension. It is our hope that future work, will take the ideas described in the project and yield a compelling, production ready solution for modernising regulatory compliance. First, there exists the issue of standardisation. The regulatory functions rely on the assumption that certain standards exist to enable their successful execution across a variety of environments. These assumptions include the existence of a standardised interfacing for running regulatory functions that enables execution support across all supporting nodes. Also, the output of the functions should conform to some standard format, such that the interpretation of the function result can be automated. These standardisation efforts should possibly exist as an open source specification, so that members of the community can propose changes and raise concerns.

Secondly, it will be important to regularly revisit the smart contract source code in light of future Solidity language updates. Many of the limitations explored in the Implementation section are on the roadmap for fixes in future versions.

Finally, communication with major institutions in this sector will be vital. No large scale change, like the one proposed in this paper, can take place in isolation and a successful integration of SmartReg will need the support of large banks and regulators. To this end, the central ideas and concepts behind SmartReg should be made publicly available and an effort made to actively gain feedback from multiple relevant parties. Although not qualitatively assessed the impact of the work presented in this chapter can be seen from the following benefits [3]:

1. Easier compliance- Machine executable rules offer a more efficient less error prone process of interpreting regulatory rules. SmartReg offers a mechanism that is automated and does not rely on intensive human input (which translates to an expensive process currently)

2. More precise and agile regulation- Machine executable rules can be thought of as software; traditional agile software development practices can be integrated for quicker regulatory policy enforcement

[3]Confirmed by the collaborators

3. Accurate and real-time information- SmartReg proposes a solution that can bring down reporting times from weeks to seconds, regulators can have a financial “view” of their world in almost-real time. This can allow them to act more proactively.

4. Standardisation- A single peer to peer reporting infrastructure can potentially integrate multiple disparate systems and promote standardisation through the industry

5. Improved Systemic Risk Control- Almost real time reporting data can allow regulators to address systemic risk more quickly

The design decisions made in SmartReg have informed a wider project carried out by the regulator and a number of banks. This is the Digital Regulatory Reporting Project (DRR). DRR has become a priority for the regulator, banks and the UK financial services industry [12, 13]. This has been the greatest impact of the work presented in this chapter.

6.1.3 RegNet

RegNet provides a framework and infrastructure for trusted data access for singular algorithmic purposes in a manner that is compliant with data regulation. Through the use of Federated Learning and Tokenization, it removes the need to explicitly share data, for the purposes of analytics, and provides access in an auditable, privacy preserving manner that is able comply with data policies such as GDPR. Tokenization also offers the ability to provide a means to administer data governance, particularly at the channel level. The work undertaken within this chapter proves the use of both technologies for trusted data access (also validated by its application on an applied use case within industry).

The need for such a trusted infrastructure that is compliant with data regulation has been further strengthened by a report released recently by the UK AI Council. It recognises that AI advances with diverse data and stresses the need for trustworthy accessible data. It also recognises that need for making public sector data safely and securely available. To achieve this goal and further validate our approach, a use case is being explore with a UK Government body. Throughout our work we have identified a number of open technical problems in FL and our approach, as listed below.

- Data characteristics - Data is also categorised by the ‘unbalancedness’ (i.e. non- Independent and Identically Distributed) of local data samples: a) co-variate shift - local samples have different statistical distributions; b) prior probability shift - local nodes may store labels that have different statistical distributions; c) concept shift dividing into: i) local nodes share the same labels but have different features, and ii) local nodes the same features but different labels; and d) unbalancedness - the data available at the local nodes may vary significantly in size.

- Model characteristics - Given the way in which FL works- it often is a challenge to inspect the training data or a subset of it, it leads to the problem of choosing the hyperparameters and configuring optimisers (for DNNs especially). For instance, these hyperparameters could include parameters such as the number of layers for a DNN, the number of nodes, the structures of the RNN/CNN etc. Optimisers could include identifying batch sizes. More research work needs to be done to provide mechanisms that mitigate/ address this challenge; we propose the use of synthetic datasets to partly address this problem

- Communications- Some FL techniques such as Vertical Federated Learning require constant communication across each of its training peers as, each peer will be training their part of the model. Given that the training process needs to be mediated by a provisioning algorithm that specifies the computation order across the peers; peers have dependent communications and need to speak with each other to exchange intermediate results throughout training. This communications overhead needs to be managed in a manner that ensures that optimum data transfer speed. Slow data transfer across peers can result in inefficient utilisation of computational resources (participants may not be able to start their training if they haven't received all the intermediate training results).

- Performance - PPML requires the use of privacy preserving cryptographic techniques. These add an additional computation complexity to an already computationally intensive process- machine learning. For the same amount of compute resources, an FL process takes a lot more time than its traditional machine learning counterpart, this can impact training times and can tie up resources. This overhead will always be present, given the additional complexity introduced, however optimisations may help reduce this overhead.

- Disparate systems - Given that FL happens across multiple peers, each that may have different systems, hosting/ cloud environments and other disparate infrastructural components, there is a major challenge to create a FL universal infrastructure that enables these systems to easily speak with each other. Most enterprises tend to employ their own infrastructure (including their own virtual private cloud environments and firewalls), in order to protect their sensitive data, that makes the deployment of federated learning infrastructure difficult. Developing infrastructure agnostic frameworks and tooling may help address this challenge. RegNet also facilitates the use of any ML orchestration framework within a channel and is left to the channel members to decide this.

- Availability - FL processes that require the exchange of intermediate results and constant communication of participating peers(e.g. during Vertical FL, as explained above). All the peers need to be reliable, if one goes down (e.g. network failure or crashes), it may interrupt the entire training process that may render all the work done by the other peers as void. Fault-Tolerance and high availability needs to be architected into the infrastructure to address this. Further research work is being done to incentivise Federated Learning through tokenization and reputation.

Whilst we proposed the core concepts of RegNet, there are still areas of extension. It is our hope that future work will take the ideas described in the project and apply them to a more complete set of industry-related use cases to further validate the need and approach of the RegNet. To that end work is still being done with a number of enterprise collaborators within the Legal, Insurance, Accountancy, Financial Services and Government sectors. A grant application for further funding on this work has been made to UK Research and Innovation (UKRI).

Appendix A

Z Notation

This appendix presents the syntax of Z notation used in chapters 1 and 2. It also provides the Z operations used for defining formulae in those chapters.

Z notation [75] is a formal specification language developed in the 70s and 80s. It is used to describe and model computing systems and provides a means to denote a clear, concise specification for computer programs and other information systems.

Syntactic Conventions

In Z notation, there exists a special type constructor referred to as schema. The schema presents the definition for the binding of identifiers or variables to their values.

The following notational conventions are used:

1. Variables, values and other conventions

   (a) User defined types are noted in UPPERCASE
   (b) Z Operators are noted in **bold**
   (c) Comments are written in *italic*
   (d) The declaration of variables that bind identifiers with values are written above the dividing line
   (e) The properties that must hold between each of the values and identifiers must be written below the dividing line
   (f) Identifiers below the line belong to the same declarative scope declared above the line
   (g) For a set defined as S, then #S represents the cardinality of the set S

(h) The concatenation of two sequences S and T can be represented as S$\hat{\text{T}}$

(i) If we let S represent an ordered triple then **S.1** is the first element of the triple, **S.2** is the second element of the triple and **S.3** is the third element of the triple[1]

2. Z Operations

   (a) $f : X \leftrightarrow Y$ represents the notion that f is the relation between types X and Y

   (b) $f : X \nrightarrow Y$ represents the notion that f is a partial function from set X to the set Y

   (c) **dom** f is the domain of f if f is a relation or function

   (d) **ran** f is the domain of f if f is a relation or function

[1]The intuition applies to other ordered sizes of S (other than just triples)

Appendix B

Representing FCA Handbook regulation

This appendix presents a flows of regulation from the Perimeter Guidance section of the FCA Handbook and an accompanying table that presents the encoding of these rules into the Formally Specified Reasoning Engine.

The example below has been taken from the "Perimeter Guidance Manual Section" of the regulatory handbook [9]. It deals with authorisations for firms. We need to identify the above parameters for each node in the diagram. The rulename is a unique id that we can arbitrarily allocate although it is best if we let a hashing algorithm handle this to avoid collisions as this has to be unique [1]. We need our facts to be those that are asserted in the working memory such that the rule will fire. Thus we can either have "y" or "n" to represent this. Our start states are simply those the nodes that point to our node. Actions are the new rules that need to be fired given the response (i.e. the transitions). Our changes are the new rules that are fired as well as any other data such as recommendations made during inference. Based on this we can easily "translate" figure B.1 to the table B.1.
We can therefore see that it is possible to "translate" rule-based regulation into formally specified rules. However, there are times that the recommendations may not be presentable due to the their arbitrary nature.

[1]An annotated version of the diagram is represented in figure B.2, it shows which node corresponds to which unique rulename

