Can SAFE Contracts be Smart? (DRAFT)

Ron van der Meyden
UNSW Sydney

Michael Maher
Reasoning Research Institute

January 24, 2022

Abstract

This paper conducts a case study of the representation of legal contracts as smart contracts, focussed on a key clause of Y Combinator's Simple Agreements for Future Equity (SAFE), a class of financial instruments used in funding startups. A number of challenges for such representation are identified, including indeterminacy of language and incompleteness in the legal text, and the open nature of the context in which these contracts operate. Approaches to dealing these challenges are proposed and a smart contract architecture and an implementation on the Ethereum platform are developed. An analysis of the implementation leads to the conclusion that the smart contract needs to be adaptable to legal rulings and backed by a legal contract. Content for such a legal contract is proposed.

Contents

1 Introduction

The ability to execute "smart contract" [Sza97] programs on distributed ledgers that decentralize control of the execution environment has enabled novel mechanisms for the enforcement of security properties in business relationships. In some applications of this technology, the programs do not have legal import, but many of the most significant (including Initial Coin Offerings, Security Token Offerings, Decentralized Autonomous Organisations, and Decentralized Finance) involve equity-like rights recognized in law. Although some proponents have hoped that the technology would be able to operate independently of legal systems, regulator actions have already asserted that such rights represented in code on distributed ledger platforms are subject to the law. This has given renewed impetus to the idea of executable formal representations of legal contracts [DCP19a].

In this paper, we explore to what extent, and how, smart contracts can be used to implement legal contracts. We do this by means of a case study using a key clause of versions of Y Combinator's Simple Agreement for Future Equity (SAFE) [Y C16b]. This is a contract where an investor purchases a right to receive shares in a company at the time of its next equity financing. SAFEs, and similar contracts, are widely used to finance start-up companies. The SAFE is ideal for an initial case study. It involves only two parties, one of which has few obligations. It was designed to be simple and layperson-friendly [CG15], and it consists of only 6 pages.

The financial and quantitive nature of SAFEs might lead one to expect that they can be straightforwardly translated to smart contract code. In fact, we identify several challenges to this goal. Principal amongst these is a problem that has long been recognized as one of the main challenges to realizing the vision of intelligent legal information systems, the lack of formality of natural language legal text [McC80]. Natural language legal text is often *indeterminate*, allowing multiple interpretations. One cause of this is that natural language concepts are often *open-textured* [Har61, Wai68, Wit53]: unlike mathematical concepts and conditions in computer code, they do not have fixed, exact, "if and only if" definitions. Instead they are expressed using imprecise terms that are underpinned by a body of concrete examples, e.g., from case law, that evolves over time as a result of court rulings. Indeterminacy presents a difficulty in designing smart contracts to implement legal contracts: smart contracts must be determinate (otherwise, there can be a failure of consensus of the validators), and so can represent only one of the possible meanings of an indeterminate contract. We identify several instances of indeterminacy that raise difficulties for the formal representation of SAFE contracts.

A further challenge for formalization of SAFEs is the *open* nature of what needs to be protected using the DLT platform: SAFEs come in myriad forms, which are subject to negotiation with the investors. Moreover, the governance structures imposed on the company (its board rules, voting rules, and constraints on its officers) as it matures, are open ended. Also raising issues for a direct translation to code is the fact that some versions of SAFEs do not directly

state how the number of shares to be issued when converting a SAFE to shares should be calculated, but instead state equations constraining the issuance.

To understand the impact of these challenges for the goal of supporting SAFE contracts using smart contracts, we develop in this paper an architecture and implementation of smart contracts covering the key Equity Financing clause of a number of variants of SAFE contracts, which we then evaluate to determine how satisfactorily they represent the original legal text.

The architecture and implementation applies a number of strategies for dealing with the challenges discussed above. To address indefiniteness, we first note that smart contract development differs from the general problem of automation of legal reasoning in that a smart contract may itself restrict the patterns of facts and events that need to be reasoned about. We develop an approach to representation of the scenario that enables flexibility in both the financial contracts that the company issues and the governance structures adopted. In particular, we abstract expressions from the legal text in a way that allows for indefiniteness in the legal text to be flexibly addressed. The abstraction allows parties to the contract to choose an approach to resolution of indefiniteness that they can agree upon. The possibilities are drawn from a spectrum, from full automation using formalized terms, to semi-automated processes in which human agents decide the satisfaction of an indefinite term at the time a decision concerning the term needs to be made in performing the contract.

One of the principles of systems architecture, particularly in the context of secure systems development, is to isolate critical functionality in components that are small enough to be verified (ideally, using formal methods), combined in the architecture in ways that ensure that the basis supporting key properties remains small. We apply this methodology, representing a company that issues SAFE contracts using a collection of smart contract modules, each of which ensures some key properties of the overall application. Key to the architecture are some known smart contract patterns: the *owner* pattern and the *atomic swap* pattern. We show how these patterns support the open ended nature of the application, and can be combined to enforce the Equity Financing clause.

To evaluate the resulting implementation, we first note that indefiniteness in the text of a legal contract being formalized, compared to the definiteness of any smart contract implementation automating the indefinite terms, places inherent limits on claims that the smart contract is a "correct" implementation. This means that we are necessarily limited to testing of the implementation in specific cases, which cannot provide absolute correctness guarantees. There is always the risk that the code will make an incorrect decision in unforeseen examples.

Moreover, we argue that even where indefinite terms have been handled using human decision making, the SAFE contract is subject to being "attacked" in a collusion between the company and an equity round investor so as to minimize the shareholding of the SAFE investor. The SAFE legal contract contains language that protects the SAFE investor against this attack by allowing a legal challenge to be raised, after the fact, when such a collusive manipulation of the contract has occurred. By contrast, the smart contract implementation on

its own, which can only enforce terms "in the present", would not protect the investor against the attack.

For this, and other reasons, we conclude that use of the SAFE smart contract needs to be backed by a legal contract that describes its intent, requirements on its use, and how issues in the operation of the contract will be handled. We discuss related issues concerning legal acceptability of the SAFE smart contract, and sketch the terns that such a legal backing contract should contain, but leave a detailed development of this contract for future work.

Finally, we also discuss approaches that might be used to ensure that a smart contract implementation of a legal SAFE contract is adaptive to legal rulings concerning its operation. Beyond the issues discussed above, such adaptiveness is necessary to handle code and platform vulnerabilities and legal interventions.

The structure of the paper is as follows. Section 2 gives further background and motivation for smart contract representations of contracts used in equity financing. Section 3 outlines relevant aspects of SAFE contracts. In Section 4 we discuss indeterminacy and other issues occurring in SAFE contracts that create impediments to implementing them as smart contracts. Section 5 addresses how these impediments might be addressed. In Section 6, we review a number of design patterns used in smart contract programming, which we use in our smart contract design. Section 7 describes the architecture and implementation of smart contracts for modelling a company and its use of smart contracts representing SAFE contracts. Section 8 argues informally that the implementation satisfies a number of technical security properties.

Section 9 then considers how to evaluate these smart contracts in comparison with the legal contract they attempt to implement. Legal issues around the need for a legal contract backing the smart contract SAFE implementation are discussed in Section 10. Methods for handling legal orders for variation of the smart contract are discussed in Section 11. Finally, Section 12 discusses some related work and Section 13 contains concluding remarks.

Our focus for the bulk of the paper is on the key Equity Financing clause of SAFE contracts. An appendix considers other parts of the legal SAFE contracts, and sketches how these might be implemented to complete the work of this paper to a full smart contract representing a SAFE. A second appendix contains preliminary thoughts on the issue of which DLT platforms would be appropriate to use for our SAFE smart contracts, in particular, whether public or private chains are best, in view of privacy considerations.

We assume that the reader has a basic familiarity with the notions of blockchain and smart contracts. Our main points in this paper are applicable to most blockchain systems and smart contract languages, but for concreteness, we present technical details in the setting of the Ethereum blockchain and the Solidity smart contract programming language.

2 Motivation

The advent of distributed ledger technologies has made it possible for ownership and trading of digitally represented assets to be secured in ways that obviate the need for centralized trusted parties such as banks, brokers, exchanges and registries. Starting with digital forms of money, in the form of Bitcoin [Nak08], increasingly complex forms of digital asset are being implemented. This has been enabled by the ability for "smart contracts" on distributed ledger platforms to secure not only facts about ownership, but also to automatically enforce rules governing the creation and transfer of ownership.

One of the early applications of this capability has been the representation of rights in financing of early stage ventures, via "Initial Coin Offerings" (ICO). These are are a form of fundraising in which digital money represented on the distributed ledger is exchanged for a "token", a digitally represented right to some aspect of the outcomes of the project to which the funds are applied. Tokens issued have represented rights including ownership of new forms of digital money created by the project, rights to some form of "utility", e.g., the right to exchange the token for services and transaction fees on the digital platform developed by the project, as well as rights to a share of the profits from the project.

Many of these fundraises were conducted in ways that violate laws governing equity financing, prompting regulator actions, but the phenomenon has transitioned into "Security Token Offerings", which seek to offer digital tokens representing equity in ways that remain compliant with securities law. Not just the crypto-currency "underground" but also traditional financial sector organizations are developing digital ledger platforms that represent equity rights. For example, the Australian Stock Exchange is developing a distributed ledger platform to replace its existing clearance and settlement system.

Benefits of the digital representation of equity rights on distributed ledger platforms are perceived to include:

- Improved efficiencies from the establishment of a consensus "single source of truth" concerning ownership of rights. Current processes in financial markets frequently involve expensive manual reconciliation of inconsistent records dispersed across multiple organizations, delaying trades and allowing central parties to extract rents from collateral held in trust during the delay.
- Enabling the rights holder or their proxy to uniquely control ownership and transfer of the asset via possession of a private cryptographic key.[1]
- Elimination of counter-party risks by ensuring atomicity of exchange transactions, e.g., a swap of an asset for digital currency.

[1] Since keys are easily distributed, it may be controversial to regulators that direct control by the rights holder is a benefit. Instead, they may require not just the possibility of unique ownership, but also unique custody of the asset by a *known* entity. Although it is less within the ethos of the cryptocurrency community, custody of the key by a proxy trusted by the regulator may be a way to address these concerns.

- Improved security and availability of the rights ownership data, through use of cryptographic mechanisms and avoidance of single points of failure.
- Automating the enforcement of rules constraining the issuance and transfer of rights, such as vesting schedules and lockups. Exchanges may require, for protection of their customers, constraints on management's power to create and issue new shares without a shareholder vote. By placing these events under the control of a smart contract whose state is accepted to be a legally valid representation of the ownership of the company, investors receive a stronger guarantee of compliance with such rules.
- Increasing global accessibility of investment opportunities and the creation of a 24/7 market.

Direct sale of shares is just one of the forms of financing used by companies; others are loans and more complex instruments that mix the properties of loans and equity. In particular, in financing of early stage ventures it has become common to use "convertible bonds", a contract that initially gives the owner rights to principal and interest on a loan, but which also enables these rights to be converted into equity in certain circumstances.

The Y Combinator Simple Agreement for Future Equity (SAFE) [Y C] was designed as a simple form of such instruments [CG15]. In particular, the design is biased more to equity than debt, by removing interest payments, and being intended to convert to shares in most cases, rather than allowing recovery of principal. SAFE contracts are used primarily by early stage ventures to raise funding from "angel investors" and incubator funds. Variants are also being used for crowdfunding, to the extent that SEC has issued an Investor Bulletin on the topic [SC]. The latter use case makes SAFE contracts close to fundraising techniques used in the cryptocurrency community. Our focus in this paper is to understand the representation of the SAFE conversion conditions using smart contracts, in order to allow the above benefits of representation of equity rights on blockchain platforms to be obtained for SAFE contracts.

3 Summary of SAFE Contract Structure

SAFE contracts come in a number of forms. As initially issued by Y Combinator, in a form we call "Pre-money SAFE contracts", to distinguish from the later "Post-money" versions, these contracts have two parameters, a "Valuation Cap" and a "Discount", that may or may not be included, giving four distinct contracts. The Valuation Cap gives a maximum valuation used to calculate a price at which the SAFE principal will be converted to shares, and the Discount relates to a percentage discount given on the price. These contracts have a common overall structure, consisting of the following sections:

- *Certification*: this identifies the parties to the contract (the Company and the Investor), as well as the values of the parameters (Date, Purchase

Amount, Valuation Cap and/or Discount) that are included in the particular SAFE, and states that the Company issues to the Investor rights to certain shares subject to terms in the following sections.

- *Events*: this lists events of relevance to the execution of the contract (Equity Financing, Liquidity Event, Dissolution Event and Termination), and describes the consequences of each.
- *Definitions*: this defines a number of terms (Capital Stock, Change of Control, Company Capitalization, Distribution, Dissolution Event, Equity Financing, Initial Public Offering, Liquidity Capitalization, Liquidity Event, Liquidity Price, Pro Rata Rights Agreement, Safe, and Safe Preferred Stock) that are used elsewhere in the contract.
- *Company Representations*: contains a number of assertions made by the company that underpin the company's capacity to enter into the contract and the legal validity thereof.
- *Investor Representations*: similarly contains a number of assertions by the investor that underpin their legal capacity to enter into the contract.
- *Miscellaneous*: describes a number of provisos relating to revisions to the contract, delivery of relevant notices, rights not implied by the contract, transfer of rights associated to the contract, treatment of invalidity of part of the contract, and the legal jurisdiction governing the contract.
- *Signature page*: where the contract is signed by the investor and a representative of the company.

For definiteness, we address the Pre-money SAFE with cap and no discount [Y C16a], and we focus on equity financing. This is defined in the SAFE as follows:

> "Equity Financing" means a bona fide transaction or series of transactions with the principal purpose of raising capital, pursuant to which the Company issues and sells Preferred Stock at a fixed pre-money valuation.

The key clause of the SAFE that we focus on in the present paper is the following, from the Events section:

> 1(a) **Equity Financing.** If there is an Equity Financing before the expiration or termination of this instrument, the Company will automatically issue to the Investor either: (1) a number of shares of Standard Preferred Stock equal to the Purchase Amount divided by the price per share of the Standard Preferred Stock, if the pre-money valuation is less than or equal to the Valuation Cap; or (2) a number of shares of Safe Preferred Stock equal to the Purchase Amount divided by the Safe Price, if the pre-money valuation is greater than the Valuation Cap.

The Equity Financing clause contains some further provisions that are not our main focus, but which do have some bearing on how we formalize the above:

> In connection with the issuance of Standard Preferred Stock or Safe Preferred Stock, as applicable, by the Company to the Investor pursuant to this Section 1(a):
>
> (i) The Investor will execute and deliver to the Company all transaction documents related to the Equity Financing; provided, that such documents are the same documents to be entered into with the purchasers of Standard Preferred Stock, with appropriate variations for the Safe Preferred Stock if applicable, and provided further, that such documents have customary exceptions to any drag-along applicable to the Investor, including, without limitation, limited representations and warranties and limited liability and indemnification obligations on the part of the Investor; and
>
> (ii) The Investor and the Company will execute a Pro Rata Rights Agreement, unless the Investor is already included in such rights in the transaction documents related to the Equity Financing.

The notion of "price per share of the Standard Preferred Stock" is treated as a primitive input available at the time of the equity round, and the "Valuation Cap" is a parameter in the Certification section that is filled in when instantiating the contract before signing. The Safe Price is defined by the following clauses in the Definitions section:

> **"Safe Price"** means the price per share equal to the Valuation Cap divided by the Company Capitalization.
>
> **"Company Capitalization"** means the sum, as of immediately prior to the Equity Financing, of: (1) all shares of Capital Stock (on an as-converted basis) issued and outstanding, assuming exercise or conversion of all outstanding vested and unvested options, warrants and other convertible securities, but excluding (A) this instrument, (B) all other Safes, and (C) convertible promissory notes; and (2) all shares of Common Stock reserved and available for future grant under any equity incentive or similar plan of the Company, and/or any equity incentive or similar plan to be created or increased in connection with the Equity Financing.

Y Combinator varied their standard contracts in September 2018, introducing a "Post-Money" version of the SAFE, which also comes in four versions. The overall structure of these documents is as above, but, in the case of the Post-Money SAFE with cap and no discount [Y C18b], the cap parameter in the certification is called a "Post-money Valuation Cap", and the Equity Financing clause is changed to

> **Equity Financing.** If there is an Equity Financing before the termination of this Safe, on the initial closing of such Equity Financing,

> this Safe will automatically convert into the greater of: (1) the number of shares of Standard Preferred Stock equal to the Purchase Amount divided by the lowest price per share of the Standard Preferred Stock; or (2) the number of shares of Safe Preferred Stock equal to the Purchase Amount divided by the Safe Price.

and the related definitions state

> **"Safe Price"** means the price per share equal to the Post-Money Valuation Cap divided by the Company Capitalization.
>
> **"Company Capitalization"** is calculated as of immediately prior to the Equity Financing and (without double-counting):
>
> - Includes all shares of Capital Stock issued and outstanding;
> - Includes all Converting Securities;
> - Includes all (i) issued and outstanding Options and (ii) Promised Options;
> - Includes the Unissued Option Pool; and
> - Excludes, notwithstanding the foregoing, any increases to the Unissued Option Pool (except to the extent necessary to cover Promised Options that exceed the Unissued Option Pool) in connection with the Equity Financing.
>
> **"Converting Securities"** includes this Safe and other convertible securities issued by the Company, including but not limited to: (i) other Safes; (ii) convertible promissory notes and other convertible debt instruments; and (iii) convertible securities that have the right to convert into shares of Capital Stock.

While the Post-Money SAFE differs from the Pre-Money SAFE in multiple ways (see Appendix III of [Y C18a]), there are two changes that are central to the understanding of the equity financing clause. First, the conditioning on the "pre-money valuation" to select the cases in the Equity Financing clause is replaced by a maximization over the number of shares. (This change resolves what is arguably a deficiency of the Pre-Money SAFE: we refer to [vdMM21] for a detailed discussion of this point.) Second, whereas "Company Capitalization" in the Pre-money SAFE excludes the number of shares issued in conversion of SAFE contracts, these shares are included in the corresponding definition in the Post-Money SAFE. This in turn affects the calculation of the Safe Price, and the number of shares issued in conversion of the SAFE. A significant issue that this raises, taken up below, is that whereas the Pre-Money SAFE gives an explicit definition of the number of shares to be issued (depending on some undefined terms), the Post-Money SAFE has a circular definition, that can be understood as stating a set of constraints that need to be solved in order to determine the shares to be issued to a SAFE investor. (See [vdMM21] for detailed discussion of the constraints and their solution.)

To focus on the key issues of the paper, we will make the simplifying assumption for the remainder of the paper that the company ensures that there are no shares outstanding, issues no converting securities other than SAFE contracts, does not issue or promise options, and does not maintain an Unissued Option Pool.

4 Impediments to Formalization of SAFE Contracts

Legal contracts are intended for human-operated processes such as execution, compliance, negotiation, arbitration and litigation, so are necessarily expressed in natural language. A consequence of this is that legal contracts are often subject to indeterminacy, which presents difficulties for their formal representation as computer code in smart contracts. One might expect that contracts in the narrow, technical domain of corporate finance are more amenable to formalization. In fact, even in this technical domain, natural language documents may suffer from indeterminacy. In the present section, we identify a number of specific ways that this difficulty arises for SAFE contracts.

We begin with a brief review of terminology used in the legal literature to describe forms of indeterminacy in legal texts, and some of ways that the law may deal with the difficulties this causes. We then identify some particular occurrences of indeterminacy in SAFEs. Our focus in this section is on identifying issues that need to be dealt with: subsequent sections describe strategies for dealing with these issues, the actual formalization we have developed that applies these strategies, and its evaluation.

4.1 Indeterminacy in Legal Text

There is a variety of forms of indeterminacy, and a variety of overlapping terminologies (which themselves can suffer from vagueness).

One way that contracts can be indeterminate is through *incompleteness*, the failure to define terms or stipulate the consequences of all possible contingencies. Empirical studies show that business contracts are often incomplete [Mac63]. This observation is the starting point for an entire area of economics [Har17].

The term *open texture* was introduced by Waismann [Wai68] to refer to the problem that an empirical concept, however carefully defined, might have uncertain applicability in unforeseen circumstances. The Oxford Dictionary of Philosophy [Bla96] gives an illuminating example. In general, the term *mother* is quite clear. Nevertheless, advances in reproductive technology have led to the situation where past uses of the term have not distinguished between "the mother that produces the ovum, the mother that carries the foetus to term, and the mother that rears the baby". Questions of applicability of concepts in unforeseen situations can be an issue even in the rigorous setting of mathematics: [Lak76] showed that, historically, the concept of *polyhedron* has repeatedly had to be adapted to cope with unforeseen examples and counterexamples.