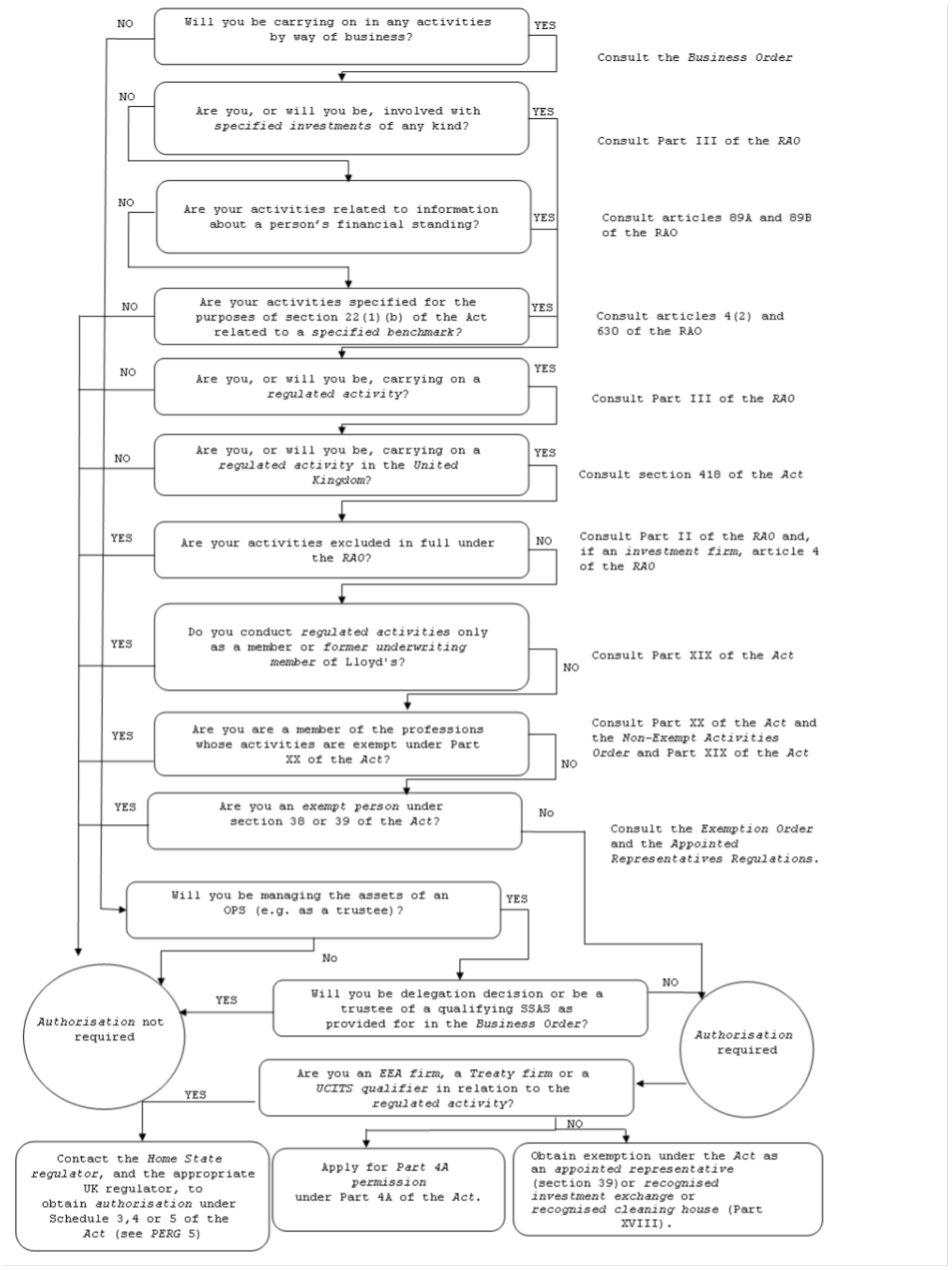
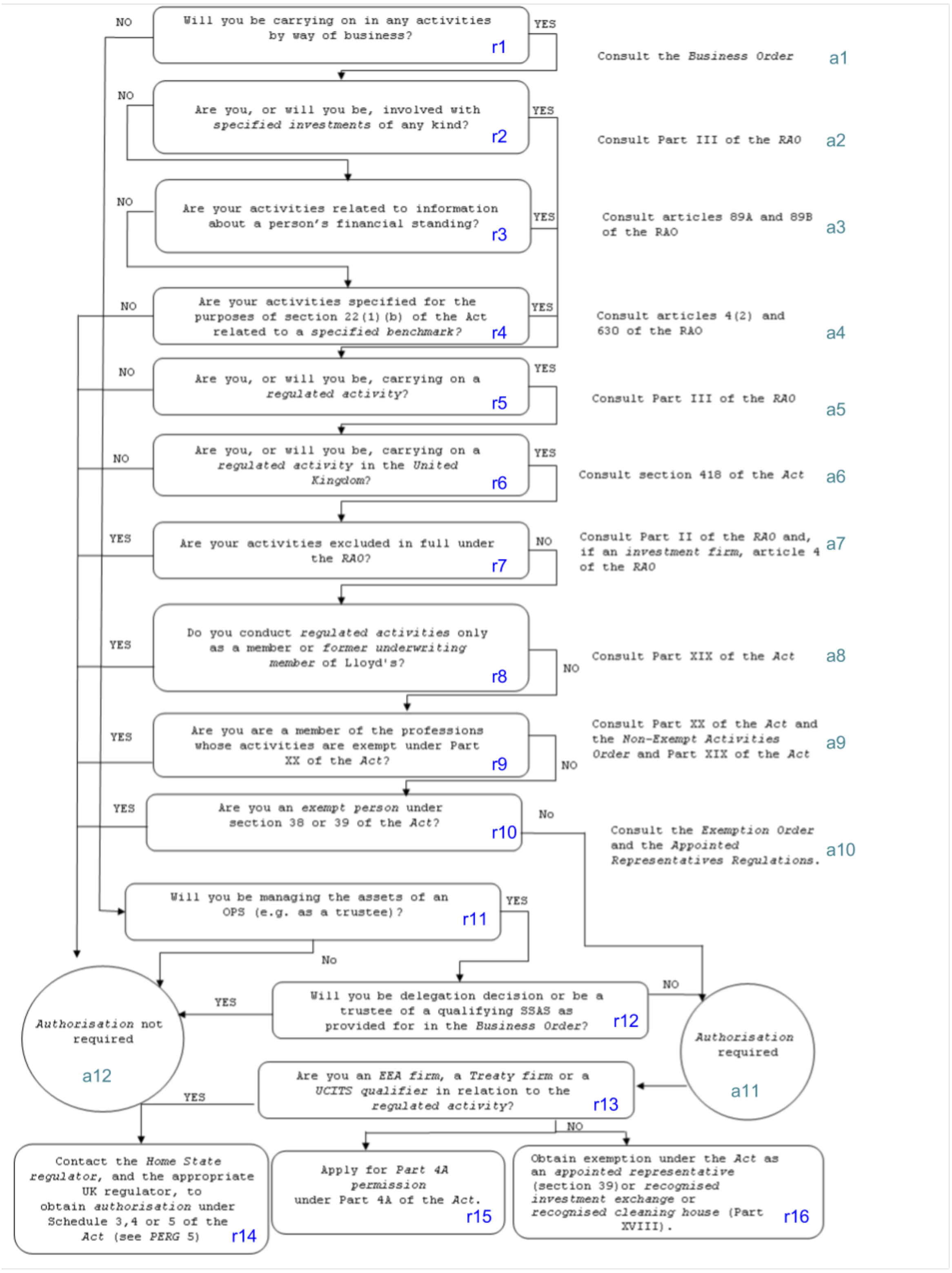

Will you be carrying on in any activities by way of business?
NO
YES
Consult the *Business Order*
Are you, or will you be, involved with *specified investments* of any kind?
NO
YES
Consult Part III of the *RAO*
Are your activities related to information about a person's financial standing?
NO
YES
Consult articles 89A and 89B of the RAO
Are your activities specified for the purposes of section 22(1)(b) of the Act related to a *specified benchmark*?
NO
YES
Consult articles 4(2) and 63O of the RAO
Are you, or will you be, carrying on a *regulated activity*?
NO
YES
Consult Part III of the *RAO*
Are you, or will you be, carrying on a *regulated activity* in the *United Kingdom*?
NO
YES
Consult section 418 of the *Act*
Are your activities excluded in full under the *RAO*?
YES
NO
Consult Part II of the *RAO* and, if an *investment firm*, article 4 of the *RAO*
Do you conduct *regulated activities* only as a member or *former underwriting member* of Lloyd's?
YES
NO
Consult Part XIX of the *Act*
Are you are a member of the professions whose activities are exempt under Part XX of the *Act*?
YES
NO
Consult Part XX of the *Act* and the *Non-Exempt Activities Order* and Part XIX of the *Act*
Are you an *exempt person* under section 38 or 39 of the *Act*?
YES
No
Consult the *Exemption Order* and the *Appointed Representatives Regulations*.
Will you be managing the assets of an OPS (e.g. as a trustee)?
YES
No
Will you be delegation decision or be a trustee of a qualifying SSAS as provided for in the *Business Order*?
YES
NO
Authorisation not required
Authorisation required
Are you an *EEA firm*, a *Treaty firm* or a *UCITS qualifier* in relation to the *regulated activity*?
YES
NO
Contact the *Home State regulator*, and the appropriate UK regulator, to obtain *authorisation* under Schedule 3,4 or 5 of the *Act* (see *PERG* 5)
Apply for *Part 4A permission* under Part 4A of the *Act*.
Obtain exemption under the *Act* as an *appointed representative* (section 39) or *recognised investment exchange* or *recognised cleaning house* (Part XVIII).

Figure B.1: FCA Handbook Authorisations Regulation, from [9]



Will you be carrying on in any activities by way of business? r1
YES
NO
Consult the Business Order a1
Are you, or will you be, involved with specified investments of any kind? r2
YES
NO
Consult Part III of the RAO a2
Are your activities related to information about a person's financial standing? r3
YES
NO
Consult articles 89A and 89B of the RAO a3
Are your activities specified for the purposes of section 22(1)(b) of the Act related to a specified benchmark? r4
YES
NO
Consult articles 4(2) and 630 of the RAO a4
Are you, or will you be, carrying on a regulated activity? r5
YES
NO
Consult Part III of the RAO a5
Are you, or will you be, carrying on a regulated activity in the United Kingdom? r6
YES
NO
Consult section 418 of the Act a6
Are your activities excluded in full under the RAO? r7
NO
YES
Consult Part II of the RAO and, if an investment firm, article 4 of the RAO a7
Do you conduct regulated activities only as a member or former underwriting member of Lloyd's? r8
NO
YES
Consult Part XIX of the Act a8
Are you are a member of the professions whose activities are exempt under Part XX of the Act? r9
NO
YES
Consult Part XX of the Act and the Non-Exempt Activities Order and Part XIX of the Act a9
Are you an exempt person under section 38 or 39 of the Act? r10
No
YES
Consult the Exemption Order and the Appointed Representatives Regulations. a10
Will you be managing the assets of an OPS (e.g. as a trustee)? r11
YES
No
Authorisation not required a12
YES
Will you be delegation decision or be a trustee of a qualifying SSAS as provided for in the Business Order? r12
NO
Authorisation required a11
Are you an EEA firm, a Treaty firm or a UCITS qualifier in relation to the regulated activity? r13
YES
NO
Contact the Home State regulator, and the appropriate UK regulator, to obtain authorisation under Schedule 3,4 or 5 of the Act (see PERG 5) r14
Apply for Part 4A permission under Part 4A of the Act. r15
Obtain exemption under the Act as an appointed representative (section 39) or recognised investment exchange or recognised cleaning house (Part XVIII). r16

Figure B.2: Annotated FCA Handbook Authorisations Regulation, from [9]

Table B.1: A model representation of FCA Handbook Authorisations Regulation

rulename	**r1**	**r2**	**r3**	**r4**	**r5**	**r6**	**r7**	**r8**	**r9**	**r10**	**r11**	**r12**	**r13**	**r14**	**r15**	**r16**
facts	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y
start	start	r1	r2	r3	r4	r5	r6	r7	r8	r9	r1	r11	r12	r4 ,r5 ,r6 ,r7 ,r8 ,r9 ,r10 ,r11, r12, r13	r13	r13
actions	r2	r5	r5	r5	r6	r14	r14	r14	r14	r14	r12	r14	r14	terminate	terminate	terminate
changes	r2 x a2	r5 x a2	r5 x a3	r5 x a4	r6 x a5	r14 x a12	r14 x a12	r14 x a12	r14 x a12	r14 x a12	r12	r14 x a12	r14 x a12	terminate	terminate	terminate
facts	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
start	start	r1	r2	r3	r4	r5	r6	r7	r8	r9	r1	r11	r12	r4 ,r5 ,r6 ,r7 ,r8 ,r9 ,r10 ,r11, r12, r13	r13	r13
actions	r11	r3	r4	r11	r11	r11	r8	r9	r10	r13	r13	r11	r15 x r16	terminate	terminate	terminate
changes	r11	r3	r4	r11	r11	r11	a7 x r8	a8 x r9	a9 x r10	a10 x a11 x r13	r13	r11 x a13	r15 x r16	terminate	terminate	terminate

Appendix C

License Registration Decision Tree

This appendix presents a flow for the registration of licenses as per the FCA Handbook. It is the use case used for the testing of the system.

The problem considered in chapter 3.3.3 was that of license registration where a client asks the question: "I would like to assist clients in buying equities. What licenses do I need?". To answer this a lawyer/ regulatory advisor must ask the following questions.

1. Will you be buying the equities for them?
2. Will you buy the equities in your name, then sell them to your client?
3. Where will the purchased equities be held? With custodian? With bank?
4. Will you put clients in touch with other brokers?
5. Will you assist your clients in opening accounts with custodians and banks?
6. Will you assist clients by taking their purchase orders and sending them to brokers, banks or custodians?
7. Will you put clients in touch with custodians? Or banks who provide custody?
8. Will you advise your clients on which equities to buy?
9. Will you receive money from your clients when they open an account?
10. Will you receive money from your client before they purchase equities? Or will the money go directly to banks and other brokers?

The set of answers are logical and can be represented as a decision tree seen from figure C.1

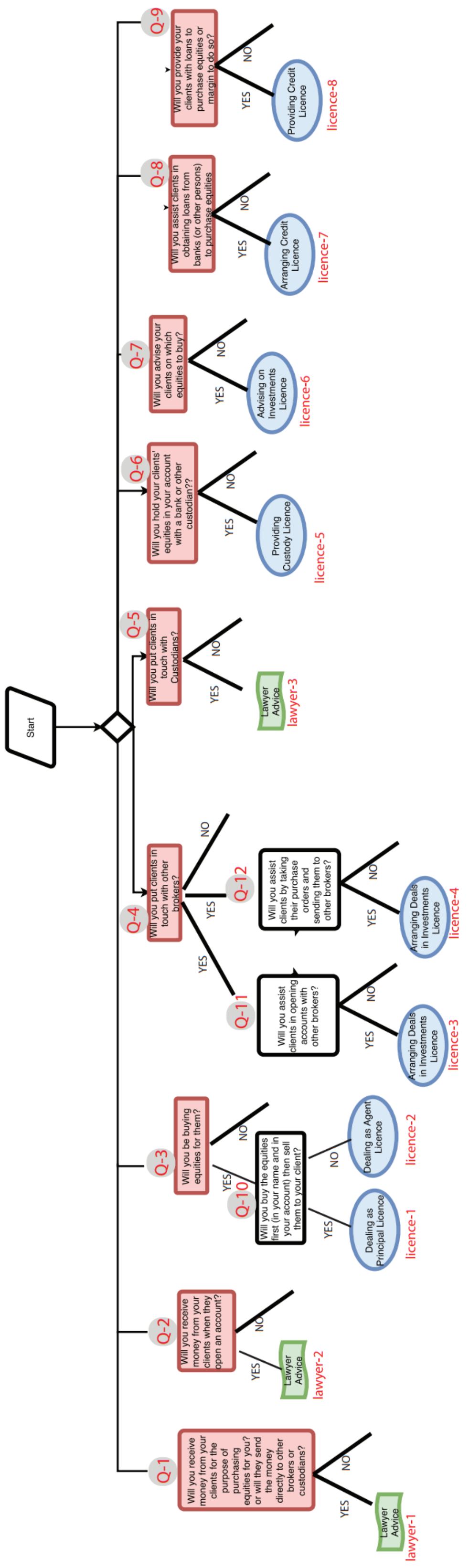

Start
Q-1
Will you receive money from your clients for the purpose of purchasing equities for you? or will they send the money directly to other brokers or custodians?
YES
NO
Lawyer Advice
lawyer-1
Q-2
Will you receive money from your clients when they open an account?
YES
NO
Lawyer Advice
lawyer-2
Q-3
Will you be buying equities for them?
YES
NO
Q-10
Will you buy the equities first (in your name and in your account) then sell them to your client?
YES
NO
Dealing as Principal Licence
licence-1
Dealing as Agent Licence
licence-2
Q-4
Will you put clients in touch with other brokers?
YES
NO
Q-11
Will you assist clients in opening accounts with other brokers?
YES
NO
Arranging Deals in Investments Licence
licence-3
Q-12
Will you assist clients by taking their purchase orders and sending them to other brokers?
YES
NO
Arranging Deals in Investments Licence
licence-4
Q-5
Will you put clients in touch with Custodians?
YES
NO
Lawyer Advice
lawyer-3
Q-6
Will you hold your clients' equities in your account with a bank or other custodian??
YES
NO
Providing Custody Licence
licence-5
Q-7
Will you advise your clients on which equities to buy?
YES
NO
Advising on Investments Licence
licence-6
Q-8
Will you assist clients in obtaining loans from banks (or other persons) to purchase equities
YES
NO
Arranging Credit Licence
licence-7
Q-9
Will you provide your clients with loans to purchase equities or margin to do so?
YES
NO
Providing Credit Licence
licence-8

Figure C.1: License Registration Decision Tree

Appendix D

Blockchain Tokenization

This appendix presents a a paper on Blockchain Tokenization. Tokenization has been used in detail in chapters 4 and 5 for the Tokenization of Regulatory Reporting and Federated Learning. The intention of this chapter is to introduce Tokenization and how it works.

D.1 Abstract

Tokenization in the context of blockchain technology has the potential to revolutionise business processes and trade, but will require legal status and regulation. In this paper we discuss the vision, impact and challenges of tokenization. We investigate the process of tokenization and provide a revised token taxonomy that encapsulates the governance, technical, functional and behavioural parameters of tokenization. We further introduce a state machine based abstraction to conceptualise tokenization and discuss the technical aspects common token platforms such as Ethereum.

D.2 Introduction

Our world is full of assets: stocks; real estate; gold; carbon credits; oil etc. Many of these assets are difficult to physically transfer or subdivide. Representation of these through abstraction allows us to overcome the problem of infungibility[1]. However, physical representations, even paper, legal agreements and other tangible media are cumbersome, relatively difficult to transfer, difficult to track and often difficult to

[1]**Fungibility** refers to the ability of goods or a commodity to be freely interchangeable with another in satisfying an obligation. Gold is a fungible asset as the value of 1 kg of gold coins is the same as 1 kg of gold ingots.

authenticate. Representing assets as digital tokens solves part of the problem: it makes them more readily ‘fungible’ and they are easier to track and manage. The problem of maintaining authenticity however remains. Traditional systems tackle this problem using centralised bodies to issue/manage these tokens e.g. centrally banked currencies or registries that manage paper-based equivalents of these assets (such as land registries and title deeds). Other examples include the issuing agents’ own platform (e.g. Game Tokens issued by the Games Developer for use within the game) or a marketplace like Stock Exchanges that require the listing of a stock.

An obvious question is the relationship between a token and currency, since both act as surrogates. By analogy, currency represents the obligation by the central bank of a country to pay the owner, of the token, a sum of equal value being represented by the token. In the example of currency, a bank note is a token; tokens abstract the value of the object/piece of data that they represent. Conceptually a token is a representation for a more valuable piece of information. This underlying piece of data is what gives them intrinsic value. Thus, for currency, abstracting monetary value in the form of currency tokens creates an instrument that is capable of being held as a medium of exchange, store of wealth and a unit of account which makes it useful.

Table D.1: Comparison of commonly used tokens

Type	**Identity**	**Physical Object**	**Value**
Currency	Represents a universal medium of exchange	Bank notes/ Coins	Market, regulated by a Central Bank
Shares	Represents ownership of company	Paper certificates/ Ledger entries	Company market capitalisation
Tokens	Represents an asset or piece of information	Data structure	Underlying/ piece of data

The digital tokens of these physical counterparts make them more convenient to transfer, use and track. However, central bodies such as Exchanges and issuing bodies are still required to maintain authenticity of the tokens. This need for centralisation and management often means that the given tokens are restricted to specific businesses, organisation or location. These restrictions often lead to inefficiencies, lack of innovation, and financial exclusion [21].

Decentralised computing systems such as Bitcoin and Ethereum offer an alternative mechanism for issuing, verifying and authenticating tokens. The novelty in these systems is that they use network effects to function; they leverage their network of peers to solve the problem of trust using a game theoretic consensus mechanism. This allows tokens to be authenticated and issued by any participant within the network.

Tokenization is a major paradigm, especially in financial and social systems, as it provides a passage through which they can evolve into decentralised or hybrid equivalents and support fractional ownership; referred to as 'divisibility'. Traditionally these systems have relied on centralised governance to manage the problem of trustworthiness, but distributed consensus within peer-to-peer network resolves this.

D.2.1 Tokens and Tokenization

Tokens were one of the first applications using blockchain technology. They pushed the use of the technology beyond its original use as a simple distributed, network incentivised accounting ledger[2]. These tokens initially were implemented within the infrastructure layer (figure D.1) of the blockchain technology stack. They were closely coupled with the network and its subsistence.

At the protocol layer (figure D.1), tokens evolved to outsourced extensions of Bitcoin's core protocol. Instead of being integrated as a feature on a software level, the first tokens were created based on misappropriating data fields in Bitcoin transactions [3].The premise of this was simple; it was possible to encode some metadata into existing Bitcoin transactions, these bits of metadata could be used to refer to something else whilst taking advantage of the strong immutability that the Bitcoin blockchain and its network provided. Often referred to as coloured coins, they

[2]As such Bitcoin could be thought of as the first token that made use of this accounting ledger
[3]Such as encoding data in the amount or `op_return` field

functioned as units of account keeping track of who owns how much of the data value they were used to represent.

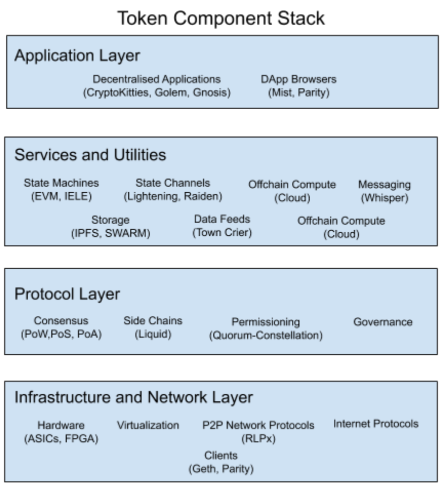


Figure D.1: A Blockchain Technology Stack, tokens evolved from being implemented in the infrastructure layer to the protocol, services and application layers

That strategy of extending Bitcoin's feature set came at the cost of having to develop an additional set of stand-alone client software that implemented the protocol needed to decode the meta-information hidden in Bitcoin transactions.

From there, tokens have evolved significantly, from blockchains being understood as dumb accounting ledgers towards general-purpose state machines with arbitrarily complex (Turing-complete) state transition rules (Application and Service Layers, figureD.1) that operate on an immutable ledger. Smart contracts, which are another name for a collection of transition rules of distributed state machines, enable the creation of tokens with complex behaviours attached, easily from within the core protocol. Instead of having to develop standalone software to allow people to make use of tokens, they can now be created and handled from within all clients supporting the core protocol of such a new type of blockchain. Most tokens are created at this layer.

There are four main aspects to understanding tokenization:

- **Tokens** - Tokens are the digital identity of something that can be owned by someone. Based on that premise, tokens can have a representational dimension, stretching into the physical, the virtual or the legal realm respectively.

- **Coins** - although tokens and crypto coins are often used interchangeably, coins are cryptocurrencies that function as a currency or medium of exchange. Coins are tokens that act as currencies.

- **Tokenization** - the process of designing and issuing tokens. For instance, the process of converting rights to real-world assets into a digital token. This is a form of 'securitization' through tokens; turning a set of assets into a (financial) security tokens.

- **Tokenomics** - an evolving field of study that explores token design with respect to incentive mechanisms, business models, network effects and other socio-economic factors that enable the token to function

Two closely related issues to these are:

- **ICOs** - Initial coin offerings are a means of crowdfunding centred on cryptocurrencies, which can be a source of capital for start-up companies. ICO tokens supposedly become functional units of currency if or when the ICO's funding goal is met and the project launches.

- **STOs** - Security token offerings, they are a means of issuing tokenized securities. These are traditional securities contracts [4] implemented as tokens via smart contracts.