Hart [Har61, Har58] reused Waismann's term, in application to Law, but seemingly expanded it to include vagueness, which Waismann had distinguished from his notion of open texture. [Sch13] views Waismann's open texture as *potential* vagueness that circumstances might bring to light, whereas Hart's notion also encompasses *actual* vagueness, such as use of the term "vehicle", which could have been made more precise. Such vagueness can be strategic in a legal text, making a broad claim of applicability and leaving exceptions to be handled later. In any case, a degree of vagueness is unavoidable. Thus, a consequence of the existence of open texture and vagueness in legal text is that adjudicators can be called upon to apply judgement in its interpretation (judicial discretion) [RM20]. As a result of rulings by the adjudicators in specific cases, the understanding of the concepts evolves over time.

Works vary in how they interpret "ambiguity" in discussing indeterminacy. We prefer to restrict that term to situations where the use of words or phrases with double meanings, the structure of the text, or inconsistency in the text, lead to indeterminacy, as distinct from vagueness of language. For example, a military regulation requiring a commander to "sanction unauthorized actions performed while out of communication with central command", is ambiguous, based on the dual meanings of "sanction". Of course, the context of the surrounding regulation is likely to disambiguate the meaning of this text. A further distinction is made between *patent ambiguity*, which is apparent from the legal text alone, and *latent ambiguity*, where the text is clear but the particular context makes the meaning unclear [Ano11]. [Sch02] analyses cases of legal ambiguity, distinguishing vagueness of categorization, lexical ambiguity, and referential indeterminacy. The military example is a case of patent lexical ambiguity.

In many jurisdictions of contract law, the doctrine of *contra proferentem* attempts to resolve such situations when the context and ordinary, natural meaning of the words are not sufficient. It is described variously as preferring the interpretation that works against the drafter of the contract, or the party with greater bargaining power, or the party seeking to rely upon the ambiguous text [Ins81, AOT12].

There is a further notion of defeasibility in the law: that apparent straightforward application of the plain language of a law can fail due to specific circumstances. A classic example is the 1889 case of a man who killed his grandfather and then claimed to inherit the estate[2]. Despite clarity of the will and the law that the grandson should inherit the estate, the court ruled against the grandson, based on the principle that "no man should profit from his own wrong".[3] This behaviour is distinct from issues of open texture, vagueness and ambiguity.

4.2 Open-textured Concepts in SAFEs

An example of the phenomenon of open texture in the case of SAFE contracts is the term "Equity Financing" used in the key clause of the SAFE contract.

[2] *Riggs v. Palmer* in the New York Court of Appeals, 1889.

[3] A more recent occurrence of a similar situation is the question of whether a presidential pardon could, in effect, pardon the pardoner.

The SAFE contract defines this term this as follows:

> **"Equity Financing"** means a bona fide transaction or series of transactions with the principal purpose of raising capital, pursuant to which the Company issues and sells Preferred Stock at a fixed pre-money valuation.

This definition contains terms that raise multiple issues of interpretation.

- What does it mean for a transaction to be "bona fide"? This is legal term denoting that the transaction is in "good faith", and not fraudulent. This is a prime example of an "open-textured" concept, that is hard to give an a priori formalization. One source of unforeseen circumstances – which, nevertheless, is foreseeable – is *aliasing*: the situation where two (or more) nominally different roles are taken by the same entity. Concerning a SAFE contract, it is, in theory, possible that the SAFE investor and the company (or its founders) are the same entity, the founders might then secure themselves preferred shares in an equity round, whereas they would usually have common stock subject to vesting. If, instead, the founders are also the equity investor, then they can manipulate the valuation to their own advantage, and to the detriment of SAFE investors. An artificial increase in the valuation will reduce the number of shares obtained by SAFE investors, leaving the existing shareholders with a greater share of the company, and the company in essentially the same financial state. This behaviour is guarded against by the restriction to "bona fide ... transactions with the principal purpose of raising capital". However, the question of whether this concept applies in a particular case may require resolution in court.

- The definition explicitly allows that the Equity Financing event consists of a "series of transactions". This raises the question of how we know which transactions are part of the event. That the sale of Preferred shares should be at a "fixed pre-money valuation" in the Pre-money SAFE suggests that two sales of Preferred Shares, at prices that imply different pre-money valuations, are not both part of the same Equity Financing event. On the other hand, suppose there is a sale of Preferred Shares, followed one week later by another sale of Preferred Shares at the same pre-money valuation. Are these part of the same Equity Financing event or not? Answering this question is likely to involve considering the suite of documents that typically capture the details of the financing round [NVC], and even its history. (Different versions of SAFE may lead to different conclusions: that the Post-Money SAFE refers to the "lowest price per share of the Standard Preferred Stock" implies an allowance that not all new investors may pay the same price.)

- How do we determine whether a transaction was done "with the principal purpose of raising capital"? The relevant facts for such a determination are likely to be the actions of human agents (their spoken assertions and

written communications, and other real-world actions) that may be complex to interpret. An example of a transaction not done for purposes of raising capital is an issuance of stock in exchange for services.

The apparent difficulty for formalization of these concepts is that it is unclear what primitive concepts should be included, that potentially the volume and complexity of relevant detail could be very large, and that evolving case law may both broaden the scope of relevant detail and force changes to whatever formalization is adopted.

For another example, the Equity Financing clause in the Post-Money SAFE leaves a small lacuna: in the event that cases (1) and (2) of this clause produce the same number of shares, it is not specified what result is entailed. Assuming "greater" is interpreted as greater or equal, it still is not clear which type of shares – Safe Preferred or Standard Preferred – should be issued to the SAFE investor: the text is then inconsistent, although the difference may be immaterial if, as seems to be the intention, the only difference between Safe and Standard Preferred shares is the price paid. This is a latent ambiguity, although to some it might be patent. (This issue is precisely the sort of "edge case" that can be detected as the benefit of a careful formalization and verification.)

Although this seems too minor to reach court, under *contra proferentem* (on the "against drafter" interpretation), Y Combinator, as SAFE investor, would receive Standard shares[4], since they drafted the contract; otherwise, (on the "against party with greater bargaining power" interpretation), the SAFE investor might expect Safe Preferred shares since the company, as party to the transaction that sets the valuation, has the power to influence that valuation. However, if the company issues Standard shares to the SAFE investor, who then sues for Safe shares, it is the SAFE investor who is relying on the ambiguous text and so might lose under *contra proferentem* (on the "against relying party" interpretation). This variance of conclusions depending on the interpretation of the legal principle itself adds to the indefiniteness of the contract!

4.3 Incompleteness

A further issue with natural language contracts that raises difficulties for their representation as smart contracts is that they have a structure that differs significantly from code. Whereas imperative code (provided it terminates) has a logical flow that provides an output for every input, natural language contracts are expressed in a collection of declarative sentences, which state conclusions about particular general situations. The coverage of all possible "input situations" may be incomplete. Moreover, some terms may be left undefined.

An example of incompleteness that arises in the case of the Pre-Money SAFE contract with cap and no discount is that the text makes use of the terms "pre-money valuation" (the new investor's valuation of the company before an equity round) and "price" (of shares paid by an investor in an equity round), but does not state how these concepts are related.

[4]Under the presumption that Safe Preferred shares are preferable to Standard shares.

In fact, these are terms of art in the domain of venture finance, and in the context of most equity rounds, are related by

$$\text{price} = \frac{\text{pre-money valuation}}{\text{pre-money capitalization}} \tag{1}$$

where "pre-money capitalization" is the number of shares issued by the company prior to the equity round. Moreover, in a typical equity round, we also have

$$\text{price} = \frac{\text{post-money valuation}}{\text{post-money capitalization}}$$

where

$$\text{post-money valuation} = \text{pre-money valuation} + \text{new investor's money} \tag{2}$$

and

$$\text{post-money capitalization} = \text{pre-money capitalization} + \text{new investor's shares} .$$

These equations have the intuitively correct consequence that, in the course of the equity round, the new investor exchanges their money for shares of equal value. One might therefore expect that the incompleteness is resolved by these contextually understood equations.

However, when a consequence of the equity round is the forced issuance of shares to the SAFE investors in conversion of their SAFEs, we in fact have

$$\begin{aligned} \text{post-money capitalization} = \; & \text{pre-money capitalization} + \\ & \text{new investor's shares} + \\ & \text{new SAFE shares} \end{aligned}$$

with the consequence that the new investor is immediately diluted as a consequence of the round, and receives shares of lesser value than their investment. The sophisticated new investor will therefore insist on a lower price than is implied by equation (1). One way to achieve this is to instead determine the price from valuation using equation

$$\text{price} = \frac{\text{pre-money valuation}}{\text{pre-money capitalization} + \text{new SAFE shares}} \tag{3}$$

which preserves the conclusion that the new investor receives shares of value, after the round, equal to their money invested. (In effect, the Post-Money SAFE with cap and no discount uses this approach to calculate the Safe Price, using the Post-money Valuation Cap in place of the pre-money valuation in Equation (3). However, this still leaves it unresolved how the equity round price is calculated.)

At this point, the SAFE investor, thinking they had signed up for issuance in an equity round determined using equation (1) may well object, since they may consider themselves to be diluted by the new investor receiving more shares at a lower price. It is not clear whether the SAFE investor has any negotiation rights

in this matter: this question is another point of incompleteness in the contract. On the one hand, the Equity Financing clause states a specific rule for conversion of the SAFE, and the contract does not explicitly give rights to negotiate its interpretation. On the other, the fact that part (i) of this clause requires the SAFE investor to execute certain documents at the time of the equity round potentially gives them some leverage to disrupt the round by withholding their signature. In practice, at least in the context of investor communities such as Silicon Valley, there is a social incentive for the parties to reach a compromise agreement: failure to come to agreement, or enforcement of unfair terms, on a particular deal may tarnish the reputation of the investor, with consequences for their participation in future deals. A SAFE, in practice, may provide lesser protection for investors who are foreign to such an investor community.[5]

A compromise that is apparently frequently reached is called the "Dollars Invested Method" of computing a price, and involves using the equation

$$\begin{aligned} \text{post-money valuation} = \ & \text{pre-money valuation} + \\ & \text{new investor's money} + \\ & \text{SAFE principal} \end{aligned} \tag{4}$$

instead of equation (2). (In effect, this treats the SAFE investors' principal as if it had been invested at the time of the equity round, rather than earlier.)

We refer to [vdMM21] for a detailed analysis of these issues for Pre-Money and Post-Money SAFEs with cap and no discount. One might take the view that the incompleteness identified above constitutes a deficiency in the natural language contracts that should be corrected when developing corresponding smart contracts. However, there is evidence that the incompleteness is deliberate. Feld [Fel15] writes

> Most notes are ambiguous as to whether they convert on a pre-money or a post-money basis. This can be especially confusing, and ambiguous, when there are multiple price caps. There are also some law firms whose standard documents are purposefully ambiguous to give the entrepreneur theoretical negotiating flexibility in the first priced round.

This point is reinforced by a change in definition of equity financing by Y Combinator. Under a Pre-money SAFE, in equity financing the company sells shares "at a fixed pre-money valuation" while, under a Post-money SAFE, the shares are sold "at a fixed valuation, including but not limited to, a pre-money or post-money valuation". For our present purposes, the main point is that SAFE contracts, as they are treated in practice, may not be deterministic with respect to relationship between pre-money valuation and price, and consequently may also leave open precisely how to calculate the number of SAFE shares to be issued, if only a pre-money valuation is given.

[5] Thanks to Alex Cook, Freehills, for helpful discussion on the question of SAFE investor negotiation rights.

4.4 Open Structure

Another of the difficulties that needs to be addressed in formalizing the SAFE contract is the *open structure* of the context in which it applies. A key aspect of this context is that the Company has a *capitalization table*, listing shareholders and the number and types of shares each holds. A SAFE note itself has a particular structure, but it is not self-contained. Instead, it creates new rights and obligations with respect to the Company's future issuance of shares and, thus, the future state of the capitalization table.

One approach that the smart contract programmer might take to accommodate this is to include, in a smart contract representing the company's financial structure, method calls that correspond to the issuance of a SAFE note. However, this is insufficient: the Company may issue a SAFE note to an investor, but it could equally enter into any number of other contracts, each with a completely different structure. Already, SAFE notes come in multiple forms: the four original "Pre-Money" Y Combinator SAFEs, the four new "Post-Money" versions. Other firms have developed their own variants on the same theme. Further variations may arise from a negotiation between the Company and the early investors. There is an infinite variety of potential contracts, so it is not feasible for each possibility to be anticipated and enabled by including a set of functions specific to the structure and logic of the contract.

A related point is that the offer of the new investor in the term sheet that triggers the price calculation may be stated in a number of different ways, e.g., "$5M for a 30% share of the company post-money" or "$5M at a pre-money valuation of $10M". Another type of openness in what we need to deal with is the governance structure of the company, which involves over time. In particular, we expect that one of the consequences of an equity round will be that a new governance structure is instituted - it is typical for the lead investors to take board positions. However, the details of this structure will be one of the points of negotiation in the equity round, and cannot be predicted ahead of time. A smart contract representation of the company therefore needs to allow flexibility in the representation of the governance rules.

4.5 "Nuncospectivity"

Conventional contracts are generally enforced retrospectively – after events related to the contract have occurred. Occasionally, when awarding injunctions, for example, enforcement is addressed prospectively, before the events occur.

However, smart contracts can provide enforcement only (or mainly) at the time of the event – a perspective we might call *nuncospect*[6]. To see the problems this presents, we return to the definition of Equity Financing discussed in Section 4.2.

The equity financing definition refers to a "series of transactions". At the time of the first transaction (or, indeed, any other transaction) it is impossible to know whether there will be a later transaction that should be interpreted as part

[6]From the Latin *nunc* meaning now, and *spectare* meaning to look.

of this series. With hindsight, it is possible to identify transactions that have similarities in time of execution or other characteristics that suggest that the transactions should be considered as comprising a series of transactions, rather than individual transactions. Similarly, the "bona fides" and "the principal purpose" of a transaction can become clearer in retrospect, when there are other events to create a context. Such events can appear irrelevant to the smart contract, for example, transfer of money between two accounts not appearing in the contract.

In the immediately following sections, we will focus on what smart contracts can enforce nuncospectively. We return to the issue of retrospective interpretation in Section 9.3.

5 Strategies for Formalization

While the difficulties discussed in Section 4 present challenges to the formalization of SAFE contracts as smart contracts, they do not prevent smart contracts providing useful levels of security for this application. One can take a number of attitudes to these difficulties. One is that indefiniteness in the text is a weakness in the design of the contract, and insist that only well-designed contracts that are free of indefiniteness should be represented as smart contracts. This leads one to first refine the contract to eliminate ambiguity and incompleteness. Alternately, one can take the attitude that the contract's indefiniteness is intentional, to allow the parties to the contract room for negotiation, or to accommodate unforeseen circumstances.

Depending on the attitude taken, the blockchain can be used to give integrity to the cap-table data and assure terms of the SAFEs to various degrees. At one end of the spectrum, we have automated enforcement of highly precise terms. At the other, the blockchain enforces the use of processes whereby human agents resolve the indefiniteness. As multiple human agents are typically involved in a contract and its operation, the design of such processes requires design decisions to be made about the powers that each of these agents should have with respect to the resolution of indefiniteness.

In what follows, we develop an approach that encompasses both these attitudes to the question of how far to formalize the terms of a natural language SAFE contract, and makes it possible to cover a range of points on this spectrum, depending on how much formalization is undertaken.

5.1 Limitation of Fact Patterns

We note that, when developing a smart contract representation, where the attitude adopted is that open texture and incompleteness in a natural language contract should be replaced by precision, it is not necessarily required to resolve all possible questions of interpretation that could arise. The issues of open texture discussed in Section 4.1 bear on an "interpretative perspective", in which, given a set of facts, we seek to determine whether a concept applies. However,

smart contracts are not necessarily required to deal with arbitrary fact patterns. This is because smart contracts themselves may impose a restricted model of the factual ground for concepts than one might find in the real world. The code can limit the possible fact patterns that can be constructed, and the developer has the option to design the code so as to avoid fact patterns that raise difficult questions of interpretation. It suffices that the parties to the smart contract are prepared to accept its limited flexibility, and that they are in agreement that the code correctly captures the concepts of concern within this limited range. (As in the methodology of Riccardian contracts [Gri04], discussed in Section 10.4, this agreement to accept the code's interpretation of the legal concepts may itself be signed by the parties as a legal contract.)

As one concrete example of this, consider the question of which transactions, in particular, which issuances of Preferred shares, should be considered to be part of an Equity Financing[7]. If we design the smart contracts so that arbitrary patterns of share issuance are supported, then we need to address this question of interpretation. However, by appropriate choice of operations, and constraints on when these operations are enabled, we can ensure that this question does not arise.

Suppose that in addition to the basic operation of issuing shares, we design the system so that there is a compound operation of conducting an equity round, with the effect of issuing shares, amongst other things. Further, suppose that whenever a SAFE contract has been issued but remains unconverted, we constrain the smart contract for the company so that when shares are issued outside of the context of this equity round operation, the type of the shares may not be Preferred shares. That is, the company may issue Common stock, but not Preferred stock, except by use of the equity round operation. Then, in the fact patterns that arise from the smart contract, every issuance of Preferred shares, that is of relevance to the terms of the SAFE, is necessarily part of an equity round. We therefore avoid the difficulty of interpreting which issuances of Preferred shares should count as part of an Equity Financing: they are explicitly marked as being so through having been generated by the equity round operation.

5.2 Treatment of Indefinite Terms

As already noted, when dealing with indefinite terms and incompleteness in a contract, there is a spectrum of approaches that can be used to determine whether a term applies in a given situation.

At one end of the spectrum is full automation. This requires that the conditions under which the term applies be exactly expressed in code, and that any incompleteness in decision making be resolved in the positive or negative. Limitation of fact patterns may help with this, by ruling difficult cases out of scope. Another factor is that in formalizing a contract, what matters is that the parties to the contract can *agree* upon the formalization for the limited set

[7] Recall that Equity Financing is defined as a sale of Preferred shares

of events the contract will cover. By contrast, a formalization of legislative text has impact on a much larger community and set of events. Nevertheless, a contract may refer to legislation. The SAFE, for example, refers to situations in which

> any "person" or "group" (within the meaning of Section 13(d) and 14(d) of the Securities Exchange Act of 1934, as amended), becomes the "beneficial owner" (as defined in Rule 13d-3 under the Securities Exchange Act of 1934, as amended)

in its definition of "Change of Control" (which is a trigger for Dissolution events). In any formalization, the possibility that code will make errors in unforeseen situations remains. (We discuss how such errors may be dealt with in Section 11.)

At the other end of the spectrum, we can leave the decision about whether a particular condition applies to be made by human agents. An issue then is *which* human agents should have the power to make this decision. Alternatives include:

- One party to the contract is trusted to assert satisfaction of the condition.
- All parties to the contract need to agree on the satisfaction of the condition.
- A trusted third party asserts that the condition holds. Attestation of facts by third parties called "oracles" is a common pattern in smart contracts. Oracles are typically used to provide information such as exchange rates that comes from off-chain sources, but one can envisage services that manually evaluate whether a natural language legal expression applies to a situation, and issue cryptographic certificates that can be processed by the smart contract.
- All parties need to agree, but in case of disagreement, a third party arbitrates the dispute.

In practice, the choice of approach may depend on which parties to the contract are affected by the particular condition, as well as the degree of trust between the parties. In the worst case, there is always ex-post recourse to the courts (we discuss this in Section 10.5).

As noted in Section 4.3, one of the forms of incompleteness in SAFE contracts, concerns the relationship between pre-money valuation and equity round price. One full automaton approach to dealing with the incompleteness would select a specific formula for computing price from pre-money valuation and hard-coding this formula into the smart contract. This would amount to the SAFE investor and the company agreeing on the conversion approach at the time the SAFE contract is issued.

An alternative, which preserves the incompleteness in the SAFE, is to admit that SAFE investors have negotiation rights in the equity round. This

leads to implementations in which the company submits on-chain a proposal for the structure of the equity round, and both SAFE investors and equity round investors are given the opportunity, on-chain, to provide their consent to the proposal, to enable it to proceed. Alternately, by withholding consent, an investor can block the proposal from progressing, incentivizing the parties to negotiate a revised proposal. In this approach, the proposed issuance of shares to a SAFE investor could be any number, open to negotiation by the human agents, so it is not necessary for the full details of the Equity Financing clause to be represented in smart contract code: this can be left in natural language form.