[4] A Securities Contract by definition is one that satisfies the Howey Test [70]

For instance, tokenization may be viewed as a natural evolution from physical assets, through paper-based derivatives, to digital tokens secured on a blockchain (see Table D.1) Although commodity exchanges have replaced ‘paper’ with electronic transactions, and standardised legal agreements, financial institutions are now trialling tokenized (cf. computer protocols intended to digitally facilitate transactions) to reduce the existing enormous overheads [21].

As discussed, tokenization of physical or digital assets potentially provides: better security, anonymity, global tradability, transferability, liquidity, instance settlement etc. [7]. An example is the authentication of diamonds. Diamonds are notoriously difficult to inspect, authenticate, secure, sell, own and transport. Using tokenization of a stock of diamonds, a customer could invest in a portion of the stock with the grades and cuts secured in a blockchain without the trouble of having to physically receive, store and protect the diamonds[24].

Table D.2: Evolution of “Tokenization”

	Commodities	**Currencies**	**Derivatives**	**Any Assets (if tokenized)**
Asset	commodities, coal	gold, salt etc.	financial	physical & digital
Token	none	bank note number	derivative	blockchain token
Transfer-ability	physical transfer	physical & electronic	paper & electronic	electronic
Divisibility	varies	usually 100 sub-units	not divisible	varies
Traceability	difficult	difficult	electronic trail	electronic trail (UTXO or similar)
Security	weak	weak	medium	potentially stronger
Anonymity	anonymous	anonymous	known	pseudo-nymous/ known
Legal	cumbersome	varies, depends on legal framework used	varies, depends on legal framework used	varies, potentially more difficult to wrap in a legal framework
Trusted participants	no	no	yes	no

D.2.2 To Tokenize or Not Tokenize

A common misconception is that tokens are a means of fundraising. Although this is an application of tokens, it is not the main purpose of tokens. There exist more traditional ways to raise funds for a venture [5]. Tokens should be thought of as a product feature that integrates deeply with business models and networks rather than a funding mechanism.

Although opportunism is driving the momentum for the development and price

[5]e.g. traditional venture capitals, bank loans and angel investors

of tokenized assets, it is important to consider that not all use cases can sustainably benefit from tokenization. It is first important to consider the properties of tokens with the intended use case in mind. To fully make use of tokenization the following need to be considered[6]:

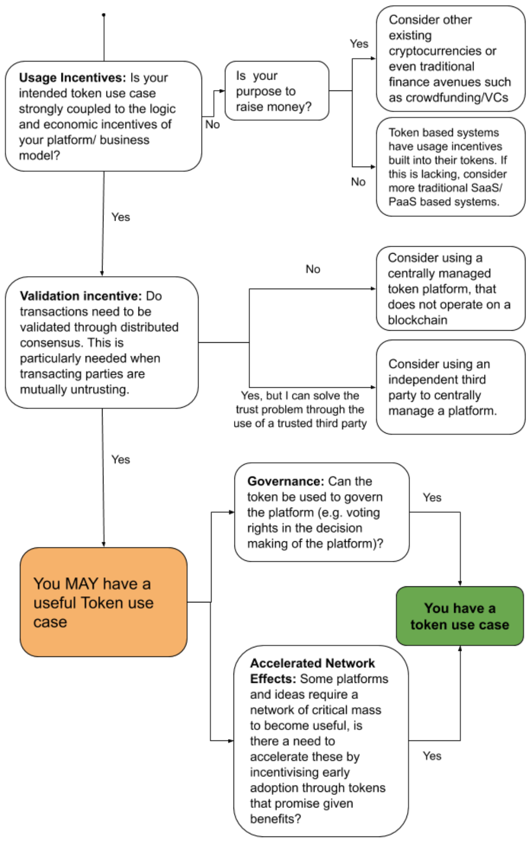

Usage Incentives: Is your intended token use case strongly coupled to the logic and economic incentives of your platform/ business model?
No
Is your purpose to raise money?
Yes
Consider other existing cryptocurrencies or even traditional finance avenues such as crowdfunding/VCs
No
Token based systems have usage incentives built into their tokens. If this is lacking, consider more traditional SaaS/ PaaS based systems.
Yes
Validation incentive: Do transactions need to be validated through distributed consensus. This is particularly needed when transacting parties are mutually untrusting.
No
Consider using a centrally managed token platform, that does not operate on a blockchain
Yes, but I can solve the trust problem through the use of a trusted third party
Consider using an independent third party to centrally manage a platform.
Yes
You MAY have a useful Token use case
Governance: Can the token be used to govern the platform (e.g. voting rights in the decision making of the platform)?
Yes
You have a token use case
Accelerated Network Effects: Some platforms and ideas require a network of critical mass to become useful, is there a need to accelerate these by incentivising early adoption through tokens that promise given benefits?
Yes

Figure D.2: Do you need a token?

D.2.3 Properties of Tokens

In designing tokens, it is important to consider four important factors:

1. **Purpose:** This refers to the value proposition that token usage provides. It is the reason for the token to exist. Factors that need to be considered include the role and function of the token with respect to the platform's business and incentive model

[6]Partly based on Oliviera et al's work [61]

2. **Incentive Model:** This refers to the incentive alignment between the token user, the platform and the platform's developer and the means through which they interact with each other

3. **Operational:** These are the attributes such as fungibility, expirability, tradeability etc that give the token functional use with respect to the platform

4. **Technical:** This refers to the technical stack on which the intended token is meant to operate. This could be native to a blockchain or bootstrapped onto an existing smart contract enabled blockchain (e.g. Ethereum)

D.3 A Taxonomy of Tokens

In [61] Oliviera et al. provides a classification table for tokens based on existing literature and empirical data. Although this classification is detailed, arguably it is not complete. We provide an augmented classification table with the following additional parameters to their existing token morphological box:

1. **Fractionability:** This parameter belongs to the Functional attributes class. This refers to the number of divisions that a given token can be broken into, in some cases token have been made divisible. This has a major influence in its incentive mechanisms and economics. For instance, with a limited supply and divisibility, the token will have a deflationary characteristic.

2. **Traceability:** This parameter belongs to the Functional attribute class. This refers to whether the token's ownership can be traced across the ledger. Some systems such as ZCash make use of zk-SNARKs based protocols that have anonymous transactions which may influence its use.

Table D.3: Classifying purpose parameters of the token taxonomy, based on [61]

Purpose Parameters		
Class	**Function**	**Role**
Coin/Cryptocurrency	Asset-Based Token	Right
		Value Exchange
Utility Token	Usage Token	Toll
		Reward
Tokenised Security	Work Token	Currency
		Earnings

Table D.4: Classifying governance parameters of the token taxonomy, based on [61]

Governance Parameters				
Representation		Digital	Physical	Legal
Supply	Schedule Based	Pre mined, scheduled distribution	Pre-mined, one-off distribution	Discretionary
Incentive System	Enter Platform	Use Platform	Stay Long Term/ Hodl	Leave platform

There are different ways to differentiate tokens, based on their attributes and parameters. A combination of given parameters gives rise to tokens with a different set of behaviours. This gives rise to a token designed to fulfil a specific role. Roles encapsulate a collection of behaviours that allow the token to function.

Tokens can be grouped in multiple ways, one such way is through the set of behaviours [7]they encapsulate. A given combination of behaviours gives rise to a token intended for a given role. A role could be considered as an encapsulation of behaviours.

For example, consider a token which will be designed to function as a currency this will belong to the class Coin/Cryptocurrency. The following key parameters (tables D.3-D.6) will give it the behaviour of a currency:

1. **Function:** Asset-Based
2. **Role:** Value Exchange, Currency
3. **Spendability:** Spendable

[7]i.e. the parameters in tables D.3-D.6

Table D.5: Classifying technical parameters of the token taxonomy, based on [61]

Technical Parameters	
Layer	**Chain**
Blockchain (Native)	New Chain, New Code
Protocol (Non-native)	New Chain, Forked Code
Application (dApp)	Forked Chain, Forked Code
	Issued on top of a protocol

Table D.6: Classifying functional parameters of the token taxonomy, based on [61]

Functional Parameters		
Spendability	Spendable	Non-Spendable
Tradability	Tradeable	Non-Tradeable
Burnability	Burnable	Non-Burnable
Expirability	Expirable	Non-Expirable
Fungibility	Fungible	Non-Fungible
Fractionability	Fractionable	Non-Fractionable
Traceability	Anonymous	Pseudonymous

4. **Tradeability:** Tradeable

5. **Fungibility:** Fungible

Common roles include:

1. **Currency:** A token can serve as a form of currency, with a value determined through private trade. For example, ZCash or Bitcoin.

2. **Resource:** A token can represent a resource earned or produced in a sharing-economy or resource-sharing environment. For example, a storage or CPU token representing resources that can be shared over a network, FileCoin.

3. **Asset:** A token can represent ownership of an intrinsic or extrinsic, tangible or intangible asset. For example, gold, real-estate, a car, oil, energy etc.

4. **Access:** A token can represent access rights and even convey access to a digital or physical property, such as a discussion forum, an exclusive website, a hotel room, a rental car.

5. **Equity:** A token can represent shareholder equity in a digital organization (e.g. a Decentralised Autonomous Organisation, DAO) or legal fiction (e.g. a

corporation)

6. **Voting:** A token can represent voting rights in a digital or legal system.

7. **Collectible:** A token can represent a digital (e.g. CryptoKitties) or physical collectible (e.g. a painting)

8. **Identity:** A token can represent a digital (e.g. avatar) or legal identity (e.g. national ID).

9. **Attestation:** A token can represent a certification or attestation of fact by some authority or by a decentralized reputation system.

Often a single token encompasses several of these roles. Sometimes it is hard to discern between them, as the physical equivalents have always been inextricably linked. For example, in the physical world, a driver's license (attestation) is also an identity document (identity) and the two cannot be separated. In the digital realm, previously commingled functions can be separated and developed independently (e.g. an anonymous attestation). This makes tokens malleable in function where a single token could for instance operate as a voting right as well as an asset.

D.4 Tokens as State Machines

Conceptually, tokens could be thought of as a value encoded state machines defined by a collection of valid states and their respective transitions functions. States are typically a collection of valid blockchain addresses that have the ability to transform the state machine to different states. Transitions are typically blockchain transactions that carry out the transitions. These transitions have strong game theoretic mechanisms that work together to fulfil that token's intended role as an instrument for transacting value. The careful design of these game theoretic incentive schemes is part of the tokens tokenomics.

For example, consider the tokenization of a bond instrument[8] in the figure D.3

[8]This is basic, and is intended to be an example

below. It has four states:

1. $S_{issuance}$ -this is the start state where the bond issuer issues tokens via a trade

2. S_{hold} -holders of the token can hold their token till redemption or trade with their counterparts or receive payments (coupons)

3. $S_{service}$ -the bond is serviced e.g. coupon payments are made

4. $S_{redemption}$ -this is the end state where the bond token is redeemed after its redemption date

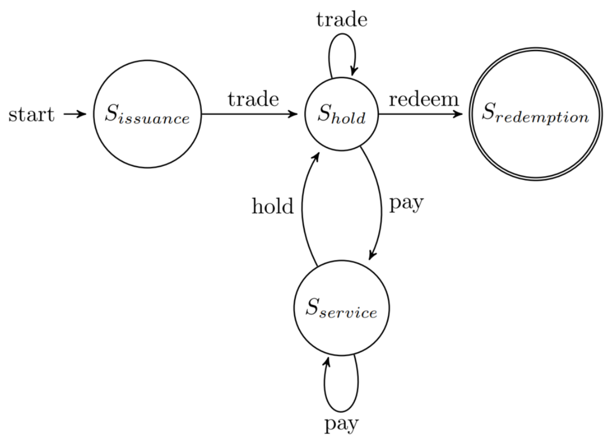

$$\begin{tikzcd}
\text{start} \arrow[r] & S_{issuance} \arrow[r, "\text{trade}"] & S_{hold} \arrow[loop above, "\text{trade}"] \arrow[r, "\text{redeem}"] \arrow[d, bend left, "\text{pay}"] & S_{redemption} \\
 & & S_{service} \arrow[u, bend left, "\text{hold}"] \arrow[loop below, "\text{pay}"] &
\end{tikzcd}$$

Figure D.3: Bond token as a state machine

The states can be transitioned through the transition functions: trade, hold, redeem and pay.

The nodes (states) themselves could be a smaller series of sub-states with auxiliary transitions that work to enrich the role and function of the token.

Alternatively, a node could encapsulate another token system with its own state machine, where the child works on top of the parent and can inherit its parent's tokenomics. This is the case for tokens issued on top of Ethereum.

D.5 Token Platforms

Tokens are implemented through smart contracts[9]. These are computer programs that run on blockchain platforms such as Ethereum [20, 85], Cardano [48] or Libra[15]. Smart Contracts on these platforms are typically implemented using a high-level language (such as Solidity in Ethereum and MoveIR in Libra) which then compile to bytecode that runs on a virtual machine.

These platforms use an account-based model where the global state is a mapping from account addresses to account values. It is these values that contain the data and execution logic for Smart Contracts. In general, to create a token (i.e. to implement the decentralised state machine) we require:

1. A data structure that maps the account addresses to token(s), for ownership
2. The functions to manage the token e.g. `transfer()`

The way different platforms handle this is different. This paper looks into two different systems that manage tokens rather differently. Although no longer publicly active (compared to other blockchains), Libra has been chosen to demonstrate the stark differences between the two and their respective smart contract virtual machines.

D.5.1 Ethereum

Ethereum is the most popular ecosystem for deploying tokens currently, the ERC20 token standard is the standard most widely used in the ecosystem. It is a fungible

[9]A smart contract is essentially a computer program that directly controls the transfer of digital currencies or assets between parties under certain conditions. There is nothing inherently 'smart' about it, nor is it a legally binding contract

"vanilla token" standard where issuers are free to implement their desired token behaviours through Solidity functions.

In Ethereum, each Smart Contract has its own account with its own address. The smart contract holds the data and the code together, similar to how classes are defined in object-oriented languages.

The functions `transfer()` and `transferFrom()` allow the tokens to be traded, it is this that gives it the functional property of fungibility. The `approve()` and `allowance()` functions facilitate the `transfer()` and `transferFrom()` functions.

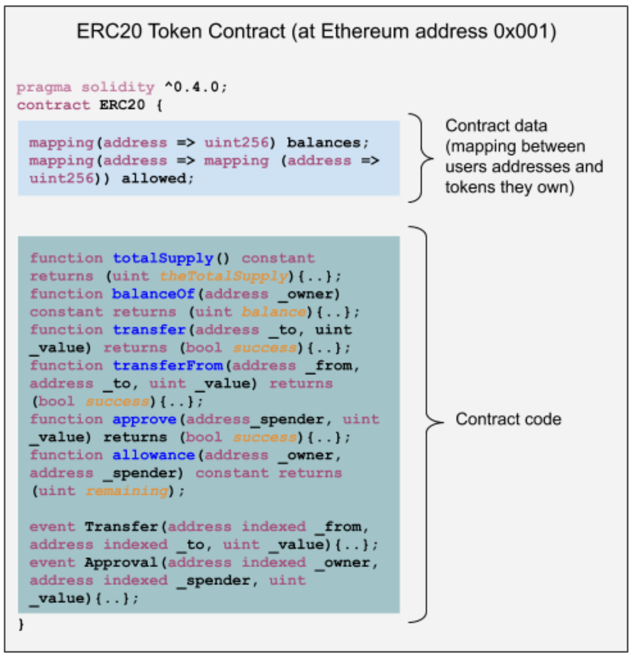
The ERC20 standard also has a number of optional to implement variables that make the token more usable by providing meta-information about the token:

1. **Name:** Returns a human readable name (e.g. "US Dollars") of the token.
2. **Symbol:** Returns a human readable symbol (e.g. "USD") for the token
3. **Decimals:** Returns the number of decimals used to divide token amounts. For example, if decimals is "2", then the token amount is divided by 100 to get its user representation

Other than the ERC20, the ecosystem has a number of token standardization proposals with different use cases. For instance, Security token standards such as the ERC1400 are being used to issue tokenized securities [52] on Ethereum as seen from table D.7.

Table D.7: Common Ethereum Standards

Token Type	**ERC Standard [28]**	**Example**
Fungible Token Standard	ERC20/ ERC 223	OmiseGo
Non-Fungible Token Standard	ERC721	CryptoKitties
Security Token Standard	ERC1400	PolyMath


ERC20 Token Contract (at Ethereum address 0x001)
Contract data (mapping between users addresses and tokens they own)
Contract code

```
pragma solidity ^0.4.0;
contract ERC20 {
  mapping(address => uint256) balances;
  mapping(address => mapping (address =>
  uint256)) allowed;

  function totalSupply() constant
  returns (uint theTotalSupply){..};
  function balanceOf(address _owner)
  constant returns (uint balance){..};
  function transfer(address _to, uint
  _value) returns (bool success){..};
  function transferFrom(address _from,
  address _to, uint _value) returns
  (bool success){..};
  function approve(address_spender, uint
  _value) returns (bool success){..};
  function allowance(address _owner,
  address _spender) constant returns
  (uint remaining);

  event Transfer(address indexed _from,
  address indexed _to, uint _value){..};
  event Approval(address indexed _owner,
  address indexed _spender, uint
  _value){..};
}
```

Figure D.4: The ERC20 token standard smart contract

D.5.2 Libra

Although Libra follows the same computational model of Ethereum where a Virtual Machine runs on a blockchain, it is different to the way it operates. In Libra (i.e. Move), Move takes a more functional approach in that the data is separated from the code [16].

This is unique to Move and other Smart Contract systems. The novelty of the Move language is the way in which the data (a 'resource' type as labelled in Move), is defined. It is a first-class object. This means that in In Move, we would both deploy a new resource type (that can correspond to a token) and code, which would allow operating on the token. These two concepts are separate. To transfer a token, we would again invoke the smart contract. However, it would not update its internal state -as a smart contract (referred to as a 'module' in Move) does not have any state associated with it by design. We rather 'move' the resource from one account to another. The data i.e. the state belongs to the user's account, not to the contract.

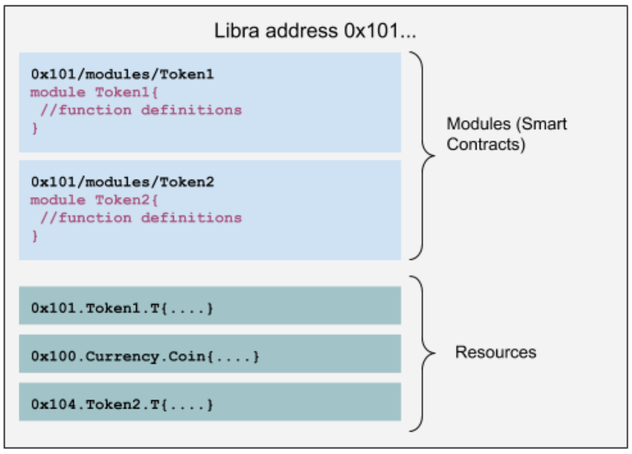


Figure D.5: Move smart contract

Conceptually, a 'resource' is a structure datatype that utilizes an ownership model but can never be copied only moved and borrowed. This is a core feature of the Move language as it guarantees that no defined resource accidentally duplicates, eliminating the possibility for double-spending or re-entrancy attacks (therefore the notion of a 'resource' corresponds well with the concept of tokens).

This is where the potential benefits of Move, in comparison to other Smart Contract languages, become apparent. If we were to deposit a number of tokens, we have to control the memory ownership of the tokens. We can only gain this ownership by splitting an existing owned token (also known as withdrawing) or when minting fresh tokens. This unique ownership property guarantees that the same tokens cannot exist elsewhere, eliminating bugs stemming from incorrect duplications, allowing double-spending and other erroneous behaviour.

Additionally, the resource-ownership model also requires that an owned token has to be either explicitly destroyed or moved to another owner. This guarantees that the token doesn't get accidentally locked inside a module and never to be retrieved again.

By design, Move offers strong asset-guarantees through the resource-ownership model. This can potentially enable developers to produce less error-prone code and move faster.

D.5.3 Token Standards and Exchange Protocols

Token Standards

Common standards are being developed as the tokenization ecosystem matures [80]. This is done through interface smart contracts. The reasons for doing this include:

1. **Improved Token Security:** Smart contracts implementation is tricky; a large number of attacks within the tokenization ecosystem have happened due to poorly designed and implemented smart contracts [10], with multiple attack vectors for malicious agents to exploit. Having a common standard that is audited by the ecosystem reduces this risk.

2. **Accesibility:** With common token standards, it is possible to create interfaces with exchanges and other token-based applications as they all adhere to common standards. Moreover, new start-ups may find it daunting to implement token smart contracts from scratch, working from token standards provides them the ability to bootstrap and grow quickly.

3. **Usability and Tooling:** Common standards promote the development of tools and applications across the entire family of standards. This provides uniformity and improved usability for end-users and token developers alike.

Exchange Protocols

Tokenization has enabled trading of tokenised instruments in completely different manner: traditional markets require centralised exchanges as a marketplace to buy and sell orders, these functions can be automated through smart contracts that

[10]The DAO re-entrancy attack is a critical example [51]

operate independently once implemented- exchanges can be implemented via smart contracts that can operate independently within the blockchain platform without any operational oversight. This is possible because all the tokens follow particular standards that are compatible with the decentralised exchange. For example, consider figure D.6:

1. Maker[11] approves the decentralized exchange (DEX) contract to access their balance of Token A.

2. Maker creates an order to exchange Token A for Token B, specifying a desired exchange rate, expiration time (beyond which the order cannot be filled), and signs the order with their private key. Maker broadcasts the order over the internet.

3. Taker[12] intercepts the order and decides that they would like to fill it

4. Taker approves the DEX contract to access their balance of Token B.

5. Taker submits the makers signed order to the DEX contract.

6. The DEX contract authenticates makers signature, verifies that the order has not expired, verifies that the order has not already been filled, then transfers tokens between the two parties at the specified exchange rate.

With the emergence of independent issuance and ownership and decentralised exchanges and protocols tokens are increasingly being traded in a new kind of market. This has been referred to as ‘Decentralised Finance’ or ‘DeFi’.