Advantages of this approach are that it leaves open-textured concepts to be dealt with by human intelligence, and gives significant flexibility with respect to choice of resolution of the incompleteness in the contracts. Moreover, the chain-code is easier to implement and verify, since it is is required only to code the proposal and acceptance process, rather than the potentially intricate structure of the SAFE contracts.

On the other hand, this approach provides no assurance of the correctness of calculations used to determine the share issuance in an equity round, so there is a higher risk of errors. It is also the case that a SAFE investor is given the power to hold up an equity round simply by withholding their consent, which is against the spirit of "automatic conversion" in the SAFE contracts, and the fact that these contracts do not explicitly grant the SAFE investors powers to influence the form of the equity round.[8]

5.3 Propose-and-Verify Implementations

On a "full-automation" approach to formalizing the Equity Financing clause, one approach would be to have the number of shares to be issued in conversion of the SAFE be computed by a smart contract representing the SAFE, given appropriate inputs. For example, in the case of the Pre-Money SAFE, given the pre-money valuation and price of the equity round, the number of shares to which the SAFE is converted, is expressed in the Equity Financing clause as a simple rule that can be easily coded. In the case of Post-Money SAFEs, things are less straightforward, however: here the Equity Financing clause states a constraint in the form of an equation based on the issuances in the equity round. The combined SAFEs therefore generate a set of simultaneous equations that need to be solved in order to determine the correct issuance to each SAFE investor. Conditions for solvability of such simultaneous equations

[8] More complex versions of this approach might address this problem by allowing an equity round to proceed with only the consenting SAFEs being converted, leaving the others to be resolved through off-chain legal processes. A problem with this is that the resulting cap table state will be "incomplete", with some SAFEs converted, but others that should have been converted in the round not converted. This complicates performance of subsequent actions, e.g., distribution of dividends, and/or execution of subsequent rounds for which the holder of the unexecuted SAFE would have had pro-rata rights. (On the other hand, one might take the view that the blockchain then correctly represents the legal situation in which the company finds itself, with unexecuted obligations subject to dispute.)

may be complex, particularly if the company has issued multiple convertible instruments of different types. For constraints written in sufficiently expressive forms, the solution problem is, indeed, likely to be intractable, making the code for computing the SAFE issuance overly complex for a resource (e.g., gas cost) constrained blockchain setting. In full generality, with convertible instrument terms written in arbitrary code, the solution problem will be undecidable, and there may be no program that always computes a unique solution, even if one exists. Even where the problem is decidable, formally verifying the correctness of code that generates and solves equation systems may be quite complex.

An alternative to computing the correct issuance on-chain is to require that the company propose a concrete number of shares to be issued to each convertible instrument investor as well as each equity round investor, and to have the smart contract code representing these investors *verify* that the proposal complies with the terms of the convertible instruments. For contracts like the Post-Money SAFEs, which state these terms as equations, this means that the code need only verify that the proposed equity round solves the equations, rather than compute a solution. This is both computationally more tractable, and easier to code. Moreover, it means that the code of the smart contract more closely resembles the natural language form of the legal SAFE, which simplifies the job of assuring the SAFE investor that the smart contracts will perform as expected. Formal verification of code that merely verifies correctness of the number of shares issued, rather than computes it, is likely to be significantly simpler.

A further advantage of an approach based on verification rather than computation is that it is potentially more tolerant of indeterminacy in the original natural language SAFE contract, should that be considered desirable. A computation approach would require that indeterminacy be eliminated in order to yield a deterministic result. On the other hand, if we treat the SAFE as stating a set of constraints on the equity round, we can tolerate the existence of multiple solutions.

5.4 Price Computations

As noted in Section 4.3, the fact that a company has issued SAFE contracts means that a sophisticated equity round investor will want a reduced share price in order to ensure that the value of their shares is not immediately diluted by the SAFE issuance. In general, price is a matter for negotiation between the company (which would prefer a higher valuation and price) and the investor (who would prefer the opposite), but equation (3) provides a rational basis on which a price can be computed from a valuation. Since terms for SAFE share issuance in this equation are given by potentially complex conditions, which may circularly depend on the share price and/or valuation, we again have a problem in the form of a set of equations to be solved.

It would be desirable to provide computational support for this problem. However, both the requirement to be able to determine the price by negotiation and the same computational complexity considerations as in Section 5.3 imply

that it is preferable to leave this support to off-chain computations. When there are multiple SAFEs with different conversion conditions, and we use the "Propose-and-Verify" approach, it would similarly be desirable to have off-chain computational support for constructing equity round proposals that will be be accepted by the on-chain verification code. As our focus in this paper is on smart contract aspects of systems supporting SAFE contracts, we do not attempt to address this problem in this paper.

6 Design Patterns

A number of general design patterns prove to be useful in the development of chaincode representations of SAFE contracts. In this section we give a general introduction to two of these patterns.

6.1 Atomic Swap

Suppose that two parties A and B have agreed to exchange digital assets a and b, respectively. If the transfers are made sequentially (e.g., first A transfers a to B and B transfers b to A), each party faces the risk that the other party will not abide by their end of the bargain (e.g., B could receive a but then refuse to transfer b to A, leaving A with a loss). Smart contracts provide a method to overcome this difficulty, in the form of *atomic swap* transactions [Her18, vdM19].

Atomic swaps can involve multiple blockchains, but for our purposes in this paper, swaps on a single chain will suffice. These can be effected by chaincode having the following structure, in the case of a swap of asset a held by party A for an asset b held by party B:

SwapContract(A, a, B, b):
Deposit(A, a): A transfers a to this contract
Withdraw(A, a): A recovers a from this contract
Deposit(B, b): B transfers b to this contract
Withdraw(B, b): B recovers b from this contract
Swap: if this contract holds both a and b then
transfer b to A and transfer a to B

We omit detailed code, giving just the interface of the smart contract and a sketch of its functionality. Each line describes an operation performable in the contract. There are operations *Deposit* whereby the parties can transfer their assets to the control of the smart contract, so that the assets come under the smart contract's control. Once both assets have been transferred, the operation *Swap* transfers each asset back to the opposite party. In case the other party fails to perform their *Deposit* action in a timely fashion, the action *Withdraw* can be used to recover an asset that has already been deposited, so that the swap then fails. The effect is that the swap happens atomically, either taking

full effect, or leaving both parties with effective control over their original asset (i.e., either holding it, or able to retrieve it from the swap contract).[9]

Usually such swaps are asset for asset (e.g., one form of cryptocurrency for another), but richer forms of swap smart contract can be constructed (e.g., a multi-party contribution of cryptocurrency or approval in exchange for a change of state in chaincode). We use such a richer form of swap below to secure aspects of equity rounds.

6.2 Controller

Code expressed in an object-oriented programming language consists of objects with a number of variables, and operations that act on these variables. Frequently, these operations need to be subjected to access controls, so that they may be performed only by agents authorized according to some policy.

A common example of this is a situation where a unique agent may perform some operations. In Solidity code, this is captured by the *owner* pattern (see, e.g., [Opec]). To distinguish this kind of ownership for purposes of access control from the notion of ownership of shares in a company, we will henceforth use the term *controller* rather than "owner" for the former. The pattern is implemented by including in the smart contract a variable `controller` of type `address` that stores the address of the controller. Each operation that may be performed only by the controller then contains a check that the address from which the operation is called is the address of the controller. In Solidity, this may be done concisely by first defining a modifier as follows:

```
modifier OnlyController {
       require(
           msg.sender == controller,
           "Only controller can call this function."
       );
       _;
 }
```

This modifier can then be applied in the declaration of each operation that may be performed only by the controller, with the effect that the check is performed at the start of the operation. For example,

```
function set_controller (address newcontroller) public OnlyController {
  controller = newcontroller ;
}
```

restricts the function `set_controller` that changes the controller, so that it may be performed only by the current controller.

Typically, the value of the `controller` variable is the address of the public key of a human agent. However, we note that once a smart contract has been

[9] See [vdM19] for an analysis of the powers of the agents in such contracts in terms of strategy logic operators.

created with the controller pattern, it is possible to impose richer forms of access control by taking the value of `controller` to be the address of a *controller smart contract* that encodes the logic of the richer form of access control. For example, to impose a 2/3 majority vote as the access control rule on the operation f uniquely performable on a smart contract C by its controller, we can create a new smart contract V that first checks that a majority of the three voters agree to perform operation f on C, and if so, makes the call $C.f$ as agent V. Once we set C.`controller` to be equal to the address of V, the smart contract V will be the only agent that is able to perform $C.f$, and it will only do so in case of a 2/3 majority approval.

Similarly, suppose we wish to restrict the powers of a (human or smart contract) controller γ of a smart contract C, so that γ is permitted to perform certain operations f on C if and only if property ϕ_f holds of C. To do this, we may create a controller smart contract R that also contains an operation named f for each operation f on C to be so constrained. The code for $R.f$ first checks that condition ϕ_f holds of C (this requires calls to *view* functions of C, which do not change the state of C) and then calls $C.f$ in case the condition ϕ_f is satisfied, or aborts otherwise. The contract R is defined to have its own controller. We then arrange to have γ set the controller of C to be the address of R, and set the controller of R to be γ. This has the effect that still only γ may perform operations on C, but it needs to do so via the contract R, and its powers to do so are restricted by condition ϕ_f for each operation f.

In settings where changes of access control policy are necessary, it is desirable to include the function `set_controller` amongst the functions so controlled. (The function to change the controller of R needs to be given a different name to avoid a clash with the mirrored `set_controller` that calls C.`set_controller`.)

We use this *constrained controller* pattern in the chaincode below.

7 Architecture and Implementation

Factors, noted above, that influence the design of our smart contract solution for SAFE contracts, are the *open structure* of the situation we are modelling, and the issue that Post-Money SAFE contracts are stated as a *constraint* on the promised shares rather by a method to compute their number. To address that there are multiple forms of SAFE contract, we have abstracted these to an abstract contract (instantiated by subclassing) that places constraints on the Company and the way it manages an Equity round. This abstraction also enables us to handle a variety of policies concerning the rights that the SAFE holder has with respect to approval of the equity round, by encoding these rights directly into the code of the smart contract representing the SAFE. To handle the openness and dynamic nature of the governance structures of the company over time, we use the constrained controller pattern.

To address the fact that Post-money SAFEs state but do not solve a constraint system on the equity round, rather than have the smart contract *calculate* the number of shares to be issued to SAFE investor, we have the company

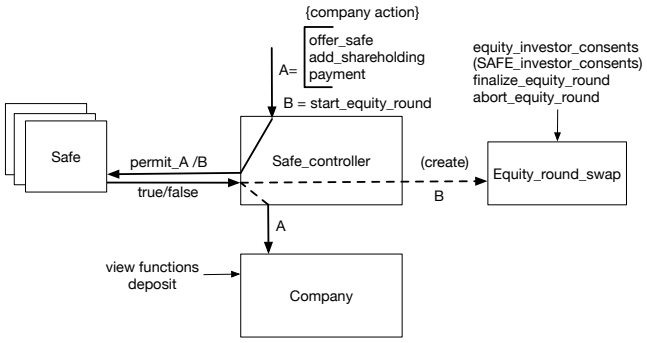

Figure 1: Architecture with SAFE_controller

propose the structure of an equity round, and have this proposal *verified* for correctness by the smart contract. This also enables the SAFE contract abstraction to have instances that are both nondeterministic (a way to handle incompleteness in the legal text) and arbitrarily expressive in its constraints without requiring costly on-chain constraint solving. Both executing an equity round and the offer-acceptance process of a SAFE are "long-running" and composed of multiple transactions. We treat them as *compound* events that are serialized with other events using a finite state-machine.

The architecture of our solution is dynamic, with its principal configurations shown in Figure 1 and Figure 2 (on page 40). Figure 3 (on page 41) depicts the finite state-machine view of some of the components, that underpins some key correctness properties.

In overview, a `Company` smart contract represents the financial structure of a company, in particular, its share structure and financial contracts issued. We use smart contracts of type `Safe` to represent the constraints that the SAFE contracts issued by a company place on the actions `A` that the company may perform. Actual enforcement of these constraints is achieved by setting the controller of the `Company` smart contract to be a `Safe_controller` smart contract. The `Safe_controller` mediates all access to the `Company` actions while SAFE contracts are in force, as depicted in Figure 1. To provide assurance to the investors that the equity round will be conducted fairly, the atomic swap pattern is used in the form of a `Equity_round_swap` smart contract, with the left diagram in Figure 2 indicating the form of the architecture while an equity round is active. If the equity round fails, the architecture reverts to that of Figure 1. To handle the post-round authorization regime, the design allows a new controller to be installed after the equity round completes.

Each of these smart contracts is discussed in greater detail in the remainder of this section. The more detailed discussion in the following subsections is based on a Solidity code implementation.

7.1 Company Financial Structure

We first lay out the context in which SAFE instruments operate: the financial structure of a company. From a financial point of view, the structure of a company consists of information including (1) its accounting journals (henceforth, we use the term "books"), and (2) its capitalization table (henceforth, we use the common abbreviation "cap table"). The books record information such as assets, liabilities and transactions. The cap table records the state of shareholder equity in the company, i.e., the number of shares owned by each shareholder.

Instruments such as convertible bonds have a mixed nature: they may have properties similar to a loan (which would be a liability, hence recorded in the books), but may also constitute an obligation to issue shares (which impacts the shareholder structure, so should be recorded in the cap table). Part of the design intent of the Y Combinator SAFE, which involves no interest payments, and therefore lacks loan-like features, was that it should be represented in the cap-table. On the other hand, other parties, e.g., tax officers, or investors valuing the company, may take a different view. Accounting standards contain detailed rules governing whether, depending on its precise construction, an instrument should be represented in the books or in the cap table for purposes of compliance with reporting obligations [Del19]. In some cases, the instrument needs to be decomposed into parts represented in the books and parts represented on the cap table. For purposes of the present paper, we take no stance on this issue, and simply represent the convertible instruments independently of both the books and the cap table, leaving accounting status as an independent consideration.

Since our primary focus in this paper is the representation of SAFE contracts and their effects on the cap table over time, we take a highly simplified view of the company books: we assume that the only assets or liabilities recorded on-chain are some of the company's holdings of the cryptocurrency native to the blockchain. (The code has no dependency on the books, but we explicitly model payments to and from the company, including payments of principal for SAFEs and shares purchased.)

Company transactions such as issuance of SAFE instruments and equity rounds involve the transfer of money to the company. Even in a possible future where it is common for investors to hold cryptocurrency that they wish to invest, it may well remain the case that most investments are made from a larger base of fiat currency holdings. Moreover, a company that receives cryptocurrency investments is likely to need to transfer the money received to parties who prefer payment in fiat. A full treatment of SAFE instruments registered and secured on blockchain should therefore accommodate fiat payments in some way. We do not attempt to cover this problem in the present paper, and assume that all money transfers are made in the native cryptocurrency of the blockchain. This assumption will enable us to determine the full extent of the security benefits that could be obtained, in principle, from a blockchain and smart contract treatment of SAFE instruments. Inclusion of fiat payments, except in the form of fiat represented on-chain by a "stablecoin" or Central Bank Digital Currency,

is likely to weaken the security guarantees that can be provided. We leave the question of what would be lost in adding a coverage of fiat payments for treatment elsewhere.

Based on the above assumptions, our on-chain representation of the state of a company will be a smart contract of class `Company`, that

- may hold cryptocurrency, representing some of the assets of the company, and
- represents the cap table of the company, in the form of a record of shares held by each investor in the company,
- represents the convertible instruments issued by the company (assumed to be SAFE contracts only).

Concrete instantiations may refine this abstract representations in various ways. In the remainder of this section we develop an abstract description of this smart contract and discuss various design questions that arise.

With respect to the record of shareholdings in the company, a first issue is to represent possible types of shares in the company. The text of the SAFE contract mentions several share types: Common Stock (held by the founders), Preferred Shares (typically held by equity round investors), and SAFE Preferred Shares (issued to the SAFE investors in conversion of the SAFE). Each share type corresponds to a bundle of rights relating to company governance, voting, dividends, participation in future equity rounds, corporate transactions such as sale of the company, and preference order and/or amounts for distribution of money in the event of a company liquidation. After an equity round, founder and employee stock is generally subject to vesting schedules, so effectively differs from other Common Stock. In general, it is open to a company to define other share types.

The most general treatment of the record of shareholdings should therefore allow for flexibility in the definition of the type representing shares. Technically, this could be done by representing shares as an abstract object interface that can be subclassed to define different share types. Since our main focus is on SAFE contracts and their effects on the *number* of shares issued, we do not attempt to develop such a detailed representation in the present paper. Instead, we assume that the three share types mentioned in the SAFE contract are the only possible types of shares. This allows us to represent the general idea of a share as an enumerated type[10]

```
enum ShareType={Common,Preferred,Safe_Preferred}
```

[10]One disadvantage of this representation is that the differences in the rights associated with each of these share types are not directly bundled with their definition. Instead, the code of company operations will need to include cases for each of the share types. We leave the development of alternative representations that address this issue for treatment elsewhere, but note that an architectural structure for a solution can be seen from our presentation of SAFE smart contracts below, using a controller to restrict the company to comply with the terms of the SAFE contracts it has issued.

An individual shareholding can therefore be represented as a tuple

```
struct Shareholding {address  payable owner;
                     ShareType  class;
                     uint number; }
```

consisting of the owner of the shares, the share type and the number of shares held. (We treat the shareholder identity very simplistically as a pseudonymous cryptographic address. For legal purposes, further identifying information would very likely need to be represented, raising privacy questions, particularly in the setting of public blockchains. We discuss privacy issues in Appendix B.)

The record of shares held in the company can now be represented as an array of such shareholding entries, represented in the smart contract by a variable `shares` declared as[11]

```
Shareholding[] shares;
```

Similarly, the record of SAFEs issued by the company can be represented by an array of entries:

```
Safe[] safes_issued;
```

where a SAFE is represented by type `Safe`.

The company's books are represented just as a value of cryptocurrency holdings. We can generally rely upon the blockchain platform to maintain the value of cryptocurrency held in the `Company` smart contract. In the setting of Solidity, this can be accessed inside the code for the smart contract via the construct `address(this).balance`.

It should be possible to update these variables by means of various operations. In order to focus on the main issue in this paper—conversion of SAFE contracts to shares— we assume SAFEs and shares are not transferable, and that the company may not cancel shares or restructure the cap table by means of stock splits or merges. We work with a minimal set of operations on `Company`:

- `add_shareholding` to issue new shares,
- `add_SAFE` to issue a new SAFE
- `payment`, which transfers some of company's cryptocurrency holdings, to pay for goods and services
- `deposit`, used by other parties to transfer cryptocurrency to the company

However, these operations are not unrestricted: only a duly authorized officer of the company should be able to issue new shares or SAFEs, and there are circumstances in which the company will be constrained in performing such

[11] Additional use of a mapping in Solidity would allow for more efficient lookup, but since our main concern in this paper is mainly to understand the logic of the situation and the ways that smart contracts support enforcement of security properties, we do not attempt to develop code that is optimal on measures such as time, space, or gas usage.

actions. For example, equity round term sheets (preliminary contracts setting out the expected conditions of an equity round as a basis for more detailed negotiations) may constrain the company from changing its capital structure until closing or abandonment of the equity round. After an equity round, the governance structure of the company is likely to have changed, for example, by instituting a board with membership from the new investors, and issuance of new stock will typically require approval of the board and/or a majority vote of shareholders. Further, as discussed in Section 4.2 and Section 5.1, to avoid questions of interpretation such as whether an issuance of Preferred shares implies the occurrence of an Equity Financing event, we would like to limit the possible fact patterns so that Preferred shares can only be issued in the context of an explicit Equity Financing event. Although not in the scope of our limited model, restrictions on rights to transfer stock may also apply, since transfers amongst investors may change voting power. Share transfers may also be subject to rules imposed by the jurisdiction of incorporation [DLA].

To capture such constraints, we therefore require a means of access control on the performance of actions on the cap table. This can be done in an abstract way using the controller pattern from Section 6.2. We take the original controller to be the agent (address) creating the `Company` smart contract, and include a function `set_controller` whereby the controller (and only the controller) may transfer control to another agent (possibly a smart contract). After an equity round, the new controller could be the (address of the) director of the board, or a smart contract that implements the desired governance structure and rules. For technical reasons, an additional controller value is used in our code at intermediate stages of an equity round. (See Section 7.4.)

7.2 Safe Contract Representation

We next present a representation of the terms of a SAFE contract using a smart contract. The representation captures constraints that a SAFE places on the company's actions.

SAFEs come in multiple forms: Pre-Money SAFEs and Post-Money SAFEs, each of which may include a cap and/or a discount. We develop a representation that can accommodate all of these types of SAFE. Our approach allows a company to issue a set of SAFE contracts of variant types. We believe the general approach can be extended to handle other types of convertible instrument, including interest bearing convertible bonds, KISS (Keep It Simple Securities), [Rai14], etc.

We achieve generality by defining an abstract smart contract class `Safe`, which may be subclassed to instantiate specific types of SAFE. (Implementations of the subclasses for the "PreMoney SAFE with cap but no discount" and the "PostMoney SAFE with cap but no discount" are given in the Appendix.) The main variables of this abstract class are

```
uint public principal;
address payable public owner;
```

```
Company public company;
uint public deadline;
Safe_status public status;
```

representing, respectively, the amount paid for the SAFE (in Wei), the party holding the SAFE, the smart contract for the company being invested in, a deadline by which the offer of the SAFE needs to be accepted by the investor, and the status of the SAFE, a value from the enumerated type[12]

```
enum Safe_status = {Offered, Active, Terminated_withdrawn,
                    Terminated_converted}
```

A `Safe` is offered to the investor in state `Offered` and becomes `Active` once accepted with payment of principal. A deadline is given for acceptance. (The offer and acceptance process is described below in Section 7.3.)

Specific subclasses of `Safe`, such as `Safe_PreMoney_Cap_NoDiscount` would add variables such as

```
uint public cap;
```

to represent the cap used in converting this particular type of SAFE to shares in the context of an equity round. The set of such parameters of a SAFE vary amongst the different types of SAFE, so each subclass defines its own constructor to allow the values of these variables to be set at the time of creating a new `Safe` instance.

Once a company has issued a SAFE, it needs to comply with restrictions imposed by the SAFE. In the `Safe` smart contract itself, we capture the logic of these constraints, but do not actually enforce them. Enforcement will be done using the controller pattern, by setting a controller for the company that queries the `Safe` contracts to determine which actions on the company are consistent with all the SAFE contracts that the company has issued. We discuss the `Safe_controller` contract used for this purpose in more detail in Section 7.3. For now, it suffices to note that the `Safe` smart contract has a variable declared as

```
Safe_controller public safe_controller;
```

that is used to record the `Safe_controller` that is used to enforce the SAFE constraints. Functions `accept_safe_offer` and `withdraw_safe_offer` on the `Safe` are called via the `safe_controller` to update the status of the `Safe`, as described below.

The logic of constraints that the SAFE places on the company is captured by functions in the `Safe` smart contract named `permit_A`, where `A` is an action (function) on the `Company`. The parameters of `permit_A` and `A` are the same.

[12]We focus in this paper on just the issue of conversion in an equity round, and `terminated_converted` represents whether or not the SAFE has been converted. A more refined formulation using an enumerated type expressing other types of termination would be appropriate in a more complete development.

The function `permit_A` may query the state of the contracts at `company` and `safe_controller` and returns a boolean value, which expresses whether the company is permitted to perform action `A` in that state. These functions will in fact be called only when no equity round is active, and no SAFE is presently on offer, because of the serialization policy described below, so we focus on that case.

These functions are `virtual` in `Safe` and are given concrete definitions in the subclasses for specific types of SAFE. For example, in our implementations, function

```
permit_add_shareholding(address payable, //owner of shares,
         ShareType, // type of shares
         uint // number of shares
         )  public virtual view returns(bool)
```

returns `true` when it is consistent with the constraints placed on the company by the SAFE for the company to issue the given number of shares of the given class to an investor.[13]

When no equity round is active and the share class is `Common`, this returns `true`, but it returns `false` if the share class is `Preferred` or `Safe_Preferred`. (We apply here the simplifying assumption of Section 5.1, requiring that all issuances of `Preferred` or `Safe_Preferred` shares must be part of a properly constructed equity round.)

This leaves the question of what this function should return in case there is an active equity round. An apparent difficulty is that while the share issuance associated to the equity round should be permitted, all other requests to issue shares should be rejected, as part of the *stand-still* provisions for the equity round. (These provisions aim to keep the financial state of the company stable during the round in order to assure the equity investors that the financial position of the company will not change materially while the round is in progress. Were this to occur, it would impact the value of the shares they are acquiring.) A further reason to reject such extra requests to add shares during the course of the equity round is that, conceptually, they should be sequenced after the equity round has completed, at which point the company would typically be under a different control regime, which might not approve the transaction.

Prima facie, this requires us to be able to distinguish requests to add shares that are part of the equity round from others. However, we can avoid doing so by taking the following view of the events in which shares are added in execution of an equity round: rather than being treated as independent events requiring individual approval, these are are part of a compound "equity round" event, which is authorized as a whole. (See the discussion of `permit_equity_round`

[13]The Solidity keyword `virtual` indicates that this declaration is describing only the interface of the function, and that actual code for the function needs to be provided in the concrete subclasses of the abstract class `Safe`. The `permit_A` are the only virtual functions of the class `Safe`. Parameter variable names have been elided and replaced by comments in the declaration because the function is virtual and therefore lacks a body that references the variables.

below.) Thus, `add_shareholding` events that are part of the equity round are not referred to the `Safe` for approval via a call to `permit_add_shareholding`. This understanding makes it correct for `permit_add_shareholding` to simply return `false` while an equity round is active.

Similarly, function

```
permit_add_SAFE(uint  // principal,
                uint // cap,
                address payable ) // owner
```

returns `true` if the state of the company is such that a new SAFE may be issued. In general, a SAFE does not prohibit addition of other SAFE's when an equity round is not active.

In the case of function

```
permit_payment(uint // amount,
               address payable ) // recipient
```

the result is generally `true`, since a SAFE generally does not constrain the company's rights to spend its funds.

Since `set_controller` is one of the operations of `Company`, we similarly should have an operation

```
permit_set_controller(address payable)  // new controller
```

In general, a SAFE does not constrain changes of the agent (e.g., CEO or a delegate) who may perform company actions.[14] However, while a `Safe` is active, the controller of the `Company` smart contract should be a `Safe_controller`, in order to automatically enforce the rights of the SAFE investors. The agent who may perform company actions is therefore not the controller of smart contract `Company`, but of the smart contract `Safe_controller`. The function `permit_set_controller` is therefore defined in our implementation to always return `false`. (Indeed, the `SAFE_controller` masks the `Company` action `set_controller` to ensure that it is not inappropriately removed as controller.) Note that changes to management of the company can still be made by use of the function `set_Safe_controller_controller` on the `Safe_controller`, as discussed below.

The above covers the functions `A` of `Company` that need to be placed under control using functions `permit_A`. The SAFE contract introduces the additional idea of an equity round and, as discussed above, we view this as a compound event that requires approval by the `Safe`. In particular, approval of the equity round is obtained by calling a function

[14] We are talking about "controller" in the sense of the management of the company and its smart contract realization here, not the effective control of the company resulting from a majority shareholding. A SAFE's Liquidity Event and Dissolution Event clauses, which we have left out of scope of the present paper, are sensitive to changes to the financial structure of the company that constitute changes to the effective control of the company via voting rights, or disposal of its remaining assets. How this interacts with the notion of controller in the present work is left for future exploration, but we conjecture that it can be handled orthogonally.

```
function permit_equity_round(
    uint,   // price of each share,
    uint,   // pre-money valuation,
    Equity_round_investment[] memory,
             // equity round investor investments,
    uint[] memory,
            // no. of shares to be issued to SAFE investors
    ShareType[] memory,
            // type of shares to be issued to SAFE investors
    address payable,
           // controller after equity round succeeds
    uint,  // deadline for completion of round
    uint) // index of this Safe in safes_issued
```

The parameters of the function include a detailed description of the structure of a proposed equity round: the price[15] at which shares are being sold to the new investors, the pre-money valuation on which that price is based, the number and type of shares to be issued to each of the new investors, as well as the number and type of shares to be issued to each of the existing SAFE investors (as recorded in the variable `safes_issued` in the `Company` contract) in conversion of the SAFE. Type `Equity_round_investment` is a struct that captures an investor, the principal to be paid, and the class of shares to be issued to the investor.

To support automated enforcement of the SAFE holder's interest, the implementation in each subclass should return `true` just in case the proposal complies with the terms of the concrete SAFE. In particular, the number and type of shares that the inputs propose to issue to the holder of the SAFE represented by the `Safe` smart contract to which the call is being made, should be as described in the "Equity Financing" clause of the SAFE (or otherwise approved by the SAFE holder). A difficulty we have already noted above is that the SAFE contract text uses open-textured concepts in describing the equity round. The terms of equity round (the pre-money valuation and share price) agreed by the founders and new investor directly determine the shareholding of the SAFE investor after the equity round, but the round is required to be 'bona fide'. What rights, if any, that the SAFE investor has to object to these terms are not clearly spelled out, except to the extent that the investor has an obligation to sign the documents associated with the equity round. If withholding their signature would delay completion of the round, they may be in a position to negotiate, but even this is uncertain.

We argue in Section 9.3 that this lack of precision in fact represents a risk to the SAFE investor, and that the founders and new investor have an opportunity

[15]The text of the Post-Money SAFE (but not the Pre-Money SAFE) allows that not all new investors in the Equity round pay the same price. The SAFE is converted based on the minimum price in this event. We have made the simplifying assumption in the code that all investors pay the same price, as the alternative appears to occur only rarely. Both price and pre-money valuation are included as parameters because the Pre-Money SAFE with Cap and no Discount does not specify a formula for computing price from pre-money valuation, and several alternatives are possible, see [vdMM21] for discussion of this.

to collude in ways that are detrimental to the interests of the SAFE investor. A smart contract implementation necessarily needs to address this lack of determinism, and implement a specific policy that embodies any actions that the SAFE holder may take with respect to the equity rounds. Multiple alternative policies might be implemented, each distributing the power to decide the round structure in different ways.

- *Full automation of the equity round clause:* this policy gives no rights to the SAFE investor, and simply encodes the formula for share issuance to the SAFE investor as a function of the pre-money valuation and price into the code for the `Safe` smart contract. The code for `permit_equity_round` returns true if and only if the number of shares issued to the SAFE investor satisfies this formula. The SAFE investor is guaranteed that their issuance will be in accordance with the formula, but not that the parameters of the equity round have been reasonably constructed.

- *No automation of the equity round clause:* In this approach, the SAFE investor first interacts with their `Safe` smart contract to approve the equity round proposal, and `permit_equity_round` returns `true` if and only if the proposal has been approved in this way. This fully protects the equity round investor, but at the cost to the founders of enabling the SAFE investor to block even reasonably constructed equity rounds.

- *Partial automation of the equity round clause:* this policy is an intermediate case between the above two, that gives the SAFE investor rights to approve a variation to the number of shares determined by the formula, but no rights to block the round when the formula is satisfied. Approval is handled prior to the call to `permit_equity_round` by another function that takes as input the equity round proposal and stores the fact that the investor has approved the proposal in the `Safe` contract. When this is the case, the call to `permit_equity_round` returns true, and when not, it returns true only when the equity round proposal satisfies the formula for issuance to the SAFE investor.

For the latter two approaches, a function `approve_equity_round` with appropriate parameters would be called on the `Safe` by the SAFE investor. Richer forms of policy, such as a blanket approval of proposals satisfying certain conditions within given time periods, can also be implemented in the same way.

In addition to the parameters of function `permit_equity_round` discussed above, which are most relevant to the terms of the SAFE, some other parameters are included: the address of the controller of the `Company` smart contract once the equity round completes, a deadline for completion of the equity round, and (as a convenience for array lookups) the index of the `Safe` contract being called in the array `safes_issued` in `Company`.

The result of a call of `permit_equity_round` on a `Safe` contract relates only to that particular `Safe` contract — it is combined with results from all other issued `Safe` contracts to determine whether the round can commence. If so,

execution of the round requires a number of additional steps before the round completes. Further details on this process are provided below.

If the equity round does eventually successfully complete, it is only then that shares are issued to the SAFE investors in conversion of their SAFEs. To signal to the `Safe` smart contract that the conversion has been performed, the `Company` calls a function `convert()` in the `Safe` contract, which sets variable `status` from `Active` to `Terminated_converted`.

7.3 SAFE controller

As discussed in Section 7.2, issuance of a SAFE constrains actions that the company may take, and the logic of these constraints is captured in the functions `permit_A` in the class `Safe`, which are used to indicate whether the action `A` of `Company` is permitted in a particular state of the company. Actual enforcement of these constraints is achieved by use of the controller pattern. We introduce a new smart contract class `Safe_controller`, whose purpose is to act as the controller of a `Company`, in order to ensure that the company is compliant with all SAFEs it has issued. Before issuing a SAFE, a company should set its controller to be an instance of a `Safe_controller` smart contract and, to ensure that their interests will be protected by the code, a SAFE investor should not invest in a `Safe` unless a `Safe_controller` is in place.

The `Safe_controller` serializes the permitted company actions with the compound processes of issuing a SAFE and conducting an equity round. This is achieved by means of state variable with values in enumerated type

```
{No_event_active, Equity_round_active,
 Safe_offered,Terminated}
```

and transitions as depicted in Figure 3.[16] Functions are enabled only at states from which they label an outgoing transition.

In state `No_event_active`, the `Safe_controller` contract functions as follows: it receives requests for actions `A` to be performed[17] on the `Company`. Before transmitting these to `Company`, it first calls the functions `permit_A` of all issued SAFEs in order to determine whether performing `A` is consistent with all these SAFEs. If not, then the request is denied, else it it is forwarded to `Company` for execution.

The `Safe_controller` contract has some additional functionality that enables the company to issue a SAFE contract to an investor by means of an atomic swap process. To issue a SAFE, a `Safe` smart contract `safe` encoding its terms is first created, specifying a deadline. The function `offer_safe(safe)` is then called on the `Safe_controller`. The SAFE investor calls `accept_safe_offer` on the `Safe_controller`, attaching the required principal, in order to accept the offer. After the deadline, the offer can be withdrawn using function

[16] Additional values would need to be added to this type to cover Liquidity and Dissolution events.

[17] One exception is action `add_safe`, which is masked and replaced by `offer_safe`, which calls `permit_add_safe`.

`withdraw_safe`. (These functions call similarly the named functions on the `Safe` to effect state transitions of the `Safe`.)

In addition to primitive actions that can be performed on the company, the `Safe_controller` also supports equity rounds through functions

`start_equity_round, finalize_equity_round, abort_equity_round .`

Function `start_equity_round` has the same parameters (except for the final convenience parameter, the index of the `Safe`) as function `permit_equity_round` discussed in Section 7.2. This function can be called to initiate an equity round. All the `Safe` contracts in the company's `safes_issued` are called using `permit_equity_round`, to verify that the proposed equity round is consistent with the terms of all contracts (or has been otherwise approved by the investor, as described above). If any of these calls returns `false`, the proposed equity round is rejected. Otherwise, an `Equity_round_swap` smart contract is created.

This `Equity_round_swap` smart contract is set to be the controller of the `Company` while it is active, but returns control to the `Safe_controller` if the equity round fails. Passing control to `Equity_round_swap` both enforces serialization by preventing company actions being performed via the `Safe_controller` and enables the `Equity_round_swap` smart contract to issue shares and convert the `Safe` contracts if the round succeeds. The changes of control of `Company` dynamically modify the architecture of Figure 1, as depicted in Figure 2.

7.4 Atomic Swap Implementation of the Equity Round

An atomicity issue similar to that solved by the atomic swap of assets between two parties (see Section 6.1) arises at the time of the equity round, but involving a larger number of parties (the Company, the SAFE investors, and the equity round investors).

This is particularly the case when the SAFE investors are given the opportunity to negotiate with respect to the conversion of their SAFE contracts to shares. Suppose that the shares are issued to the new investors using a process that is completely independent of the issuance of shares to the SAFE investors. Then, from each party's point of view, there is a risk that some other party will be granted a larger number of shares than expected. The rational new investor will have calculated the price of their investment based on expectations of the number of SAFE investor shares to be issued at the time of the equity round. If more shares than this are issued to the SAFE investors, the new investor will consider themselves to have been unfairly diluted.

A similar issue arises between different new investors, even in a situation where the SAFE investors are not given negotiation rights, and even when all new investors are sold shares at the same price. Simply varying the number of shares issued may result in changes to the power balance between these investors in company governance. An aggrieved party has recourse to the courts to attempt to rectify the situation if their dispute cannot be resolved by negotiation, but this entails litigation costs. Each would like to be assured that they

will be issued the number of shares negotiated, or due from a SAFE, and that variations from the agreement will not alter their expected relative stake.

The methodology of atomic swaps can be applied to resolve this difficulty, bundling the individual share issuances into a single transaction, and ensuring that only the mutually agreed issuance of shares to each of the parties occurs in the equity round. Suppose that the parties with negotiation rights come to agreement on the number and type of shares to be issued to each of the SAFE and equity round investors. We can provide assurance to all parties that the equity round will be conducted in accordance with this agreement by codifying it using an atomic swap-like smart contract that first collects the new investment moneys, and then issues shares to all of the investors as per the agreement. In case some of the required consents or moneys are not obtained by a deadline, the contract aborts and the moneys can be recovered by the respective investors.

The `Equity_round_swap` smart contract in the implementation realizes this idea. It has local variables corresponding to the arguments of the function `start_equity_round` of the `Safe_controller`, already discussed in Section 7.3. These variables encode the proposed share issuance to each of the investors in the round, price and pre-money valuation. Additionally, the contract has variables `safe_controller` that records the address creating the `Equity_round_swap` smart contract (presumed to be a `Safe_controller` contract[18]), and `company`, recording the company smart contract for which the `Equity_round_swap` contract will issue shares. A `status` variable of enumerated type

```
enum SwapStatus {Awaiting_consent,Finalized,Aborted}
```

and initial value `Awaiting_consent` records the current state of equity round workflow.

The process whereby the `Equity_round_swap` is created, by invoking function `start_equity_round` in `Safe_controller`, has already been discussed in Section 7.3. The `Equity_round_swap` smart contract operates as follows:

- SAFE investors, to the extent that they have approval rights on the equity round, give these approvals prior to creation of the `Equity_round_swap` smart contract.
- While in state `Awaiting_consent`, the swap contract accepts function calls `equity_investor_consents`. An equity round investor uses this to consent to the proposed round and pay their principal. Only the correct investor (identified by their address) may make this call, and they must attach the correct principal when calling the function. Correct calls are recorded in the state of the `Equity_round_swap`, which also stores the principal payments.

[18] While a `SAFE_Controller` is the controller of the `Company`, an `Equity_Round_Swap` contract created from any address other than the `SAFE_Controller` will not be able to have any direct effect on the `Company`. This is because it needs to first become the controller of the `Company` itself, and this requires that the `SAFE_Controller` transfer control. But the `SAFE_Controller` will only grant control to an `Equity_Round_Swap` that it has itself created.

- When all the equity round investor consents and principal payments have been received, anyone may call a function `finalize_equity_round` on the `Equity_round_swap` contract, which does the following:
  - Call `finalize_equity_round` on the `Safe_controller` contract. The effect of this call is to terminate the `Safe_controller`.
  - Next the function uses the `Company` function `add_shareholding` to issue shares, as per the proposed equity round, to each of the equity round investors and each of the holders of the SAFE contracts. (If new shares are to be added to the option pool, these would be created at this time.)
  - For each of the SAFE contracts, the company calls `Company` function `convert`. This in turn makes a call from the `Company` contract to the `convert` function of the corresponding `Safe` contract, signaling that the SAFE has been converted.[19]
  - The function then transfers the total equity round investor money received to the `Company` contract.
  - Next, the function calls `set_controller` on `Company` to set the controller of the `Company` to the party or smart contract (parameter `new_controller` in the `start_equity_round` call) agreed to be the controller after the equity round (capturing the post-round governance structure, see the discussion in Section 7.1).
  - Finally, the `Equity_round_swap` contract is terminated by setting the variable `swap_status` to `Finalized`, so that it may perform no further actions.
- It may happen that even though there was an agreement on the structure of the round, not all parties give the consent and payment required for the `finalize_equity_round` operation to be enabled. To cover this eventuality, the proposed equity round has a deadline. Once this has passed, a function `abort_equity_round` of `Equity_round_swap` may be called to do the following:
  - Call function `abort_equity_round` on the `Safe_controller`, to advise it that the equity round has aborted.
  - Restore control of the `Company` to the `Safe_controller`.
  - Set `swap_status` on `Equity_round_swap` to value `Aborted`. This enables the equity round investors to call a function `withdraw` on `Equity_round_swap` to obtain a refund of any money they had sent to this contract.

[19] For security purposes, the `Safe` smart contract needs to verify that the conversion call is being made in the context of an equity round completion by an appropriate agent. There are various ways this could be realised. In our implementation, we use that fact that the sender of the call must be the `Company` smart contract, and that, once a `Safe_controller` has been installed as its controller, `Company` can only receive such a call from an `Equity_round_swap` later established as its controller.