[11]This refers to the Market Maker, the entity that creates the liquidity in the exchange

[12]This refers to the individual/institution on the other side of the of the Maker and on the other side of the trade

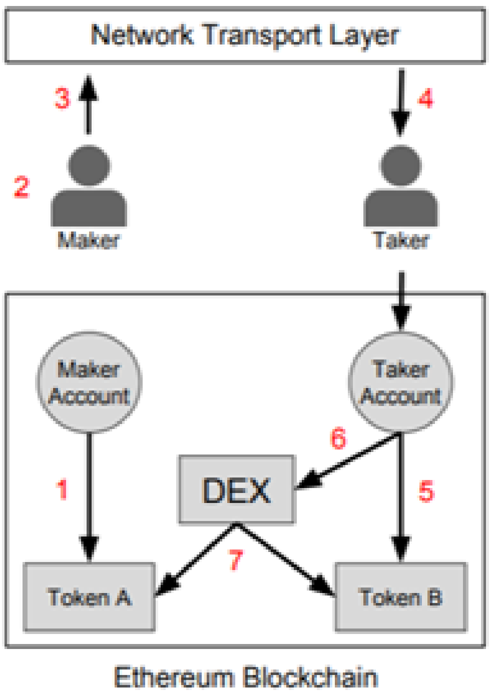


Figure D.6: Decentralised exchanges, adapted from [84]

Tokenizing Securities

Computer programs by nature are composable; smaller programs can combine with other smaller programs to form larger more complex and capable programs. This property of composability is also true for tokens as they are simply smart contracts based programs. By implication, financial securities can be composable through tokenization as seen from figure D.7.

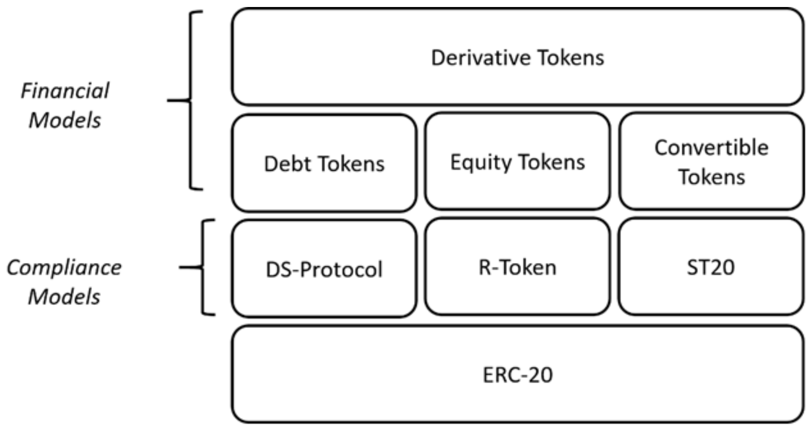


Figure D.7: Composability of Financial Instruments through Tokenization

D.6 Tokenization challenges and advantages

D.6.1 Advantages

Although not exhaustive, tokens and tokenization can bring the following benefits to business processes, markets and society:

1. **Greater automation:** -Smart contracts can be used to program tokens and their behaviours which provide greater automation

2. **Programmability:** - Tokens are state machines whose behaviours can be programmed, enabling them to carry out several roles simultaneously

3. **Improved Regulatory Compliance**- Through smart contracts it could be possible to write ownership and law directly into a token, the token will be able to execute, regulate and govern itself

4. **Greater efficiency and scalability:** - Greater automation can make business, trading and clearing processes quicker and more efficient, allowing them to grow

As discussed in the previous chapter, tokens are state machines capable of transacting value. These value-based transactions, of the state machine, often map to business processes that were once difficult to automate due to the issue of trust e.g. settlement, trading and funding.

Greater Automation

Tokenization can automate a large number of these tasks that require some sort of trusted intermediation to perform them. For example, most digital payments require a trusted intermediary to clear and process the activity, tokenising the payment instrument and running it on a blockchain enables disintermediation [Bitcoin, Ethereum].

Programmability

Through smart contracts, it is possible to implement a set of behaviours on a tokenised instrument that was previously operationally difficult to implement, for instance consider the ability to programme the attribute of fractionability for a token, how fractionable a token is depends on the ability of the token to be "broken down" further i.e. be further divisible. A fractionable token of a tokenised asset such as a building can enable fractional asset ownership. This programmable attribute can enable fractional ownership of large assets. This can potentially translate into greater liquidity and inclusion into the market as it reduces the barriers to ownership. The asset is available to a bigger pool of investors willing to own a part of the asset, which previously could have been available to wealthy individuals only.

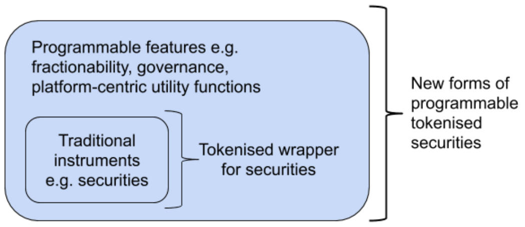

Programmable features e.g. fractionability, governance, platform-centric utility functions
Traditional instruments e.g. securities
Tokenised wrapper for securities
New forms of programmable tokenised securities

Figure D.8: Creating new forms of programmable securities

This flexibility, in augmenting the features of an instrument, offers the ability for tokens to imbibe a large number of attributes that can enable the token to function with multiple roles[13].

Improved Regulatory Compliance

Although counter-intuitive as tokens are perceived to be the tool of anarchists and criminal, tokens actually provided a better platform for compliance and regulatory oversight. Tokenized securities can give regulators the opportunity to apply more compliance and control, rather than less. When we look at regulators today, they are mostly reactive. They spend a lot of time and money on identifying organisations that break the rules. Once identified, it takes years and millions for prosecutions and trials. The tokenization of securities can completely change this status quo,

[13]The roles and attributes discussed in the previous chapter D.3

putting regulators back in control, enabling them to govern proactively.

Since blockchain technology and smart contracts make it possible to write ownership and law directly into a token, the token will be able to execute, regulate and govern itself. For example, a Security Token can be programmed to verify who can buy and sell it and therefore restrict Security Token holders from trading it to any address that has not passed the required verifications, assuring the issuer that their tokens will only be held by authorized investors. Another use case could be the restriction on the token transfer.

Moreover, the whole audit trail and data from the Security Token's creation and compliance processes are uploaded to the transparent and fully auditable blockchain. Regulators will be able to drastically reduce their compliance cost.

Greater Efficiency and Scalability

Greater automation through tokenization can translate into further efficiencies in business processes, as a larger number of previously intermediated tasks have the potential to be automated [73]. For instance:

- Real time trade settlements, enabled via tokenization, can reduce settlement risks
- Disintermediation can remove transaction costs and the relative costs of carrying out a business activity
- Administrative complexities (such as collecting signatures for documents, payments etc.) can be reduced as they are carried out on the token instrument, via smart contracts and digital signatures
- Features such as immutability can provide better auditing and accounting
- Service provider functions, such as trade and some forms of dispute resolution, can be automated through smart contracts

D.6.2 Challenges

Although tokenization has the potential to revolutionise business processes through greater automation and decentralisation, there are a number of challenges that need to be overcome. These include, but is not restricted to:

1. **Scaling:** Greater tokenization and the increased use of tokens will drive the transactional volume of tokens on that need to be processed on blockchains. Current networks do not have the capacity to handle these immense volumes.

2. **Mass adoption:** For adoption to grow beyond technically-savvy audiences and include a broader base of individual users, accessibility of blockchain-related technologies need to play a crucial role.

3. **Regulation:** Tokens are a medium to transact value and manage value. Jurisdictions across the world recognise this as a regulated activity but have no clear regulatory frameworks in place to accommodate them.

4. **Improved Standards:** For greater adoption, scalability and uniformity tokens need to follow uniform standards.

Scaling

Greater tokenization and the increased use of tokens will drive the transactional volume of tokens on blockchains. If tokens are to become widely used the problem of processing high transaction volumes and crowded the network needs to be addressed. This has brought transactional throughput scaling challenges. For example, in the options for addressing include[14]:

- Scaling Ethereum itself so that it can handle greater transaction throughput, projects include Serenity and Casper

[14]Again, most other blockchains address scaling in a similar manner

- Reduce the number of transactions processed on the main chain by moving the bulk of transactions to a second layer and only using the base layer during transaction settlement (e.g. Roll Ups, State Channels)

"Main Chain" or first layer solutions such as Sharding are on the Ethereum roadmap and will additionally be complemented with "Layer Two" scaling mechanisms that further provide even higher throughput, private transactions, and lower transaction fees [42].

Before discussing second layer solutions, consider Ethereum as a settlement ledger as opposed to a "world computer". This means that the purpose of the ledger is to settle transactions which have been conducted off the main chain and enforce value transfers accordingly (It is this use case of the blockchain serving as an unbiased third party for arbitration on which all second layer solutions operate).

At a high level, most layer two solutions follows the approach, or some variation of it [42]:

- Two or more parties agree to a set of rules by which they will be to join and exit the Layer Two solution.
- These parties then encode those rules into a smart contract which requires that each party put down a security deposit.
- After putting down their security deposits, all parties can operate between each other off-chain while submitting intermittent updates to the on-chain smart contract.
- When one or more parties wished to exit the layer two solution, they will typically provide some cryptographic proof that is an accurate representation of each parties' remaining security deposit.
- There is a challenge period where the proof can be disputed and thrown away. If the challenge period elapses, then the related parties will exit the layer two

solution with their updated balances.

Layer Two innovations e.g. Plasma are already processing real payments. Scaling blockchains is tricky but smart contract support can enable novel scaling solutions and greater extensibility than other chains attempting to scale with a second layer on strictly Unspent Transaction Output (UTXO) based scripts, which aren't as extendable, by design [42].

Mass adoption

Tokens need to invest in 'design thinking' as user-friendly design is at the core of any successfully adopted technology [40]. This facilitates the engagement of two main audiences - individual users and bigger organisations. For example, wallet technologies that allow users to transport cryptoassets often require the understanding of highly technical concepts such as addresses and public-private key encryption.

Additionally, not only users but also operators of exchange platforms often face difficulties interacting with traditional fiat-money-based banking facilities in the process of conversion of cryptocurrencies/assets to fiat currency. Nevertheless, institutions such as banks, central banks and governmental organisations are considering adopting tokens in various contexts ranging from increasing intra-bank transaction speed and cost effectiveness to helping ensure the transparency of overseas aid programmes.

Regulation

Currently, the regulations being defined and implemented and governance around cryptocurrencies and initial coin offerings (ICOs) vary greatly across different nations [70, 10, 39]. For example, Switzerland, Gibraltar and Australia have taken a proactive and positive approach towards ICOs; Japan has legitimised Bitcoin by declaring it a legal currency, and allowed Ripple to develop an app to speed up intra-bank transactions, but China took a hard line and banned all ICOs, in addition to cryptocurrency trading and mining in 2017. Like common technical standards, tokenization needs common regulatory standards.

Improving Standards

Although the token community has been active in the development of uniform standards, there is still a void in the development of standardised tokens. For instance, the ERC20 is a very basic standard, it does not deal with more nuanced token behaviours and these are left for the individual to implement as they wish. Common behaviours such as those found in [61] [80] could help create richer standards and promote better token security.

D.7 Token Regulation and the Law

Tokens are a medium to transact and manage value. Jurisdictions across the world recognise this as a regulated activity but have no clear regulatory frameworks in place to accommodate them. A large part of the token eco-system is “self-regulated” through reputation. For mass adoption, tokens will need to have a more explicit regulatory and legal status.

Regulation through Reputation

Reputation and trust are particularly important in the token ecosystem and business models. A good reputation system employed by the network makes the system resilient to manipulation and gaming; and creates a safe environment for network participants to interact and reveal their preferences truthfully. Additionally, a firm’s good reputation and its ability to ensure trust in the ecosystem determines the firm’s ability to signal good quality. This is essential for attracting investors, eventually increasing user engagement and securing financial longevity. Moreover, the bad reputation of certain token ecosystem participants can create a negative externality for ‘good’ actors and prevent the overall ecosystem from growing. As such reputation is seen by the token ecosystem as a mechanism for regulation.