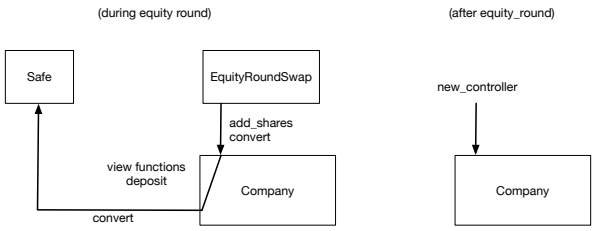

Figure 2: Architecture configuration during an equity round

8 Correctness Arguments

We now discuss how the architecture supports the reasoning required to justify a number of key correctness and security properties. We believe that the sketch below can form the basis for a formal proof of these properties, but we have not yet undertaken this. We ignore implementation-platform specific issues such as gas costs.

At the top level, each of the parties has their own interests, which the smart contracts are intended to enforce. To be assured that these properties will be enforced, each of the parties needs to needs to verify the components indicated.

- **SAFE investor:** The SAFE offer process either terminates with the SAFE active and registered by the company, or with the SAFE terminated and no change to the investor's cryptocurrency holdings.
  Verification Requirements: `Safe_controller` (operations `offer_safe` `accept_safe_offer` and `withdraw_safe`), `Company` and the investor's individual `Safe`.

- **SAFE investor:** Once the SAFE contract has been issued, in the first subsequent Equity Financing event[20]) the company will issue the shares promised in the Equity Financing clause.
  Verification Requirements: The investor's individual `Safe`, and `Company`, `Safe_controller`, and `Equity_round_swap`.

- **Equity round investor:** All and only the shares promised in the Equity Round proposal will be issued if the round is successful, the agreed controller will be installed on `Company`, and the principal will be paid to the `Company`. If the round is not successful, the money paid by the investor can be recovered.
  Verification Requirements: `Equity_round_swap` and `Company`.

- **Company:** The company will grant no more shares than required by the contracts and Equity Financings that it offers, relinquish no more control

[20]If there is one: the SAFE contract does not require that the company conduct an Equity Financing. Other forms of termination, such as by liquidation or dissolution, are not yet supported in our code. The code and the properties would need appropriate modifications once these are added.

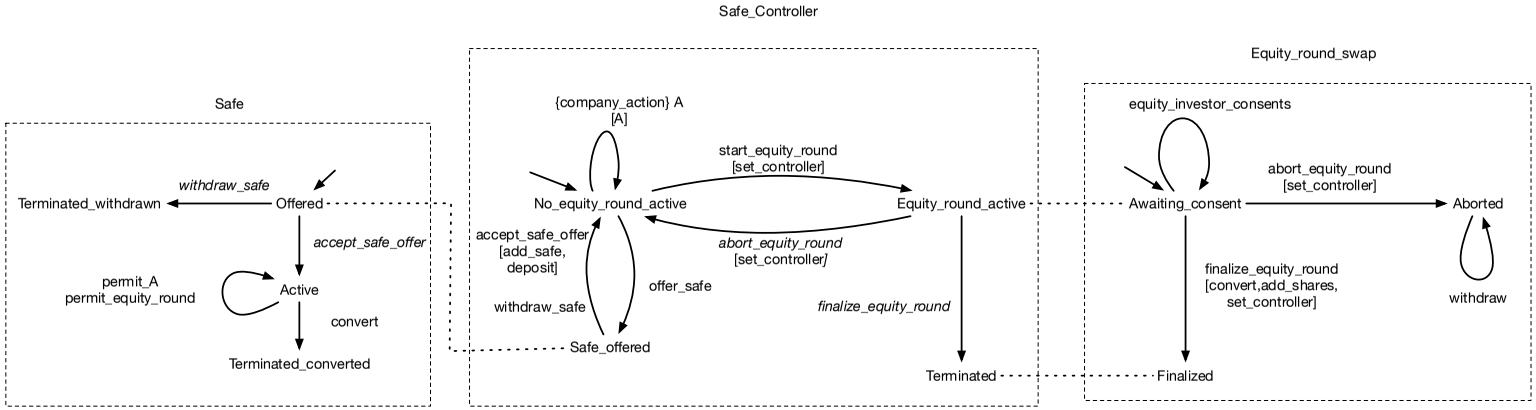

Figure 3: Contract State Transitions

than required by those agreements, and receive principal agreed.
Verification Requirements: All `Safe` contracts, `Company`, `Safe_controller`, `Safe_controller`, and `Equity_round_swap`.

- **All parties:** For security, no other parties should be able to cause a change of state that results in the SAFE investors, Equity round investors or Company losing crypto-currency or a shareholding.
  Verification Requirements: All `Safe` contracts, `Company`, `Safe_controller`, `Safe_controller`, and `Equity_round_swap`.

The verification that the parties need to conduct includes both source code correctness and checking that on-chain bytecode instances of the contracts have been correctly compiled from the source code and have the right parameter settings.[21] For example, the SAFE investor, before accepting the offer of a `Safe` s should verify the following:

- The `Safe` s has the expected setting of the parameters `owner` (this should be an address controlled by the investor), `company` and applicable terms such as `cap` and/or `discount`, and is in state `Offered`.
- A `Safe_controller` sc is the current controller of the `Company`. This controller is in state `Safe_offered`, as a result of a call of `offer_safe`(s), and s.`safe_controller`= sc.
- The bytecode of s has been compiled from source that correctly expresses the terms of the SAFE as agreed between the company and the investor. In particular, the `permit_equity_round` code returns `true` if and only if the proposed number of shares the SAFE investor will receive is as specified in the Equity Financing clause.
- The code in `Company` correctly represents and maintains shareholdings and issuance of `Safe` contracts. The code in `Safe_controller` correctly protects the rights of `Safe` holders. The code in `Equity_round_swap` ensures that an approved equity round proposal is eventually correctly acted upon if the consent and principal required for the round is obtained. The on-chain bytecode for these contracts is correctly compiled from the source code verified.

Our design of the smart contracts applies one of the principles of systems architecture, particularly in the context of secure systems development, which is to isolate critical functionality in components that are small enough to be verified (ideally, using formal methods), combined in the architecture in ways that ensure that the basis supporting key properties remains small and "local".

[21] Checking correspondence between on-chain bytecode and source code can be done using services such as Etherscan (`http://etherscan.io`) or Tenderly (`http://tenderly.co`), but requires trust in these services and the channels through which they are offered. On-chain factory contracts could be used to amortize, or delegate to a trusted third party, the effort that the parties need to put into verifying the contract source code and checking that it corresponds to on-chain bytecode.

Specific localization benefit is clearest for the `Safe` offer-acceptance process and the equity round process.

Note that isolating money in `Equity_round_swap` ensures that it is not co-mingled with other company money and at risk of premature expenditure. Location in the `Equity_round_swap` contract and its receipt of control also enables the equity round investor to be assured of correct transfer of the money and issuance of shares without inspecting the `Safe_controller` or the code of the individual `Safe` smart contracts.

Handling of the SAFE investor's money could have been similarly managed using a swap contract, but this has not been done since it is attached to the `accept_safe` action on `Safe_controller`, which transfers it to the `Company` immediately on successful execution, and the blockchain reverts the transfer if the execution is unsuccessful.

The final security property is largely derivable from use of the controller pattern and authorization checks in the form `require(msg.sender=..)` on each of the operations, with some exceptions that do not adversely affect the principal parties (e.g., anyone may `deposit` cryptocurrency into `Company`.)

The other properties can be derived from more specific properties such as the following, each with an explanation of how the design and implementation ensure that it holds. (We focus below on properties supporting the SAFE investor's receipt of shares in the Equity Financing.)

Proof of these properties is facilitated by the architecture via its state-machine view of the contracts, given in Figure 3. States correspond to the status variables in the contracts. Actions on the contracts are enabled only in the states indicated. An attempt to execute an action when not enabled is handled as a failed and reverted transaction on the blockchain. Some of the events triggering state transitions also trigger events in other machines. This has been indicated by naming the triggered event identically, but in italic font. (For example, action `accept_safe_offer` is called by the SAFE investor on the `Safe_controller`, but triggers a call from there to the similarly named action *accept_safe_offer* on the `Safe`.) When an action calls an action on the `Company` smart contract, this has been indicated by including it in square brackets after the action name. For example, action `finalize_equity_round` on `Equity_round_swap` calls `set_controller` on `Company`. Correctness and completeness of the state machine model can be verified statically from the code (using its `require` statements and subroutine calls). Many invariants also follow from the state machine model and a static analysis of write statements.

In the following, we use *captable state* to refer to the values of variables `shares` and `safes_issued` of a `Company`, and *captable action* to refer to actions on `Company` that alter these variables (e.g. `add_shares` and `add_safe`).

The following properties hold from the time a `Safe_controller` *sc* is installed as the controller of `Company` *c* in its initial state `No_event_active` until the first successful call of *sc*.`finalize_equity_round`:

Property Con: Either *c*.`controller` = *sc* and *sc*.`status` is one of the val-

ues `No_equity_round_active` or `Safe_offered`, or we have that sc.`status` = `Equity_round_active` and there exists an `Equity_round_swap` contract e created by a call of the function `start_equity_round` on sc, with c.`controller` = e and e.`status` = `Awaiting_consent`.
Justification: From `No_event_active`, the first call of `set_controller` on `Company` must come from `start_equity_round`. This sets sc.`status` equal to `Equity_round_active`, creates the `Equity_round_swap` smart contract e in state `Awaiting_consent` and transfers control of c to e. If there is no call of `finalize_equity_round` (on either sc or, equivalently, on e), then either e remains in state `Awaiting_consent` as the controller of c, or there is a successful call of `abort_equity_round` on e which returns control of c to sc and sets sc.`status` = `No_equity_round_active`.

Property Ser: The `Safe_controller` serializes captable actions on the `Company` with the compound events of executing an Equity Round and the Offer-Acceptance process for a `Safe`.
Justification: This follows from the enabledness of actions in the state-machine. Each of the captable actions on `Company` executes atomically, the Equity Round process executes while `Safe_controller` is in state `Equity_round_active`, and the Safe offer-accept process runs during state `safe_offered`. No `Company` captable actions can be called from `Safe_controller` in the latter two states.

Property SI: A `Safe` in state `Active` and registered in `safes_issued` remains in state `Active` and registered in `safes_issued` until an equity round is successfully finalized. Thereafter, it is in state `Terminated_converted`.[22]
Justification: The only action of `Safe_controller` or `Equity_round_swap` that affects registration is `finalize_equity_round` (which triggers `convert` on the `Safe` via `Company`, and clears `safes_issued`). The `Safe` accepts a `convert` action only from `Company`, no action in `Safe_controller` calls `convert` on `Company`, and only the action `finalize_equity_round` on `Equity_round_swap` calls `convert` on `Company`. Since by property Con, either the `Safe_controller` or an `Equity_round_swap` is controller until the first successful call of function `finalize_equity_round`, the `Safe` remains in state `Active` and registered in `safes_issued` until the `finalize_equity_round`.

Property SC: While a `Safe` in state `Active` is registered in `safes_issued`, all actions on `Company` are compliant with the `Safe`.
Justification: This follows from properties Con and SI and the fact that the `Safe_controller` either blocks a `Company` action or queries all contracts in `safes_issued` for permission to perform a company action before transmitting it to `Company`.

Property ERC: A call sc.`start_equity_round` completes successfully iff the terms of the equity round proposed in the parameters are compliant with the

[22]This property will require revision once the Liquidity and Dissolution events are added to the code, to cover these alternate ways to terminate a SAFE.

coding of the `Safe` contracts in `safes_issued`, with respect to the state of the `Company` c at the start of the call. These terms remain invariant in the `Equity_round_swap`.

Justification: The terms are approved by the `Safe` action `permit_equity_round` and stored in the `Equity_round_swap` e created by the call of `start_equity_round`. There are no writes to the variables storing the terms by the actions of e in any of the other actions on `Equity_round_swap`.

Property ERI: There is no change to the `Company` captable state between the start of a successful `start_equity_round` call and the start of termination of the equity round using `finalize_equity_round` or `abort_equity round`. The latter leaves the captable state of the `Company` invariant.
Justification: This follows because the contract e created by the function `start_equity_round` is controller of the `Company` until either the function `finalize_equity_round` or `abort_equity round` runs successfully. The only action on e that makes calls to `Company` is `finalize_equity_round`.

It follows from properties Con, SI, ERC, and ERI that after a `Safe` s has been issued, the first successful occurrence of `finalize_equity_round` occurs on an `Equity_round_swap` that stores an equity round issuance proposal that complies with the terms of s on the captable state of the company at the time `finalize_equity_round` runs.[23] The operation `finalize_equity_round` applies that proposal to the `Company`. This establishes that the key property concerning the Equity Financing required by the SAFE investor is enforced by the smart contracts.

9 Evaluation

Having developed a general framework for the representation of SAFE contracts as smart contracts, and implemented Equity Financing clause components of a number of SAFEs in this framework, we now turn to an evaluation of these results. In particular, we consider the question of whether the smart contracts can replace the use of legal SAFE contracts. We begin with a discussion of applicable evaluation methodologies, and then focus on a comparative analysis of the way that the legal and smart contract SAFEs address a number of risks. We also consider whether our methodologies for dealing with indefiniteness adequately cover the content of the Equity Financing clause in the legal SAFE.

[23] We assume here that s.`permit_equity_round` depends only on the captable state, and not on, for example, the time, or on the amount of cryptocurrency holdings of the company (which might change as a result of `deposit` operations during the time the `Equity_round_swap` runs).

9.1 Evaluation Methodologies

In evaluating our SAFE smart contract implementation, a first question is: what specifically is the object of the evaluation? SAFE legal contracts were designed to address issues in equity financing, and one could evaluate to what extent these contracts, and our smart contract implementations of them, provide satisfactory solutions for this business purpose. As we do not aim to improve upon the business functionality of legal SAFE contracts, we will not attempt to cover this specific question. Instead, we take a comparative approach: assuming that the legal SAFEs are fit for purpose, to what extent do our smart contracts correctly express their functionality? As our work has only covered the Equity Financing clause, our answers will also be specific to this aspect of the SAFE.

Our smart contracts are structured into several components. A number of these, the `Company` contract representing the company, the `Safe_controller` the `Equity_round_swap`, are abstract and technical. Except for laying out the lifecycle and context of a SAFE, the details of the SAFE variants are not represented in these components. A consequence of this is that a formal specification of these components could be written, and formal methods, the most rigorous verification methodology used in computer science, could be applied to give a very high integrity proof of the correctness of the design and implementation of these contracts with respect to this formal specification. (Formal methods are based in the use of mathematically precise notations and the use of automated and semi-automated tools to verify correctness of components with respect to their specifications.) A sketch of such a verification has been given in Section 8, but we leave the details to future work[24], and will not address this aspect of evaluation in the remainder of this section.

A benefit of the way that we have decomposed the implementation is that this leaves the remaining components, the `Safe` contracts, as the sole components that directly related to the legal SAFE contracts, so that the evaluation of the extent to which the smart contracts correctly implement the legal SAFEs can be focussed on this component. In particular, given the properties of the other components, we need to ask: does a SAFE smart contract respond `true` to the question of whether a particular operation is permitted exactly when the legal SAFE would imply that this operation is permitted?

One inherent problem with this question is that the starting point, the legal SAFE, is an imprecise natural language document, whereas the smart contract code has a more precise operational semantics. Particularly where the indefinite terms have been formalized in code (rather than having their satisfaction determined by a process involving human agents), it is therefore not possible to set an exact standard for correctness of the smart contract with respect to the entire legal text.

However, where fragments of the natural language text are sufficiently precise, a correspondence to parts of the smart contract code can be established.

[24]The Solidity programming language, as presented in its documentation, falls short of having a full formal specification of its operational semantics, but work in this direction is in progress, e.g., [JKL+20].

The equations for the Equity Financing clause are indeed precise, so the correspondence can be checked by inspection, under the assumption that the event in question meets the definition of Equity Financing. Ideally, we would like to be able to establish the correspondence unconditionally, but we then must face the difficulty that the conditions under which this event applies involve open-textured concepts, as discussed in Section 4.2. The extent to which our approaches to handling open texture resolve this difficulty is discussed in Section 9.3.

At a broader level, adequacy of the formalization can be evaluated in a number of ways less demanding than a mathematical proof. For example, acceptance on inspection by a legal expert, or by users (company, investors) could be indicative of quality. A problem with this measure, however, is that it may be difficult to find appropriate individuals with sufficient knowledge both of the law and of coding (in the specific language used) to be able to make the judgements required. It would be beneficial, to this end, for the smart contract language to use a programming paradigm that structures programs in ways close to the natural language structures found in contracts. An imperative language such as Solidity does not meet this requirement.

Were these conditions and equations in a legal contract expressed in a *controlled natural language* [Kuh14], a fragment of natural language syntax that has been equipped with a formal semantics that has been justified to accurately capture the natural language semantics, then we could in fact establish a mathematical equivalence with smart contract code. However, establishing adequacy of the controlled natural language with respect to the natural language, is an even more general and challenging problem than the problem of validating a particular text in the controlled natural language! Nevertheless, if the particular controlled natural language text expressing the contract has been written by an individual who understands both its formal semantics and the way that, qua natural language text, it will be interpreted by its intended audience of legal practitioners (lawyers and adjudicators) then there is more likely to be a strong correspondence between the formal and informal meanings.

One of the difficulties with evaluation by inspection is also that it can be difficult to envisage, or reason about, all the possible ways that a contract may be performed. This requires reasoning in the abstract, a cognitively more difficult task than reasoning about concrete facts. Even experts on the intended functionality of a system are likely to be better at answering questions about the "correct" behaviour of the system in concrete examples than in the abstract. This motivates an evaluation methodology based on testing the behaviour of the smart contracts in concrete test scenarios. One can determine whether the smart contract yields the same conclusions as the exemplar.

In principle, one potential source of test cases for smart contracts implementing legal SAFE contracts is case law. Obviously, the usefulness of case law for testing purposes requires that the contracts under consideration in the cases are not bespoke variants of a SAFE. We can hope to benefit here from the fact that SAFEs are intended as a standard contract (although variations have already proliferated). Unfortunately, SAFE contracts may be too new for case

law to provide a satisfactory set of cases for this purpose: there appear to be very few relevant court rulings, and such as do exist are concerned mostly with legal issues that do not bear directly on how a SAFE should be interpreted.[25] Such contracts are "rarely litigated in court" ([CG15], footnote 186). A small number of worked examples of SAFE contract performance are available in "Safe Primer" documents provided by Y Combinator [Y C16b, Y C18a].

The ultimate measure of acceptability is therefore likely to be the reactions of users to situations arising once the smart contract has been deployed. We can then gather metrics such as how often the smart contract results are contested by the parties, and how often an arbitrating authority overrules the on-chain result. As measures of adequacy of formalization, metrics are not without their problems, however. They may be measuring the litigiousness of the users or inclination of the arbitrator to overrule the on-chain result, rather than quality of the formalization. However, these metrics may be of some value for comparisons between two different smart contract formalizations of an informal contract, if one is less contested, or overruled, than the other. These measures also do not separate the issue of formalization quality from the fitness for business purpose, though the latter might rightly be considered to be the more important concern.

The inherent gap between a contract subject to indefiniteness and a smart contract implementation of that contract is familiar to software engineers in the form of the gap between informal requirements and more formal specifications and/or code. Here also, a methodology based on testing, and revision after deployment is the prevailing evaluation methodology. Development methodologies such as *agile* methods aim to accelerate the feedback loop by which user reactions to a working system in concrete scenarios results in refinements of the code.

[25] Some examples are:

- JOHN LEVINSON AND ELLEN LEVINSON, Plaintiffs, v. TWIAGE, LLC, JOHN HUI, GREGORY P. SANTULLI, Defendants. Supreme Court, New York County, 2018. This case relates to claims an investor was fraudulently induced by a company to enter into a SAFE contract.
- CU*ANSWERS, INC., Plaintiff, v. G2LINK, LLC, Defendant. Case No. 2:18-CV-04525-JDW. United States District Court, E.D. Pennsylvania. December 31, 2019. . This case concerns the interpretation not of the SAFE involved, but whether issuing the SAFE satisfies the "Qualified Financing" conditions for conversion of another contract, a Convertible Promissory Note.
- INV Accelerator, LLC, Plaintiff, v. MX Technologies, Inc., et al., Defendants. No. 19-CV-2276 (AJN). United States District Court, S.D. New York. This case deals with the issue of whether a SAFE contract was entered into by the incubator INV Accelerator and a company Goldbean. Goldbean signed a participation agreement with INV Accelerator the terms of which stated that Goldbean would be issued with a SAFE for $ 0.5M, but that SAFE was never signed. More pertinent to the interpretation of the SAFE (but not the Equity Financing clause that is our main focus) is that Goldbean sought to escape the consequences of the Liquidity Event clause when it was subsequently acquired by MX Technologies, by changing the acquisition to a hiring of staff and a transfer of assets to MX Technologies, leaving the company as a shell.

9.2 Risks Controlled

Both legal contracts and computer security mechanisms can be understood as attempts to control risk and make the possible futures more predictable. One approach to evaluating a proposal to implement a legal contract using smart contracts on a blockchain, is to consider the proposal's risk management properties, and compare them to an approach using a legal contract enforced only through the legal system.[26]

Investment in early stage startups involves many risks, and SAFE contracts have been designed to provide protections against some specific financial risks. In particular, they provide downside protection for the investor in case the company is not able to sufficiently increase its valuation, while allowing the investor to benefit from the upside of a high valuation. For the Equity Financing clause that has been our main focus in this paper, the technical parts of the text of the legal SAFE are precise, and the smart contract code is obtained by a straightforward translation to code. Hence, subject to the necessary assumptions about the completeness and legal status of the on-chain representation of the company's cap table, discussed in Section 10, the SAFE smart contract provides similar protections. One proviso on this similarity relates to the open-textured concepts involved in the Equity Financing clause. We discuss these at greater length in Section 9.3.

Some risks to an investor are obviously not mitigated by a legal SAFE contract, and we should therefore not necessarily expect that smart contract implementations of a SAFE will be able to mitigate these risks. Risks outside the scope of a SAFE legal contract include behaviors by company management that are not illegal, but which disadvantage the SAFE holder. For example, founders may cause losses to SAFE investors by failure to develop the company, or by poor management of company funds. Founders may also disadvantage SAFE investors, decreasing the value of the shares to be received in advance of conversion of the SAFE, by paying out dividends[27], or spinning out some of the assets of the company and using a stock distribution to shareholders. If the company is able to become profitable enough to fund its own operations and development, it may avoid ever conducting a priced equity round, preventing the SAFE investors from obtaining any benefit from their investment. These uncovered risks might well be priced in by sophisticated investors (via the values of the SAFE parameters, i.e., the cap and or discount) when purchasing a SAFE, so it can be questioned whether these are actually weaknesses of the instrument or deliberate omissions in its construction.[28] It is up to the SAFE investors (e.g., through their personal engagement with company management) to ensure

[26] The practice of legal contracts may differ from the theory - business has a social dimension and often operates on trust. Contracting parties may prefer to use contracts for purposes of moral force, and avoid litigation in favor of negotiated settlements that differ from the enforceable terms of the contract [Lev17].

[27] The Post-Money SAFEs but not the Pre-Money SAFEs provide for SAFE holders to receive dividends.

[28] Attitudes of the courts to protecting convertible bond holder interests against those of shareholders, and the potential judicial bases for intervention are discussed by Bratton [Bra84].

that company governance effectively mitigates these risks. Where risks outside the scope of a legal contract can be mitigated by governance, these risks can be mitigated in exactly the same way when the legal contract is implemented as a smart contract.

Other risks outside the scope of a SAFE legal contract include illegal activity such as embezzlement of company funds by company staff, or provision of fraudulent data on the basis of which a valuation is established by an investor. Criminal law provides a measure of deterrent protection and recompense in this case. To the extent that there is a clear legal jurisdiction for the company's activities, the same protections are available when the SAFE is represented as a smart contract.

One risk outside of the scope of the legal SAFE that is addressed in the smart contract implementation is the risk of errors in the company's cap table due to company insider fraud, incompetence and/or outsider interference (e.g. malicious hackers of computer records). By placing shareholder records under the control of keys in the hands of investors or their proxies, a smart contract implementation distributes this risk away from the company or its registry, and replaces it by the residual risk of attacks on the blockchain validators and the mechanisms and processes by which cryptographic keys are protected.

It is in the spirit of the smart contract field to design novel mechanisms that mitigate risks by means of economic incentives and/or penalties. A common approach is "staking", or posting of cryptocurrency or token collateral by one or more of the parties to an interaction, with a smart contract enforcing that the collateral is lost in case evidence can be provided that the party has not adhered to required behaviors. Our aim in the present paper has been to determine to what extent a legal SAFE can be implemented on its own terms as a smart contract, rather than to develop new mechanisms for the problem solved by legal SAFEs, so we have not attempted to develop such alternate mechanisms. However, it is questionable whether collateralization would be an appropriate mechanism, given that legal SAFE's are not in practice collateralized, and indeed to do so would be against the spirit of the high-risk tolerant investment posture of SAFE investors. The aim of a SAFE investment is to make funds available for company development, and locking funds up defeats the intent of the investment. Where the founders have non-monetary assets that they are prepared to put up as collateral for an investment, the resulting instrument would no longer be a SAFE. Our smart contracts could be extended to build in protections such as burn-rate constraints, but we do not pursue this for similar reasons.

9.3 Non *bona fide* Equity Rounds

As noted above, the Equity Financing clause in the legal SAFE contract contains "open texture" language which is difficult to express in code. Our proposed approaches to dealing with this difficulty for smart contract representation include limitation of fact patterns and either deferring decisions on the applicability of open texture concepts to (negotiated) human consideration, or replacing im-

precise contract terms by precise ones in the interest of enabling automated enforcement. How satisfactory are these approaches for dealing with the Equity Financing clause? Do they enable the legal language to be entirely replaced by corresponding smart contract code? The following example (repeated from [vdMM21]) suggests that our smart contracts leave the SAFE investor exposed to a risk against which the legal SAFE provides protection.

Example 1 *Consider a company whose founders have 1,000 shares, and suppose an investor, Saffron, buys a Pre-Money SAFE with a valuation cap of $1,000,000 and no discount, for principal $10,000.*

We suppose that later, a new investor Neville has $1,500 to invest, and that the development of the company has gone poorly, so that its valuation has dropped significantly to around $3,000, or $3 per share. If this investment were made using an equity round that discharges the SAFE according to the Standard method, the numbers of shares after the equity round would be Founders: 1,000, Saffron:3,333, Neville: 500, and proportional shareholdings, roughly, Founders:21%, Saffron:69% and Neville:10%. Saffron would have control of the company in this situation.

To avoid this outcome, Neville and the company collude to limit the influence of Saffron after an equity financing, by structuring Neville's investment into two rounds rather than one. It is agreed to structure the two rounds so as to give Neville a 1/3 stake at the end of the two stages. (Note that this is equivalent to the stake Neville would obtain if the SAFE were not present.)

Specifically, in spite of the low valuation, in the first round, Neville provides equity financing of $1,000 at a Pre-Money Valuation of $1,000,000 and hence a price of $1,000 per share[29]*. After this financing, the founders have 1,000 shares, Saffron has 10, and Neville has 1, so the proportional shareholdings are, roughly, founders: 99%, Saffron: 0.99%, and Neville: 0.01%. The SAFE is terminated in this round. Saffron might well be satisfied with the deal since it implies a valuation of her shares roughly equal to her original investment, so she avoids a paper loss.*

Later, in a second round of equity financing, Neville uses the remaining $500 to buy 504 shares at a price $0.99206 (a valuation of $1,002.976). The SAFE has terminated, so Saffron receives no shares in this round. The capitalization is now 1,511 shares, of which the founders have 1,000, Saffron has 10, and Neville has 505, or proportions of 66%, 0.7% and 33.3%.

Thus, overall, rather than a 10% stake, Neville has obtained his desired 1/3 stake for the same money, and with a significantly different outcome for the founders and Saffron. The founders retain almost all of the controlling stake that Saffron would have obtained in a Standard SAFE round. If Neville had paid this price in a single round, Saffron would have received 3,334 shares and become the majority shareholder.

If Saffron has a pro rata *rights agreement with the company then she can obtain some shares at the same price as Neville, but that simply maintains her*

[29] Note that the valuation in this first round is equal to the cap. It was shown in [vdMM21] that this gives the optimal share to the founders.

shareholding at about 1%. She still has arguably been defrauded of a majority share in the company.

Our general approach allows smart contract implementations of SAFE contracts to select a point on the spectrum between full automation of the equity financing in which the SAFE converts, and implementations of processes that give the SAFE investor bargaining rights in the construction of that equity round. Because the smart contract enforcement of the computations or processes is "nuncospective" (as discussed in Section 4.5), and the manipulation might not be detected until the second round, neither approach necessarily prevents the manipulation from occurring. However, it is reasonable to take the view that the legal SAFE contract protects against such collusion by requiring an Equity Financing to be a "bona fide transaction or series of transactions". Thus, the initial round of equity financing might not be considered *bona fide*. Alternatively, the two rounds of financing might be considered as a series of transactions, so that the number of shares due to Saffron would be based on a collective price. Such judgements can be formed only after the events have taken place. Enforcing such interpretations would require a legal judgement. There are therefore reasons to retain the protections of the legal text even when working with smart contract implementations of the SAFE.

One way that the SAFE investor might protect themselves against this manipulation on the part of the founders and new investor is on the basis of a valuation of the company at the time of the first equity round. Startup company valuations can be highly subjective: there is no market in the shares and the company is likely to still be unprofitable. Nevertheless, a valuation by an independent party might provide a basis for negotiation or legal arbitration.

On the consent approach, the SAFE investor can use an independent valuation in their negotiations at the time of the first equity round. In the example above, if the independent valuation comes in at $3,000, or a price of $3, then Saffron would have to persuade the founders and new investor that the lower second round price is fairer. An equity round for $1,000 at $3 would give a distribution of founders: 1,000, Saffron: 3,333, Neville: 333, or, proportionally, roughly, founders: 21.5%, Saffron: 71.5%, Neville: 7%. Compared to the results above for Neville's first round investment at $1,000, this is a significant loss of control for the founders, but substantially more control for Neville (7% versus 0.01%), so if he were acting rationally, Neville would take Saffron's side. If Neville rejects this valuation in favor of a higher valuation then Saffron is alerted to the fact that something fishy is afoot, and has the option of litigation, which might be a sufficient threat that Neville is prepared to compromise.

In case the round is automatically converted, the objection would have to go direct to court, which might again side with the independent valuation. Going to court after the second round would provide even stronger evidence for Saffron's case. The risk of an adverse legal ruling might be sufficient to deter Neville from engaging in the manipulation.

A variant of the automated conversion smart contract could be constructed where the SAFE investor is able to dispute a proposed equity round on-chain

using a valuation signed by an a-priori selected arbitrator, triggering a switch to the consent based approach if the valuation difference is greater than a threshold. (Another alternative is to always determine the price of the round using a valuation signed by this arbitrator. This, however, has the disadvantage of constraining the founders' negotiations with the new investor, who would often expect to have a strong hand in setting the price of the round.)

9.4 Evaluation Summary

In the above, we have considered evaluation of our proposed approach to smart contract implementation of SAFE contracts from a number of angles. The conclusion of our discussion of evaluation methodologies is that the inherent gap between the indefiniteness of the legal SAFE contract and the definiteness of the smart contract implies that it is necessarily the case that, beyond the partial assurance provided by testing, we cannot obtain a complete assurance that the smart contract will behave in the same way as the legal contract in the context of the legal system. A consideration of risks leads to the conclusion that some risks must necessarily be addressed by the legal system. In particular, a legal SAFE provides protections against a two-round manipulation that the smart contract cannot fully address on its own. On multiple grounds, therefore, we conclude that in order to provide the full legal functionality of a legal SAFE, a smart contract implementation should be associated to a legal contract that provides protections that the smart contract omits. We therefore consider smart contract interactions with the legal system in the next section.

10 Legal Issues

In this section we discuss a number of issues of law arising when representing SAFE contracts as smart contracts. A full treatment of this topic would be jurisdiction dependent, and may need to delve in to case law, legislation and regulation. We do not attempt to go to this level of detail, confining ourselves to the identification of some legal issues and options for addressing them.

10.1 Applicability of Law

A first question is whether or not law is applicable to smart contracts. Some have held that it is not. We might use the term "Smart Contract Absolutism", for the position that the state of the blockchain is definitive with respect to facts concerning ownership of on-chain assets and powers exercisable with respect to those assets, and not subject to any external legal jurisdiction.[30] On this view, the state of the blockchain cannot be incorrect, and an event on-chain is

[30]The position is often expressed as "Code is Law". We prefer not to use this phrase to describe the position, since it appears to have been first used by Lawrence Lessig [Les00] to make a quite different point: that code implicitly imposes regulatory assumptions, and ought to be designed to build in Constitutional, i.e., legal values.

permissible if and only if it is enabled by the code of the underlying blockchain and its smart contracts.

In practice, this position has encountered a number of difficulties. One is that smart contract code may contain errors and security vulnerabilities, so that its behavior does not match the intentions of its designers and the expectations of its users. Errors sometimes have devastating consequences, as in the attack on a security vulnerability of The DAO in June 2016 [Pop16], in which around $US50M worth of value was stolen from a smart contract. Ironically, the adherent of Smart Contract Absolutism has deprived themselves of legal protections against such malfeasance. Indeed, the attacker in this instance justified their gains as "fair game" using Smart Contract Absolutist arguments. Interestingly, the Ethereum community in general took the position that the attack was not part of the intent of the smart contract, and took action to defend it and retrieve stolen funds by making changes to what had been supposed to be an immutable blockchain. (Their interest in doing so may have been more to protect the still-nascent Ethereum platform, however. This precedent has not been followed for later attacks.)

A second difficulty for a Smart Contract Absolutist position taken with respect to a smart contract is that legal authorities may nevertheless assert that its operation falls within the scope of the law. An example of this is the SEC Report of Investigation on The DAO [US 17], in which the SEC asserted that The DAO tokens should be classified as securities subject to US securities laws (although no action was taken since The DAO had already been dissolved after being attacked).

In general, where a smart contract impinges on real world matters in the scope of the law, one expects that the law will assert itself.[31] The legal system provides a social function of supporting trust in interactions between parties through the enforcement and remediation of contracts. The paper [PD19] asks whether smart contracts offer the prospect that these roles of the legal system can be eliminated by use of smart contracts. It concludes that, while a reduced reliance on the legal system may be achievable, smart contracts cannot escape legal review, and the option of contracting parties to seek legal remediation cannot be eliminated.

This is likely to apply, in particular, to on-chain representations of company cap tables and equity instruments, which represent rights relating to persons and entities that are subject to a legal environment, and engage in real-world activities. We therefore expect that the law will assert itself with respect to smart contract implementations of cap tables and related instruments such as SAFE contracts.

[31] Arguably, the Smart Contract Absolutist position is more sustainable for wholly virtual cryptocurrency holdings, which do not represent any fact in the real or legal world. Even here however, the fact that persons ascribe value to these holdings has led to the assertion of tax law over these holdings.

10.2 Legal Status of Smart Contracts

If one does not take a Smart Contract Absolutist position, and accepts that matters that one is aiming to represent in a smart contract do fall within the scope of a particular legal jurisdiction, several further questions then arise.

From the point of view of the users of the smart contract, it becomes desirable to understand the precise legal standing of the matters that it purports to represent. In particular, for inherently legal facts such as those relating to ownership of shares in a company, it is desirable to understand how the cap table as represented on-chain is related to the cap table of the company "in law".

A first question is whether the law accepts that the blockchain has any relevance for questions relating to a company's cap table. In some cases, the law may in fact rule that a smart contract *cannot* be an official record of the cap table of a company. For example, regulators may require companies to use approved (non–blockchain) registries to represent their shareholder records, or insist that particular custodians be used to hold share certificates. If this is the case, then the blockchain can at best be a non-official copy of the official register. In general, such a regulatory stance is more likely to occur in the case of publicly traded companies, with private companies (including the startups that are most likely to use SAFEs) given significant freedom with respect to how they maintain their shareholder records.

Once it is allowed that the blockchain may record data relevant to the cap table in law, a second question is whether the cap-table in law is determined solely by the state of the blockchain, or also by facts not represented on-chain. Shares may typically be issued by a company using paper certificates. If the company issuing shares on a blockchain retains the option to issue paper shares, from the point of view of the law, the official state of the cap table will be determined not just from the blockchain, but also by the off-chain paper certificates. Other real-world facts concerning actions of the company's officers may also be pertinent.

This means that the on-chain representation of the cap table may differ from the legally accepted state. From the point of view of the holder of a SAFE contract represented on chain by a smart contract, such a divergence is problematic, since actions of the smart contract assume that the on-chain record is complete. For example, the equity round clause makes use of the notion of Company Capitalization (the total number of shares issued). If the shares recorded on-chain are only a subset of the shares issued in law, then the calculations performed by the SAFE smart contract on-chain will not yield a correct share issuance to the SAFE contract holder at the time of the equity round.

To avoid this problem, some method for assuring the completeness of the on-chain representation is required. Possible approaches to such providing such assurance could include

1. public communications by the company promising to use the blockchain exclusively to represent its cap table, provided that the law accepts such

statements as giving the blockchain representation legal force,

2. inclusion in legal contracts associated to on-chain SAFE, and other share issuances, of terms to the effect that the on-chain representation of the cap table is the official representation, again assuming that the law considers the company to have the right to include such terms, or

3. the fact that the blockchain is operated by an entity recognized by the securities regulator as the official registry for the company.

None of these approaches *guarantees* that the on-chain representation is correct, but they do provide injured parties the option of seeking legal remediation when errors occur.

A similar question concerning legal status arises for the convertible instruments issued by a company: if these are represented using on-chain smart contracts, are these legally recognized? Particularly for Post-Money SAFEs, for which the conversion calculation makes reference to other SAFEs issued, it is again critical for correctness that the on-chain representation of SAFEs covers *all* SAFEs that the company has issued in law.

It would appear from the above considerations that a SAFE smart contract, both in order to obtain legal standing, and to be meaningful from the point of view of correctness with respect to the company's cap table in law, should be accompanied by a legal contract that contains terms relating to the legal standing of the on-chain representations of the cap table and the SAFE contract. This adds to the reasons that there should be a legal contract backing the smart contract that were already identified in Section 9. We discuss the possible form of such a legal contract in Section 10.4.

A further legal issue with respect to shares represented on-chain is that regulatory authorities may require that they be notified of certain changes to the cap table. This could be implemented either by the regulator directly reading the on-chain information, by using off-chain processes, or both, so does not impact either the legal contract or smart contract except inasmuch as it may be beneficial to include events in the smart contract that can serve as triggers for any off-chain processes.

10.3 Legal Activation

Contract law governs conditions under which parties cause a contract to be activated, and come to be committed to its terms. These conditions may include a process of offer and acceptance, and often involve signing the text of the contract.

The usual process for parties to become committed to a smart contract, is that one party places the smart contract on-chain, and one or more parties then participate in it by sending it transaction messages, which may have attached value. These messages are signed, but the content signed is not the "text of the contract": rather, the signature applies to the message being sent to the smart contract, which includes the function being called, the values of the parameters,

and the amount of any attached cryptocurrency. The message serves to identify the sender (by public key) to the smart contract, and to authorize the blockchain miners to transfer the cryptocurrency amount from the control of the signing key to the control of the smart contract.

A process similar to this is used in the case of our implementation for the activation of a SAFE. The company first creates a `Safe` smart contract using the constructor of one of the specific subclasses representing particular types of SAFE. This smart contract is then "offered" to the SAFE investor by calling the operation `offer_SAFE` of `SAFE_controller`, with parameters the `Safe` contract and a deadline for acceptance. The SAFE investor "accepts" the offer by calling the operation `accept_SAFE_offer` of `SAFE_controller`, and attaching their principal payment in cryptocurrency. The effect of this is that the payment is transferred to the `Company` smart contract, and the `Safe` is registered as one of the issued SAFE contracts.

A potential legal problem with this approach is that although it may establish certain *de facto* powers on the blockchain, the parties' assent to the terms of the contract is *implicit*, demonstrated by the fact that they participate in the contract's individual events. This approach is most consistent with the Smart Contract Absolutist perspective, in that the parties choose to participate in the smart contract and are, as a result, required to accept the behaviour of the smart contract as authoritative. However, what is missing from a legal perspective is direct evidence of their assent to the contract as a whole, usually provided by having both parties sign the contract.

If the parties desire, or are required to, enter into a legal contract that either gives the smart contract legal force, or covers aspects of their agreement and dispute resolution needs that are not covered by the smart contract, then a more explicit process may be required in order to give effect to this contract. One option is to have the parties sign a paper contract off-chain, that includes enough detail about the smart contract to link the two.