Regulation plays an important role in correcting informational asymmetries and negative externalities when a market is susceptible to potential deficiencies. In the token context, a friendly regulatory environment plays an important role in increasing adoption, but the global and cross-border characteristics of token business models create problems for the traditional regulatory measures available to nation states. This implies the necessity of a global regulatory framework or perhaps a self-governance model in which regulation is largely replaced by good reputation mechanisms. Currently, the regulations being defined and implemented and governance around cryptocurrencies and initial coin offerings (ICOs) vary greatly across different nations. For instance, Switzerland, Gibraltar and Australia have taken a proactive and positive approach towards ICOs; Japan has legitimised Bitcoin by declaring it a legal currency, and allowed Ripple to develop an app to speed up intra-bank transactions, but China took a hard line and banned all ICOs, in addition to cryptocurrency trading and mining in 2017.

Regulators, financial crime regulators globally will be concerned with precisely the same attributes (above) that make crypto-tokens so revolutionary, especially their global tradability on decentralized exchanges. It is quite likely that all crypto-tokens eventually be considered ”monetary equivalents” or “substitutes for value” and therefore be subject to AML/CFT compliance. That means the industry must prepare to comply with, at a minimum, current regulatory expectations with respect to customer/owner identification, transactional analysis for suspicious activity detection and reporting.

When it comes to regulation and the legal premise of tokens it is very important to note that the law is technology agnostic. It regulates relationships between people and situations. It grants rights to persons who meet certain criteria and imposes obligations on persons who meet certain criteria. For instance, in the case of a bilateral agreement between a company and an individual such as employment, the law will provide for certain protections of the individual (such as minimum wage and anti-slavery provisions). However, the commercial relationship between the parties is left to them to agree and can be executed by any mutually acceptable means. The wages of the employee can be as high as the company is prepared to pay but cannot fall below the minimum legal requirement.

D.8 Conclusions

In this paper, we have discussed *Tokenisation*; its evolution, vision and impact, and the requirements for legal status and regulation of tokens. Tokenization and token use can potentially transition into mainstream used and be adopted by a global if the challenges are addressed.

We started off by discussing the characteristics of tokens, their behaviours and classification from literature. We provided an improved taxonomy of tokens and discussed how these could be combined to create tokens that fulfil a given role. We recognise that there is work that needs to be done in this area to better understand the impact of each functional attribute on the token's tokenomics. Possible questions that arise include how anonymity, burnability and expirability affect a tokenization.

We then investigated token standards, their importance and their use currently. We recognise that this is a constantly evolving field and that blockchains, smart contract languages and other tools are constantly being improved and therefore by implication these will be improved as the ecosystem matures.

The benefits of tokenization were then discussed in detail, we recognise that greater liquidity, improved settlement and financial inclusion are the promise of tokenization. We realise that there isn't much formal literature that supports these promises other than ideological arguments put forward by the token community and therefore point out that the extent to which tokenization will realise these arguments needs to be analysed. For example the impact of liquidity on a class of instruments could be analysed if they were to be tokenised or whether parallels from traditional liquidity and risk models be translated into tokenomic equivalents.

Tokenization challenges were then discussed in detail and possible mechanisms to address these. The token community realises that these are major issues and is constantly working to address these (e.g. the use of Sharding to address technical scalability and transaction throughput processing). We recognise that as the tokenization ecosystem evolves newer challenges will arise. However, problems such as scalability will always be a concern as it is a relative predicament that depends on the number of people that use it[15]. The important thing to understand is that these need to be partially addressed for the benefits of tokenization to materialise.

We then discussed tokens and the law. It is important to once again reiterate that the law is technology agnostic and as such its premise it to regulate the activity and not the means. The law is mature enough to regulate tokens as in principle it is the contractual activity carried out by tokens that is regulated. However, more approachable and clear legal frameworks are required to address tokens and their idiosyncrasies. We also discussed how tokens can be used to automate some aspects of the law such as the issuing of certificates.

Trust scales very poorly in our society and one of the main purposes of tokens is to make trust scale up in a world of ambiguity. Any form of value needs to be easy to transact with, easily recognisable or verifiable by users, and easy to carry - for it to work well as a payment system. Many commodities that are currently

[15]In this regard a network like the internet is still scaling, from initially scaling to process thousands of emails, to large files to high definition videos network scalability is a relative measure that depends on the processing needs of the network at a give point in time

utilised to store wealth are not easy to carry or transfer; Bitcoin (or a more evolved cryptocurrency) offers a new alternative.

Potentially, we could even see tokenization of certain economies, in a manner similar to dollarisation. It is also worth noting that the potential of tokens goes well beyond merely creating new forms of money/ assets to compete with existing fiat currencies and financial instruments. Tokenization has the potential to reshape industries, as well as disrupt traditional business models, organisational and governance structures. It encompasses designing entirely new ecosystems and allocation mechanisms based around tokenonomics. This is a multidisciplinary mechanism design challenge that requires an understanding of technology, economics, business, finance, law, psychology and geo-politics.

Appendix E

SmartReg

This appendix presents the screenshots of the UI for the submission portal for the SmartReg project.

E.1 SmartReg UI Screenshots

Figure E.1 shows the React UI for creating a regulatory report submission. The user can choose which regulator the report needs to be made to and the code (reporting rules) with which their data (selected as JSON input) needs to be executed against to generate a report.

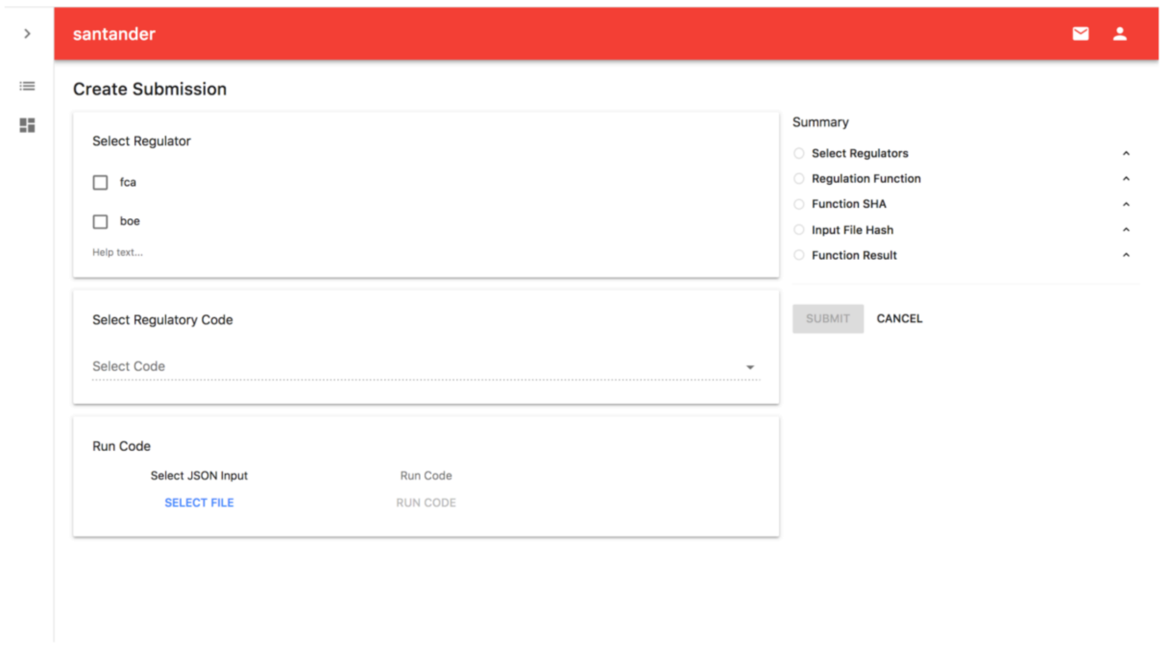
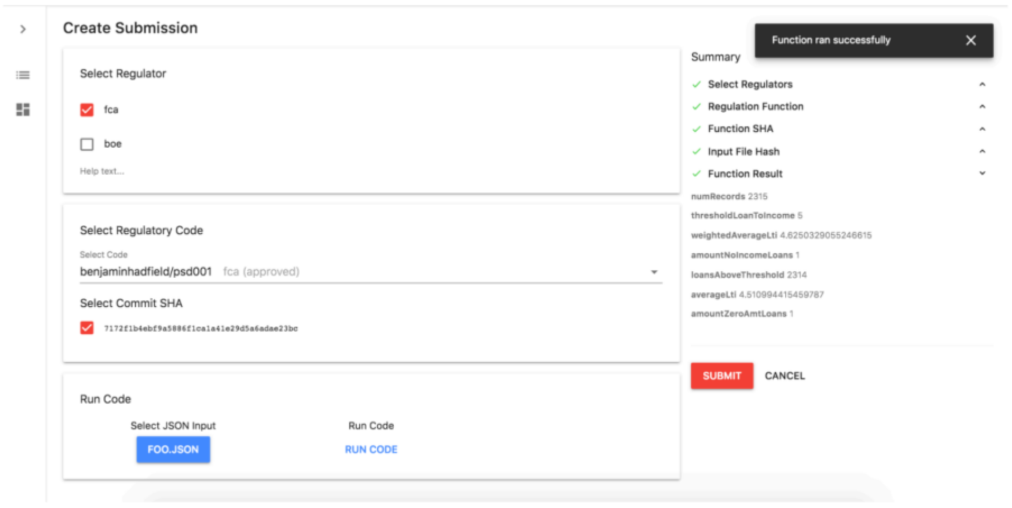


Figure E.1: UI: Empty submissions form

Figure E.2 shows what a completed form looks like. It shows the commit SHAs of the report and the reporting parameters of from the reporting execution (as seen from the right hand column).


Create Submission

Select Regulator

[✓ fca]

boe

Help text...

Select Regulatory Code

Select Code: benjaminhadfield/psd001 fca (approved)

Select Commit SHA

[✓ 7172f1b4ebf9a5886f1ca1a41e29d5a6adae23bc]

Run Code

Select JSON Input: FOO.JSON

Run Code: RUN CODE

Function ran successfully

Summary

✓ Select Regulators

✓ Regulation Function

✓ Function SHA

✓ Input File Hash

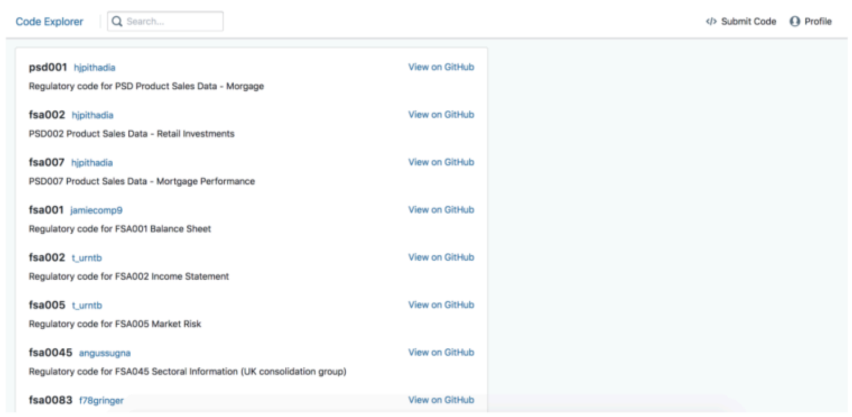
✓ Function Result

numRecords 2315
thresholdLoanToIncome 5
weightedAverageLti 4.6250329055246615
amountNoIncomeLoans 1
loansAboveThreshold 2314
averageLti 4.510994415459787
amountZeroAmtLoans 1

SUBMIT CANCEL

Figure E.2: UI: Fully populated submission form ready to be submitted

Figure E.3 shows the GitHub integration with for the reporting code (i.e. machine readable rules).



psd001 hjpithadia — View on GitHub
Regulatory code for PSD Product Sales Data - Morgage

fsa002 hjpithadia — View on GitHub
PSD002 Product Sales Data - Retail Investments

fsa007 hjpithadia — View on GitHub
PSD007 Product Sales Data - Mortgage Performance

fsa001 jamiecomp9 — View on GitHub
Regulatory code for FSA001 Balance Sheet

fsa002 t_urntb — View on GitHub
Regulatory code for FSA002 Income Statement

fsa005 t_urntb — View on GitHub
Regulatory code for FSA005 Market Risk

fsa0045 angussugna — View on GitHub
Regulatory code for FSA045 Sectoral Information (UK consolidation group)

fsa0083 f78gringer — View on GitHub

Figure E.3: UI: List of regulatory functions available to a Bank

Bibliography

[1] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, et al. Let's encrypt: An automated certificate authority to encrypt the entire web. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2473–2487, 2019.

[2] AminCad. Market share of ethereum based tokens. *Medium*, 2018.

[3] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[4] Guy Aridor, Yeon-Koo Che, and Tobias Salz. The economic consequences of data privacy regulation: Empirical evidence from gdpr. 2020.

[5] Douglas W Arner, Janos Nathan Barberis, and Ross P Buckley. Fintech, regtech and the reconceptualization of financial regulation. 2016.

[6] W ArnerDouglas, P BuckleyRoss, et al. Fintech and regtech in a nutshell, and the future in a sandbox. 2017.

[7] Tomaso Aste, Paolo Tasca, and Tiziana Di Matteo. Blockchain technologies: The foreseeable impact on society and industry - ieee journals & magazine. *Ieeexplore.ieee.org*, 2017.

[8] Auth0. Jwt handbook. *Auth0.com*, 2021.

[9] Financial Conduct Authority. Perg 2 annex 1 authorisation and regulated activities. *FCA Hand Book.*

[10] Financial Conduct Authority. Fca cryptocurrencies guidance. *URL: https://www.fca.org.uk/publications/consultation-papers/cp19-3-guidance-cryptoassets*, 2019.

[11] Financial Conduct Authority. Digital regulatory reporting - feedback statement. *Fca.org.uk*, 2020.

[12] Financial Conduct Authority. Digital regulatory reporting - phase 1 report. *Fca.org.uk*, 2020.

[13] Financial Conduct Authority. Digital regulatory reporting - phase 2 report. *Fca.org.uk*, 2020.

[14] Financial Conduct Authority. Fca handbook. *URL: https://www.handbook.fca.org.uk/*, 2020.

[15] Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. State machine replication in the libra blockchain. *The Libra Assn., Tech. Rep*, 2019.

[16] Sam Blackshear, Evan Cheng, David L Dill, Victor Gao, Ben Maurer, Todd Nowacki, Alistair Pott, Shaz Qadeer, Dario Russi Rain, Stephane Sezer, et al. Move: A language with programmable resources. *Avalaible at: https://developers. libra. org/docs/move-paper (Consulted on April 1, 2020)*, 2019.

[17] Chris Brummer and Daniel Gorfine. Fintech: building a 21st century regulators toolkit. *Milken Institute*, page 5, 2014.

[18] Craig Buckler. Sql vs nosql: The differences sitepoint. *SitePoint*, May 2017.

[19] Talha Burki. Pharma blockchains ai for drug development. *The Lancet*, 393(10189):2382, 2019.

[20] Vitalik Buterin. Ethereum whitepaper. *URL: https://ethereum.org/en/whitepaper/*, 2013.

[21] Addison Cameron-Huff and Addison Cameron-Huff. Op ed: How tokenization is putting real-world assets on blockchains. *Bitcoin Magazine*, 2020.

[22] Christopher D. Clack, Vikram A. Bakshi, and Lee Braine. Smart contract templates: foundations, design landscape and research directions. *Arxiv.org*, 2016. `http://arxiv.org/pdf/1608.00771.pdf`.

[23] ConsenSys. Tokens on ethereum. *URL: https://medium.com/@ConsenSys/tokens-on-ethereum-e9e61dac9b4e*, 2015.

[24] ConsenSys. Tokens on ethereum. *ConsenSys*, 2015.

[25] Nick Cook. From innovation hub to innovation culture. *www.fca.org.uk*, 2019.

[26] Oracle Corp. Trail: Javabeans(tm). *Trail: JavaBeans(TM) (The Java Tutorials)*.

[27] Ronald Cramer, Ivan Damgrd, and Jesper Nielsen. Multiparty computation from threshold homomorphic encryption. *Lecture Notes in Computer Science*, 7, 11 2000.

[28] Jean Cupe. Token ercs in ethereum: Erc-20, erc-223, erc-777 and erc-721. *Medium*, 2018.

[29] Ivan Damgard, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. *Lecture Notes in Computer Science*, pages 643–662, 2012.

[30] Liz Denhup. Understanding unidirectional data flow in react. *Medium*, 2021.

[31] Solidity Documentation. Solidity ethereum.

[32] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[33] React Facebook. React.

[34] Internet Engineering Task Force. Http authentication. *Tools.ietf.org*, 2021.

[35] E Friedmann-Hill. jess in action. *Greenwich, CT: Manning*, 2003.

[36] Rose F Gamble, PR Stiger, and RT Plant. Rule-based systems formalized within a software architectural style. *Knowledge-Based Systems*, 12(1):13–26, 1999.

[37] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[38] Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[39] UK Government. Crypto assets taskforce report. *Assets.publishing.service.gov.uk*, 2018.

[40] Zeynep Gurguc and William Knottenbelt. Cryptocurrencies- overcoming barriers to trust and adoption. *Imperial.ac.uk*, 2018.

[41] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. Gossiping gans. *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning - DIDL '18*, 2018.

[42] Hunter Hillmann. The case for ethereum scalability. *Medium*, 2019.

[43] Dongyan Huang, Xiaoli Ma, and Shengli Zhang. Performance analysis of the raft consensus algorithm for private blockchains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):172–181, 2019.

[44] A Hunt and D Thomas. Orthogonality and the dry principle. *Artima.com*, 2003.

[45] Sandia Inc. Jess, the rule engine for the java platform. *Jess, the Rule Engine for the Java Platform*. `http://www.jessrules.com/`.

[46] Fleur Johns and Caroline Compton. Data jurisdictions and rival regimes of algorithmic regulation. *Regulation & Governance*, 2020.

[47] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[48] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham, 2017. Springer International Publishing.

[49] Leslie Lamport. Generalized consensus and paxos. 2005.

[50] Kari Larsen, Shariq Gilani, et al. Regtech is the new black-the growth of regtech demand and investment. *Journal of Financial Transformation*, 45:22–29, 2017.

[51] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe. Reguard: Finding reentrancy bugs in smart contracts. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 65–68, 2018.

[52] Poly Math. *Medium*, 2018.

[53] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv.org*, 2020.

[54] Georgios Meditskos and Nick Bassiliades. Clips–owl: A framework for providing object-oriented extensional ontology queries in a production rule engine. *Data & Knowledge Engineering*, 70(7):661–681, 2011.

[55] C Mohan. Blockchains and databases: A new era in distributed computing. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1739–1740. IEEE, 2018.

[56] MongoD.

[57] Henrique Moniz. The istanbul bft consensus algorithm. *arXiv preprint arXiv:2002.03613*, 2020.

[58] JP Morgan. Quorum. *URL: https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf*, 2021.

[59] Satoshi Nakamoto. Bitcoin whitepaper. *URL: https://bitcoin.org/bitcoin.pdf*, 2008.

[60] Crispin Niebel. The impact of the general data protection regulation on innovation and the global political economy. *Computer Law & Security Review*, 40:105523, 2021.

[61] Luis Oliveira, Liudmila Zavolokina, Ingrid Bauer, and Gerhard Schwabe. To token or not to token: Tools for understanding blockchain tokens. *Jacobs Levy Equity Management Center for Quantitative Financial Research Paper Series*, 2020.

[62] OpenMined. What is secure multi-party computation? *OpenMined Blog*, 2020.

[63] Tim OReilly. Open data and algorithmic regulation. *Beyond transparency: Open data and the future of civic innovation*, 21:289–300, 2013.

[64] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and lfar Erlingsson. Scalable private learning with pate. *arXiv.org*, 2020.

[65] A Rahman. A framework for decentralised trust reasoning. *University of London*, 2005.

[66] Leonard Richardson and Sam Ruby. *RESTful web services.* " O'Reilly Media, Inc.", 2008.

[67] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[68] Ermin Sakic and Wolfgang Kellerer. Response time and availability study of raft consensus in distributed sdn control plane. *IEEE Transactions on Network and Service Management*, 15(1):304–318, 2017.

[69] Susan V Scott and Markos Zachariadis. Origins and development of swift, 1973–2009. *Business History*, 54(3):462–482, 2012.

[70] Securities and Exchange Commission. Framework for investment contract analysis of digital assets. *Sec.gov*, 2019.

[71] Charles Severance. Roy t. fielding: Understanding the rest style. *Computer*, 48(6):7–9, 2015.

[72] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[73] Samridha Shreshtha. Why a distributed system for the trading industry instead of a centralised one. *Medium*, 2018.

[74] Małgorzata Śmietanka, Hirsh Pithadia, and Philip Treleaven. Federated learning for privacy-preserving data access. *Available at SSRN 3696609*, 2020.

[75] J Michael Spivey and JR Abrial. *The Z notation*. Prentice Hall Hemel Hempstead, 1992.

[76] Philip Treleaven, Bogdan Batrinca, et al. Algorithmic regulation: Automating financial compliance monitoring and regulation using ai and blockchain. *Journal of Financial Transformation*, 45:14–21, 2017.

[77] AI Council — Gov UK. Ai roadmap. *Assets.publishing.service.gov.uk*, 2021.

[78] Tom Van Engers. Estrella- estrella project website.

[79] Huw Van Steenis. The future of finance. *Bankofengland.co.uk*, 2021.

[80] Fabian Vogelstellar and Vitalik Buterin. Eip-20: Erc-20 token standard. *Ethereum Improvement Proposals*, 2015.

[81] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

[82] Sandra Wachter. Normative challenges of identification in the internet of things: Privacy, profiling, discrimination, and the gdpr. *Computer law & security review*, 34(3):436–449, 2018.

[83] Mark Walport. Fintech futures: The uk as a world leader in financial technologies. *Report to UK Government Office for Science*, 2015.

[84] Amir Bandeali Will Warren. 0x protocol. *URL: https://0x.org/pdfs/0x*$_w$*hite*$_p$*aper.pdf*, 2020.

[85] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[86] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[87] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[88] Andrew Chi-Chih Yao. How to generate and exchange secrets - ieee conference publication. *Ieeexplore.ieee.org*, 1986.

[89] Karen Yeung and Martin Lodge. *Algorithmic regulation*. Oxford University Press, 2019.