Alternatively, if the legal jurisdiction in question accepts digital signatures[32], we could handle explicit legal assent as follows. Fields included in the smart contract that can store a signed copy of the legal contract (or a more space efficient signed copy of the hash of the contract), one for each party. The smart contract can include operations by which the parties to the contract can save their signed copies into these fields. The remainder of the smart contract can be enabled once both parties have stored the correct signed content. This has the effect of activating the smart contract once both parties have signed.[33]

For some contracts the legal assent could be one-sided: the offering party creates on-chain a smart contract containing a signed copy of the offer docu-

[32] For digital signatures to have legal force, some means of binding the signature keys to legal identities is required, e.g., a legally recognized public key infrastructure.

[33] When legal activation is achieved by both parties creating (and somehow logging on chain) certain signed messages that represent their signature on the legal contract, each party will not wish to be legally committed until the other has also signed. Van der Meyden [vdM21a] discusses this issue, and develops formal characterizations of the offer and acceptance process and a process of "signature in counterparts".

ment, and the accepting party interacts with the smart contract, possibly providing payment as consideration, but without providing a signature on the offer document. This may work for contracts in which the only obligations are on the side of the offerer: the accepting party is able to back claims for damages for non-performance using the signed statement by the offeror. However, the SAFE contract includes commitments from both parties. The SAFE investor is required to assent to the "Investor Representations", specifically with respect to legal capacity and to being accredited.[34] The Equity Financing clause, moreover, states an obligation of the SAFE Investor, that they should execute (sign) the Equity Financing documents. It is less clear how the investor could be bound to this obligation if they have not explicitly signed the SAFE.

10.4 Towards a Smart Legal Contract

In the preceding sections we have identified a number of reasons that a written legal contract should be used by the parties even when a smart contract has been used to implement a SAFE. These reasons include ensuring that the smart contract's representation of the cap table will be valid in law, ensuring that the smart contract correctly reflects the intended legal state of affairs, leaving interpretation of open-textured concepts to human processes, covering parts of the contract that are not implementable in the smart contract, and providing a fallback ensuring availability of remediation in cases of errors. We now consider the form and necessary content of such a contract. (Since some of the key provisions are likely to be jurisdiction dependent, we do not attempt to develop detailed contract text, but focus on what needs to be expressed.)

To cover additional risks that are associated with the use of the smart contracts, ensure that these will be interpreted in law as they are intended, and enable the parties to rely upon the smart contract concerning the state of the company's cap table, we suggest that the clauses with the following implications could be included in a contract between the SAFE investor and the Company.

- The Company warrants that the `Company` contract at (address), controlled by the `SAFE_controller` at (address), will be operated in such ways as to ensure that it is maintained in a timely manner as a complete and correct record of the state of the Company's cap table and convertible instruments issued.

- The Company warrants that it will take all actions necessary (e.g., legal filings and notifications) to ensure that the record in the `Company` and `SAFE_controller` smart contracts will be considered in law by (the applicable jurisdiction) as the official record of the Company's cap table.

- The company will conduct its next equity round on-chain, by submitting a `start_equity_round` transaction. (See Section 7.4.)

[34] An alternate way for these to be established could be by including in the activation conditions of the smart contract that digital certificates attesting to these facts be provided to the smart contract.

- The Company will submit a `start_equity_round` transaction only when the smart contract is in a state completely and correctly representing the state of its cap table and convertible instruments issued.

- The Company warrants that in any `start_equity_round` transaction submitted, the values of the parameters correctly and completely represent a valid Equity Financing event as defined in this contract.

- The Investor warrants that they will take all actions necessary (e.g. monitor the blockchain or appoint a watchtower service) to ensure that they will be alerted in a timely manner of any states arising in the operation of the smart contracts that require submission of a transaction or transactions by the Investor. The Investor will react to such alerts by submitting a transaction, as required, by the deadline given.[35]

- (In a variant of the SAFE smart contract in which the investor has the opportunity to consent to the terms of the equity round, with the hash of the Equity Financing documents included in the parameters) the SAFE investor's submission of a `safe_investor_consents` transaction will imply that the investor assents to the terms of the Equity Financing documents. (See Section A.1.)

- Disputes relating to the operation of the smart contract will be adjudicated in the first instance by an independent arbitrator appointed by (Arbitration Authority) according to (appointment process description).

We note that these provisions allow for some divergence between the on-chain record and the legal state, e.g., in case of bugs in the smart contract, malicious attacks, or legal rulings, but imply that the Company will correct the record in such events.

A number of options are available for the form of the legal contract, and the way that it relates to the smart contract code. Not all parts of the original SAFE will be represented in the smart contract — these parts should be retained as natural language text. For parts of the original SAFE that are represented in the smart contract, the question arises of whether to retain these parts in the legal contract, or to replace them either by code or references to code.

Grigg [Gri20] has argued that there are a number of problems with the idea of a direct inclusion of code in contracts. First, code may not pass a "readability test" for contracts "in writing", since code is "opaque" even for experienced programmers, in the sense that the correctness of code in all circumstances can be difficult to ensure. Next, code will normally be accompanied by explanatory documentation in natural language, which would be more transparent and hence better suited for interpretation by courts than the code, for which they will require expert witnesses. Finally, where both code and natural language text exists expressing the same content, one must dominate. The statement of

[35] An alternative to this clause would state that the company has an obligation to inform the investor of the actions that need to taken, and specify the acceptable means of communication.

dominance can only be expressed in natural language, so Occam's Razor suggests that natural language should dominate, although the natural language may explicitly defer to code.

We largely concur with these points, though we would note that the process of writing code and formally verifying its properties (the latter rarely undertaken in general, but increasingly being pursued for smart contracts) is likely to reveal ambiguities and unforeseen implications of the natural language text. The improvements to code resulting from this process can be reflected back to yield improvements to the natural language text. We would also argue that use of mathematical style, including equations, would clarify the natural language text of financial contracts. Being able to automatically extract a formal model of the contract from the contract itself (rather than having to manually construct one) would enable the parties to conduct formal analysis and simulation of the performance of the contract in order to decide whether to accept a contract offer. The best way to combine the benefits of natural language and automation would therefore be to represent the formalizable parts of the contract in a controlled natural language, enabling extraction of code.

Pending availability of a suitable controlled natural language, the best compromise is a plain natural language contract with references to code. In the case of the SAFE, the legal contract that results from the above considerations then looks largely like the original SAFE contract plus the above provisions relating to operation of the smart contracts. One might question, therefore, what benefit is obtained from the smart contract. By way of answer, we point to the motivational remarks in Section 2 – that a legal contract may be used to overrule a smart contract in some cases does not vitiate the benefits resulting from the smart contract, including improved efficiency, security and consensus between the company and its investors.

Riccardian contracts [Gri04] have been proposed as an integration of computable elements and natural language text that can be be accepted as contracts by the legal system. In their original form, these contracts applied to simple money-like instruments (e.g., deposits in a bank), and were structured as a digitally signed set of key-value records that could be bundled together with the instrument so that a human recipient would receive a statement of its terms and conditions, while at the same time supporting automated processing. Richer forms have since been developed [Cla18].

As a SAFE is not transferable, bundling of the SAFE smart contract and the legal text may not be necessary, but can nevertheless be achieved by including in the smart contract a (write-once) field for the signed (hash of the) legal text, smart contract source code and compiler information (so that on-chain bytecode can be verified to be a compilation of the source).

10.5 Adjudication

Once it is accepted that a smart contract will be accompanied by a legal contract, which may be used to overrule the smart contract, the issue arises of how disputes relating to the legal contract will be handled. The usual legal processes

will of course apply, but these come at a high cost (in time as well as money), which counteracts the efficiency benefits that smart contracts are intended to provide.

If the parties are unable to negotiate a resolution on their own, a cheaper and more efficient process may be to make use of private courts and arbitration services, whose rulings are increasingly accepted by legal systems and their enforcement processes in a number of domains [Can04] . There have been proposals to develop an arbitration ecosystem specific to blockchain and smart contract environments, which would have the benefit that the arbitrators would have greater expertise in the technology than can be expected from existing court systems.

CodeLegit [Cod] has proposed an approach to arbitration of disputes relating to smart contracts using *Blockchain Arbitration Libary* code that is combined with the smart contract. The parties to the smart contract are assumed to have signed a legal agreement containing the text

> Any dispute, controversy or claim arising out of or relating to this contract, or the breach, termination or invalidity thereof, shall be settled by arbitration in accordance with the Blockchain Arbitration Rules.

A party that wishes to dispute an event in the smart contract may make a function call that pauses the smart contract and starts the arbitration process. The process involves an *Appointing Authority*, who appoints an arbitrator (possibly with input from the parties concerning a list of candidates), and is responsible for reactivating, terminating or modifying the the smart contract depending on the settlement or ruling of the arbitrator resulting from the arbitration process.

Kleros Courts [Kle] is a proposed crypto-economic protocol for adjudication. Arbitrators are required to stake cryptocurrency in order to participate in this ecosystem, and are incentivized to make "correct" decisions as a result of the protocol allowing decisions to be contested and referred to a larger group of arbitrators, with the original arbitrator penalized for failing to make the majority decision.

Both the technology and the ecosystem of blockchain arbitrators are nascent, and it will take time for them to reach an acceptable degree of trustworthiness (e.g. for reliability, correctness of decisions, and freedom from corruption). Their legal status, critical for enforceability of their decisions, also remains to be determined. As SAFE contracts ultimately derive their enforceability from a national or state legal jurisdiction, acceptability of the arbitration process to that jurisdiction is likely to be critical.

11 Enabling Response to Legal Orders and Variations

As noted in the introduction to Section 10, the legal system supports trust in multi-party interactions through enforcement of contracts and remediation

of their breach. We now consider how the possibility of legal orders impacts the implementation of smart contracts in general and SAFE smart contracts in particular. The implementations discussed above do not directly support variations required by legal orders, so accommodating the discussion of the present section in the code is is left to future work.

Enforcement orders sometimes place no special demands on a smart contract. Where the history of an interaction is consistent with a legal contract, except that some party has not honored an obligation under that contract[36], an enforcement ruling may assert the legal validity of the contract (if this is disputed by the offending party) and order that the obligation be complied with. When the agent of the obligated action is human, and the action occurs either in the real world or can be performed by that agent submitting a number of transactions to the blockchain, there are no special demands on the smart contract. Compliance with the order to perform the action suffices to transform the smart contract state so that it conforms to the legally required state.

Examples where this circumstance may arise in the case of smart contracts for SAFEs are the Company's obligation to perform an equity round on-chain by submitting an appropriate `start_equity_round` transaction (see Section 7.4), and the company's implicit obligation to notify the SAFE investor and provide equity round paperwork and the SAFE investor's obligation to sign this paperwork (see Section A.1).

Similarly, obligations enforced through the smart contract are largely unproblematic. On event-driven platforms such as Ethereum, it is typically necessary for some party to submit a transaction to trigger an on-chain performance of the obligation. In effect, this transforms the obligation for an action to be performed into an obligation to submit a transaction. Commonly this is achieved by enabling the required transaction to be submitted by any party who might be disadvantaged by failure to perform the obligation, giving them the powers to ensure performance. (We do require that some party who has the power to submit the transaction is aware of the contract state and the obligation or opportunity to submit the transaction.)

An example of this is the obligation on the company to issue a certain number of shares to the SAFE investor in the event of an equity round. For the SAFE share issuance obligation, a call to the `finalize_equity_round` function guarantees conversion of the SAFE contract to shares, once consent of all required investors has been obtained and payment received. Any party may make this call. In particular, the SAFE investor has the power to enforce the obligation to issue shares once the equity round is ready to complete.

In automated conversion implementations, the *number* of shares issued is as

[36]The simplest case of this is obligations that are not subject to a deadline. It may also hold where a deadline for performance has passed, but late performance of the action would restore the smart contract to a state consistent with the remainder of the contract. Where nonperformance by the deadline triggers contract conditions such as penalties that would remain in place even if the obligated action is performed late, the legal analysis of the situation becomes more complicated, and and enforcement rulings may involve contract modifications, discussed later in this section.

prescribed by the SAFE contract. The sense in which the obligation is being enforced is more subtle in implementation in which the SAFE investor is required to consent to the equity round terms. Here, the SAFE investor may consent to a number of shares to be issued which differs from that described in the legal contract. In effect, the parties to the contract have first agreed on-chain to a modification of the obligation, and it is the modified obligation that is enforced on-chain. (It would be beneficial for this process in the operation of the contract, and its legal interpretation, to be made explicit in the legal contract governing use of the smart contract.)

In general, however, rulings arising from contract disputes may either invalidate the contract, or require modifications to the contract. Rulings that the contract has been breached, and that the interests of one of the parties harmed, typically also involve some form of *remedy* or *compensation*.

There are competing views of how the appropriate compensation should be determined. One view is that the compensation should restore the injured party to a state before the breach occurred. Economists have taken the contrary view that compensation should be determined so as to ensure that the rational behavior of agents, given the compensation to be expected for breaches, will satisfy some criterion, such as "efficient breach", in which the contract is honored unless all parties (e.g., two parties with a sales contract and a third party that offers a higher price for the goods) benefit in the event of a breach by some party. A survey of economic analyses of compensation [Cra03] has concluded that this perspective does not lead to a unique "correct" theory of compensation, since there are many criteria that such a theory might seek to optimize, leading to divergent conclusions about the appropriate compensation in a given case. In either case, analysis of concrete examples may involve considerations of *causality*, itself a subject of considerable philosophical dispute [Moo03].

In computer science, there is a related area of work motivated by the concurrent processing of long running database transactions, where one transaction may cause another to fail. One response to this situation would be to "roll back" the interfering transaction, restoring the database to a prior state. However, this is problematic when some of the events have had effects in the real world, or in other systems that are not under the control of the database. It has been proposed to address this issue using "compensating transactions" to undo the undesired effects. Here also, there are many competing theories for how such compensating transactions should be determined [CP13], depending on what one chooses to optimize.

In the light of the unsettled nature of these theoretical approaches to compensation, and their inherent dependence on subjective choices about what should be optimized, automation of smart contract remediation and compensation grounded in a specific theoretical account of compensation would not appear to provide sufficient coverage of the requirement to be able to deal with court rulings. Adjudicators and courts will apply their own choice of approach, and may well make decisions that lack strong theoretical grounds. What is more critical is to enable compliance with the full range of potential legal revision and remediation orders.

The way in which legal revisions are dealt with at the level of smart contracts may depend on the underlying blockchain and its governance processes. Permissioned blockchains are the least problematic, since the validators are known entities and, provided that they accept orders from the contract's jurisdiction, can be ordered to make modifications to the blockchain. history. Matters are more complex on open blockchains, where the blockchain is immutable (except for very infrequent hard forks), validators are not necessarily known, and are internationally dispersed. Even on permissioned chains, the governance policy may prefer to minimize the extent to which validators are called upon to modify the blockchain and prefer that smart contracts be coded so as to accommodate legal modifications.

A classification of legal revisions has been discussed by Marino and Juels [MJ16], who consider the capacity of smart contracts in Ethereum's Solidity language to be designed to accommodate various legal revisions. They identify two types of legal changes to a contract: *rescission* (or termination) and *modification*. Rescission renders the contract legally invalid and frees the parties from its constraints, whereas modification leaves the contract in place, but forces changes to its terms. Depending on the contract state, both types of orders may require compensation to reverse the effects of actions already performed with respect to the contract. Each type of change is caused by one of three types of powers:

- "by right", where one of the parties to the contract has rights in the contract to cause the change,
- "by agreement", where the parties agree to make the change to the contract, or
- "by Court", where an appeal to make the change is made to the court by one of the parties, and the court rules in favor of the request.

Thus, we have six notions, from "Termination by Right" to "Modification by Court" (the latter also known as "Reformation"). The view in [MJ16] is that the contract is embodied entirely in the smart contract, so contract modifications need to be applied directly to the smart contract. Features of Ethereum's Solidity language that enable realization of contract changes are identified, specifically,

- contract self-destruct as a means of termination,
- inclusion of a contract parameter that represents its termination state, and use of this parameter in function pre-conditions,
- updates of smart contract parameters, and
- variables of type function, which enable functions implementing a contract provision to be updated by assignment.

In general, these features need to be used very carefully since they potentially undermine the security guarantees that a smart contract was intended to provide. Some code patterns for each of the legal revision types is given in [MJ16], but the arguments made for their correctness are sketchy and informal – we feel this direction deserves further study.

We now consider some of the specific issues raised by rescission and modification of SAFE contracts with respect to smart contract representations of the SAFE and the context of the state of the company as represented on-chain. One general observation relevant to all forms of change is that Miscellaneous provision (a) states that the contract may be amended only with approval of both the SAFE investor and the Company. This indicates that there is not a unilateral right of either party to Rescind or Modify, and all changes must be "by Agreement" or "by Court". We do not attempt here to resolve the issue just raised of how to manage access control so as to ensure that security properties of the smart contract are not undermined by changes to accommodate legal rulings.

When the SAFE contract has not yet converted to shares, its invalidation itself can be implemented by simply removing the `Safe` smart contract from the list of issued SAFEs. In case of modification when the SAFE has not yet converted, a simple substitution of a new `Safe` smart contract suffices.

Matters are more complex when the contract is found to be invalid after the SAFE has already converted, and has had effects on the cap table of the company. Consider the following scenario:

1. An elderly unaccredited investor A purchases a SAFE from startup company C for $200,000 at cap $1M. Company C failed to take any actions to verify that A was an accredited investor.

2. Company C performs well in its startup phase, and attracts an equity investor who contributes $1M at a pre-money valuation of $3M. In the equity round, investor A is issued shares valued at $600,000 at time of the round.

3. A large well-established and well-funded competitor enters the market for company C's product and rapidly steals its market share. The valuation of company C drops to $400,000.

4. Investor A dies and child B inherits the shares. B challenges the validity of the SAFE contract on the grounds that A was senile and unaccredited. The court agrees and rules that the contract was invalid.

In this situation, more needs to happen than the mere deactivation of the `Safe` smart contract. Indeed, the contract has already terminated, and its effects are already reflected in the state of the blockchain. It seems likely that the court would order a monetary compensation to B, and that the shares issued to A be revoked.[37] Making a monetary payment is likely to be within the powers of the

[37]The former is to the detriment of other investors, the latter in their favor, since they

company. However, in general, investors would not want the company to have the power to arbitrarily revoke shares. This issue could be handled by requiring such changes to be approved by the investor directly affected (in this case, B holds the shares, and would be motivated to approve the share cancellation in order to receive the monetary compensation), or to require a signed order from the court. A PKI and certificate reasoning process would be necessary if this is to be automated.

We remark that, in the specific situation of the present paper, a simpler expedient is available to resolve difficulties with amendment of smart contracts if the degree of flexibility required has not been programmed in, or the court ordered changes are blocked by the access control programmed in the smart contract. We have the advantage that the on-chain representations of the cap table are mere data, which are invested with value through an event in the legal domain: the company's declaration that the blockchain representation shall be its true record of the cap table. To effect a change, it suffices for the company to revoke this declaration and declare that henceforth, a smart contract newly created with the amended state serves as the company's cap table. Any unterminated SAFE contracts can similarly be represented as newly created smart contracts when setting up this new representation.[38] (In our implementation, transfer of any monetary value held in the `Company` smart contract is not constrained by the the `Safe_controller`, so is independent of the cap table representation. However, if transfers of money required approval of investors or SAFE holders, the interaction of such controls and legally required modifications to the cap table would require careful consideration.)

Obviously, the possibility of such a change weakens the sense in which the chaincode enforces security properties. In particular, it leaves the investors with only legal guarantees about the integrity of the cap table representation, rather than a strong technical guarantee. However, if there is any dispute about the correctness of the new cap table, with respect to what the Court has ordered, then that can be taken up in the legal domain. It is not within the powers of the company to simply erase the old representation, so there is strong evidence to support any claims of fraud or error on the part of the company.

---

consequently increase their percentage shareholding, but the net effect is unclear. It could be to the detriment of the other investors, for which they in turn may have grounds to claim breach of contract with the company, but such an action would likely be treated as a separate matter.

[38] This approach to dealing with legal rulings is similar to one that has been used to deal with security risks to smart contracts arising from adversarial attacks exploiting programming errors. It is common practice for the operators of a smart contract to control a "kill switch" that deactivates the smart contract and allows them to transfer the assets in the contract to a newly created contract. Here also, the other users of the smart contract are required to trust that the operator will not abuse this power.

12 Related Work

In this section we discuss a number of areas of work most closely related to SAFE contracts and the issues around indeterminacy of legal text that we have addressed. For more general treatments of legal issues in blockchain and smart contracts we refer to a number of recent survey works in [DCP19b].

12.1 SAFT: Simple Agreement for Future Tokens

Simple Agreement for Future Tokens (SAFT) [BBSC17] is a legal contract, modeled on the SAFE, that was developed in 2017 and has been used for raising funds for blockchain projects. The main motivations for the SAFT differ somewhat from the SAFE. Whereas a SAFE converts to shares (a security), conversion of a SAFT is intended to deliver a holding of "tokens", e.g., cryptocurrency or "utility tokens". It was a tenet in the blockchain community at the time that such digital assets represented either money or rights to use software, and should not fall under the purview of securities law. However, the stance of securities regulators was unresolved, creating legal uncertainty for projects financing their development efforts by advance sale of tokens. The aim of the SAFT was to borrow the classification of the SAFE as a security, to give certainty that the financing was conducted in a regulatory compliant fashion as a securities offering, while at the same time hoping to avoid the classification as securities of the tokens ultimately delivered.

SAFTs were not offered as on-chain tokens, but were sold to accredited investors as a paper legal contract, promising future token issuance on a blockchain (typically, the chain to be developed using the invested money). This makes the objectives of the SAFT project very different from our objective, in this paper, of representing SAFE contracts as on-chain smart contracts.

Some significant projects raised funds using the SAFT contract, but the objectives of the SAFT have not been met. Regulators have treated many "utility tokens" as securities, or other forms of regulated investment. Moreover, the US SEC has litigated against some very large financings using SAFTs: Telegram's raise of $US 1.7B and Kik's raise of $US 100M [Byr20]. The courts have accepted the SEC argument that the Telegram raise was a "disguised public distribution" (an artificial two-stage process that was designed to circumvent a prohibited one-stage process in which securities are sold directly to the public) making the SAFT investors non-compliant underwriters of a public offering rather than compliant purchasers of securities. It is therefore unlikely that SAFTs will continue to be used as originally intended. However, this leaves open the use of (on- or off-chain) SAFE contracts for the issuance of securities registered on-chain, our objective in the present paper.

12.2 Blockchain-based Cap Table Management Projects

Public companies often outsource the management of their shareholder records (and functions such as share issuance and transfer, payments and communica-

tions to shareholders) to entities called, depending on the jurisdiction, as "share registries", "share secretaries" or "stock transfer agents". Prominent examples are Computershare and AST. Due the to public nature of their clients, such entities are generally subject to securities-regulator rules.

In general, companies using SAFE contracts are private companies, with relatively few shareholders, that tend to maintain their own cap tables. As private companies grow, their record-keeping and compliance burden increases, and there is therefore also a market for service providers offering cap table management support to private companies. Existing service providers for this sector include Carta [Car] and Certent [Cer].

The above-mentioned service providers do not appear at time of writing to have directly developed blockchain-based services, though some have relationships with entities that are developing permissioned blockchain platforms (e.g., Computershare will connect its services to the Australian Stock Exchange's distributed ledger for clearance and settlement, and has worked with the SETL proprietary blockchain platform [SET].

A number of new players have emerged that are approaching this market from the direction of open standard blockchain platforms. Examples of blockchain-based digital asset, cap table management and investor relations services are Boulevard Global (previously MyStake) [Bou], KoreConX [Kor] and Vertalo [Verb]. or another form of convertible equity instrument. One project that has work in this direction is OpenLaw [Opea], which provides a platform that combines legal and smart contracts by including variables in the legal text whose values are used to instantiate smart contracts. They have given a demonstration of the platform using SAFE contracts [Opeb]. In this demonstration, the four types of SAFE texts are unified by means of parameter settings, and smart contracts are deployed to the Ethereum blockchain using once the contract has been signed by the parties. However, the smart contract code does not appear to contain an encoding of the SAFE contract logic of the kind we have developed in the present work.

12.3 Open Texture in AI & Law

As already noted above, the difficulty posed by the formalization of open-textured concepts has long been an issue recognized in the field of AI & Law [McC80]. The AI & Law literature had an early focus on judgement systems, but has mostly abandoned this to focus on developing systems that provide support for lawyers rather than seek to replace parts of the legal system. Legal support systems may be predictive (how is a case likely to be decided?) or argumentative (what arguments can be advanced for or against a proposition?) Neither objective is particularly useful for our goal of developing a smart contract capturing the intent of a legal contract. Whereas the aim of most AI & Law research is to build artificially intelligent systems with human-like intelligence, which tends towards an interpretative focus on dealing with arbitrary fact patterns, as noted on Section 5.1, in developing smart contracts we have the option of restricting the scope of fact patterns generated so as to simplify

the interpretation of open-textured concepts.

The problem of formalizing open-textured concepts (in Hart's sense) was addressed in [BCS88]. They critically examine several possibilities. One approach, which they called *approximation*, is simply to replace a vague concept by a similar sharp concept. In the context of a legal decision support system, they proposed what is essentially an incremental form of approximation, involving the inductive generation of rules both for and against an object being an instance of the concept, allowing a user to make an informed choice as to whether the object is an instance of the concept. As the authors say

> Resolution of open texture is at bottom a matter of choice, and it is the duty of the user to make that choice

As an aside, we can see analogous considerations concerning automated decision-making. Whatever concept is being adjudicated by automated decisions, there can be cases demonstrating the open texture of the concept. Moreover, when the decision concept concerns people, the "foundational indeterminacy of human identity" [Hil19], appears analogous to Waismann's form of open texture when compared to a sharp concept defined by a profile. Article 22 of the European General Data Processing Regulation (GDPR), expresses a right not to be subject to automated decision-making. We can see this as a continuation of the sentiment of [BCS88] above. [Alm19] proposes building contestability into all stages of an automated decision-making system as an improvement over *post hoc* appeals against decisions.

Other approaches proposed for dealing with open-textured concepts have included "case-based" techniques [Ash92], including machine learning techniques such as clustering and comparing a current case to "similar" cases (e.g. by computing a nearest neighbor to already decided cases according to some metric) [Pop93], use of artificial neural networks [Ben93], and proposals based on prototypes and deformations, subject to a coherence criterion [MS81]. These works address open texture in Hart's sense, but not in Waismann's original sense. They attempt to learn distinctions based on past cases, but the possibility of unexpectedness that is central to Waismann's sense is, by its very nature, not addressable by this approach. The context for work that addresses issues of open texture includes legislation [SSK+86] and case law [San91]. These areas of law do not directly solve issues of smart contract formalization.

More recently, the "Rules as Code" movement has worked on systems providing support for drafting legislation in several countries [Wad21]. The legislation is represented in a form that can then be interrogated for inconsistencies and unintended consequences. This is a tool for analysis of draft legislation, but also potentially a supplement to the legislation that can provide a basis for tools to support navigation of the legislation and other services.

Of the literature that addresses contract formalization [KW05], relatively little appears to deal with indeterminacy. One exception is the work of Daskalopulu [Das99], which applies logical techniques, including Bench-Capon and Sergot's [BCS88] proposal to express open texture concepts using rules to express conditions under which a case can or cannot be included within the scope of a

concept. As decisions made using such rules remain subject to errors and are not sensitive to growth of the body of legal precedent, as noted in Section 9, backing by a legal contract and adaptability to legal rulings remains necessary.

It is interesting to note that the TAXMAN work of McCarty [McC76], applying AI to the interaction of corporate reorganization and tax law, deals with an issue similar to ours with *bona fide* equity rounds: how to interpret a sequence of corporate transactions. (This has also emerged as an issue in the SAFT cases discussed in Section 12.1.) In some cases, the law treats the consequence of a sequence of events differently from the set of consequences of the events individually. Tax law has hinged on questions relating to the *intent* underlying a sequence of corporate transactions, on the basis of which legislation may make allowances with respect to tax consequences that would usually apply. However, as argued above, smart contracts do not need to represent such reasoning on-chain; it suffices that they support amendments arising from legal rulings in which such reasoning has been applied (See Section 11.)

13 Conclusion

We have conducted a detailed investigation of the key Equity Financing clause of SAFE legal contracts, and considered whether, and how, it makes sense to represent such contracts using smart contracts. Our answer to the question of whether SAFE contracts can be smart is a *qualified* "yes".

In the course of the investigation, we identified a number of challenges to this objective, and discussed strategies for dealing with these challenges. Using these strategies, we developed a smart contract architecture for representing a company using SAFEs, which we implemented in the Solidity smart contract programming language. An analysis of the implementation noted some inherent limits to our ability to claim that it fully and correctly implements the legal contract, and identified a particular "attack" against which the legal contract provides protection, but to which the smart contract, on its own, is vulnerable. For this, and other reasons, we recommend that the smart contract be used in conjunction with a legal contract that constrains the way that it is used. We have sketched some of the required content for such a contract, and some of the associated legal issues, but we leave open the development of the (necessarily jurisdiction dependent) text for such a contract.

The work described does not yet provide a full coverage of the SAFE contract, which also covers Liquidity Events and Dissolution events, and has other clauses relating to the validity and interpretation of the contract. Preliminary thoughts on extending our smart contract to cover these parts of the contract are included in Appendix A. A usable decentralized application using our smart contract would need cover these other parts of the SAFE, deal with privacy issues, possibly by use of permissioning on a private blockchain platform, and provide a user interface for interaction with the smart contract. A preliminary analysis of the privacy and platform issues is given in Appendix B.

Some of these topics have been taken up in other works building on the

present paper. A UNSW Bachelors thesis has extended coverage to Dissolution and Liquidity events, deployed the result to a permissioned blockchain and developed a prototype user interface [Cou21]. An analysis of game theoretic issues arising in the context of Dissolution events is provided in [vdM21b].

These contributions leave many issues and improvements required to develop a fully satisfactory smart legal contract framework for convertible instruments, some of which we list here:

- We have assumed that the equity round can be fully described by pre-money valuation, price and share issuances to each investor class. Conceivably, other types of convertible instruments require additional parameters, but it is difficult to predict what these might be. A more abstract formulation of the equity round description, that enables additional parameters to be added by subclassing, might therefore be beneficial.

- The approach described assumes that all SAFEs will convert in the context of an equity round, so that the `SAFE_controller` can be terminated when finalizing the equity round and control can be passed to the `Equity_Round_Swap`. A more subtle management of control is required if unconverted SAFEs or other types of convertible instrument could remain after the equity round, and the `SAFE_controller` needs to be kept in place.

- The criterion we have applied for existence of an equity round, that Preferred shares are being issued could be open to manipulation, where a company colludes with equity round investors to have these accept Common shares instead. In the worst case, this stratagem could be used to forever prevent conversion of the SAFE contracts, depriving the SAFE investors of any benefit from their investments. Since in any event, the SAFE contracts do not come with a guarantee that they will ever be converted, this risk seems inherent in the SAFE contracts, rather than being a fault of our smart contracts.

- We have treated the different types of shares that a company may issue using an enumerated type `ShareType`, without going into the details of the rights associated to different types of shares. It would be reasonable to also attempt to cover these using smart contracts that express these rights.

- We have represented stand-still provisions during an equity round overly simplistically as a block on payments while the `Equity_round_swap` is active. An alternate, arguably more accurate, way would be to represent the term sheet for the equity round as a smart contract, referred to by the `SAFE_controller` and/or the `Equity_round_swap`.

- In principle, when issuing a SAFE contract, it may be necessary to check that the contract is consistent with other contracts signed. (For example, post-money SAFEs give out a proportion of the company, so the set of

proportions needs to add up to less than 1.) Off-chain tools to assist with this task would be useful.

However, if there is a diversity of instruments being used by the company, this check may be computationally complex, and may become undecidable with a rich enough representation format for SAFEs and other convertible instruments. Already in the case of some SAFE's there are valuations at which they cannot be converted [vdMM21], so there are moreover questions to be resolved concerning the meaning of "consistent".

An alternative is to not check, and leave failures of convertibility as a "run-time" issue, to be discovered at the time of the equity round. In case this problem is encountered, the company and investors are likely to want to make a compromise. The approach of verifying consent to the conversion, rather than a full automation, supports such compromises.

- In order to focus on the main issues, we have assumed away a number of aspects of cap tables that are commonly used by startups, such as the option pool, lockups and vesting schedules. A useable implementation would need to provide these.

The abstractions in our framework provide significant flexibility for its users to make choices with respect to the details of the SAFE contract, in particular, the way that it handles the challenges of indefiniteness of the legal text. Ultimately, the market will be final arbiter on the question of whether, and which, SAFE smart contracts deliver the potential benefits discussed in Section 2. In any event, we hope that the present work provides a useful detailed exemplar for issues arising in converting legal contracts to smart legal contracts, as well as a guidepost to work required to develop tooling to support this task.

References

[Alm19] Marco Almada. Human intervention in automated decision-making: Toward the construction of contestable systems. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, ICAIL*, pages 2–11. ACM, 2019.

[Ano11] Anon. Ambiguity. In *Encyclopaedia Britannica*. Horace Everett Hooper, 1911.

[AOT12] Scott Alden, Alex Ottaway, and Jennifer Tetstall. Drafting contracts: guidance on managing ambiguity. `https://www.mondaq.com/australia/contracts-and-commercial-law/163072/drafting-contracts-guidance-on-managing-ambiguity`, 2012.

[Ash92] Kevin D. Ashley. Case-based reasoning and its implications for legal expert systems. *Artificial Intelligence and Law*, 1:113–208, 1992.

[BBSC17] Juan Batiz-Benet, Marco Santori, and Jesse Clayburgh. The SAFT project: Toward a compliant token sale framework. Online `https://saftproject.com/static/SAFT-Project-Whitepaper.pdf`, Oct 2017.

[BCS88] Trevor Bench-Capon and Marek Sergot. Towards a rule-based representation of open texture in law. In Charles Walter, editor, *Computer Power and Legal Language*, page 39–61. Quorum Books, New York, 1988.

[Ben93] Trevor J. M. Bench-Capon. Neural networks and open texture. In Anja Oskamp and Kevin Ashley, editors, *Proceedings of the Fourth International Conference on Artificial intelligence and Law, ICAIL '93, Amsterdam, The Netherlands, June 15-18, 1993*, pages 292–297. ACM, 1993.

[Bla96] Simon Blackburn. *The Oxford Dictionary of Philosophy*. Oxford University Press, 1996.

[Bou] Boulevard global. Online: `https://www.boulevardglobal.com`, last accessed 24 Dec 2021.

[BPW12] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25(1):57–115, 2012.

[Bra84] William W. Bratton. The economics and jurisprudence of convertible bonds. *Wisconsin Law Review*, pages 667–740, 1984.

[Byr20] Preston Byrne. With Kik and Telegram Cases, the SEC Tries to Kill the SAFT. `https://www.coindesk.com/with-kik-and-telegram-cases-the-sec-tries-to-kill-the-saft`, 2020.

[Can04] Andrew J. Cannon. A pluralism of private courts. *Civil Justice Quarterly*, 23:309–323, 2004.

[Car] Carta. Online: `https://carta.com`, last accessed 24 Dec 2021.

[Cer] Certent. Online: `https://certent.com`, last accessed 24 Dec 2021.

[CG15] John F. Coyle and Joseph M. Green. Contractual innovation in venture capital. *Hastings Law Journal*, 66(1):133 – 183, 2015.

[Cha20] Chanticleer. Australia's new investor passport. Australian Financial Review, Oct 2020. Online `https://www.afr.com/chanticleer/australia-s-new-investor-passport-20201008-p5635p`. Accessed 25/10/2020.

[Cla18] C. D. Clack. Smart contract templates:legal semantics and code validation. *Journal of Digital Banking*, 2018.

[Cod] Codelegit. http://codelegit.com/.

[Cou21] William Coulter. *Building Smart SAFEs*. Honours thesis, UNSW School of Computer Science and Engineering, Nov 2021.

[CP13] Christian Colombo and Gordon J. Pace. Recovery within long-running transactions. *ACM Computing Surveys*, 45(3):28:1–28:35, 2013.

[Cra03] Richard Craswell. Instrumental theories of compensation: A survey. *San Diego Law Review*, 40(4), 2003. Available at https://digital.sandiego.edu/sdlr/vol40/iss4/5.

[Das99] Aspassia-Kaliopi Daskalopulu. *Logic-Based Tools for the Analysis and Representation of Legal Contracts*. PhD thesis, Imperial College London, 1999.

[DCP19a] Larry A. DiMatteo, Michel Cannarsa, and Cristina Poncibò, editors. *The Cambridge Handbook of Smart Contracts, Blockchain Technology and Digital Platforms*. Cambridge Law Handbooks. Cambridge University Press, 2019.

[DCP19b] Larry A. DiMatteo, Michel Cannarsa, and Cristina Poncibò, editors. *The Cambridge Handbook of Smart Contracts, Blockchain Technology and Digital Platforms*. Cambridge Law Handbooks. Cambridge University Press, 2019.

[Del19] Deloitte. A roadmap to the issuer's accounting for convertible debt. Online https://www2.deloitte.com/content/dam/Deloitte/us/Documents/audit/ASC/Roadmaps/us-aers-a-roadmap-to-the-issuers-accounting-for-convertible-debt.pdf [Accessed Oct 18, 2019], 2019.

[DLA] DLA Piper. Restrictions on transferability of shares. Online https://www.dlapiperintelligence.com/goingglobal/corporate/index.html?t=38-restrictions-transferability-of-shares. Accessed 28/3/20.

[Fel15] Brad Feld. The pre-money vs. post-money confusion with convertible notes. Online https://feld.com/archives/2015/06/pre-money-vs-post-money-confusion-convertible-notes.html [Accessed Sep 5, 2019], 2015.

[Gri04] Ian Grigg. The Ricardian contract. In *Proceedings of the First IEEE International Workshop on Electronic Contracting*, pages 25–31. IEEE, 2004.

[Gri20] Ian Grigg. Response to the UKJT's Public Consultation on crypotoassets, DLT and Smart contracts under English private law. Online https://iang.org/papers/UKJT-response_on_SmartContracts-IanGrigg.pdf, May-June 2020.

[Hag] Hague convention on the law applicable to certain rights in respect of securities held with an intermediary. Online `https://www.hcch.net/en/instruments/conventions/full-text/?cid=72`.

[Har58] H.L.A. Hart. Positivism and the separation of law and morals. *Harvard Law Review*, 71(4):593–629, 1958.

[Har61] H.L.A. Hart. *The Concept of Law*. Clarendon Law Series. Clarendon Law, 1961.

[Har17] Oliver Hart. Incomplete contracts and control. *American Economic Review*, 107(7):1731–1752, 2017.

[Her18] M. Herlihy. Atomic cross-chain swaps. In *Proc. ACM Symp. on Distributed Computing*, 2018. Version at `arXiv:1801.09515`.

[Hil19] Mireille Hildebrandt. Privacy as protection of the incomputable self: From agnostic to agonistic machine learning. *Theoretical Inquiries in Law*, 20:121 – 83, 2019.

[Ins81] American Law Institute. *Restatement (Second) of Contracts*. American Law Institute, 1981.

[JKL+20] J. Jiao, S. Kan, S. Lin, D. Sanan, Y. Liu, and J. Sun. Semantic understanding of smart contracts: Executable operational semantics of solidity. In *IEEE Symposium on Security and Privacy (SP)*, pages 1265–1282, 2020.

[Kle] Kleros. `https://kleros.io/`.

[Kor] Koreconx. Online: `https://www.koreconx.com`, last accessed 24 Dec 2021.

[Kuh14] Tobias Kuhn. A survey and classification of controlled natural languages. *Computational Linguistics*, 40(1):121–170, March 2014.

[KW05] Steven O. Kimbrough and D.J. Wu, editors. *Formal Modelling in Electronic Commerce*. Springer, Berlin, Heidelberg, 2005.

[Lak76] Imre Lakatos. *Proofs and Refutations*. Cambridge University Press, Cambridge, UK, 1976.

[Les00] Lawrence Lessig. Code is law, on liberty in cyberspace. *Harvard Magazine*, 2000. Online `https://www.harvardmagazine.com/2000/01/code-is-law-html`.

[Lev17] Karen E.C. Levy. Book-smart, not street-smart: Blockchain-based smart contracts and the social workings of law. *Engaging Science, Technology, and Society*, 3:1–15, 2017.

[Mac63] Stewart Macaulay. Non-contractual relations in business: a preliminary study. *American Sociological Review*, 28(1):55–67, 1963.

[McC76] L. Thorne McCarty. Reflections on TAXMAN: An experiment in artificial intelligence and legal reasoning. *Harvard Law Review*, 1976.

[McC80] L. Thorne McCarty. Some requirements for a computer-based legal consultant. In Robert Balzer, editor, *Proc. of the 1st Annual National Conference on Artificial Intelligence, Stanford University, CA, USA, August 18-21, 1980*, pages 298–300. AAAI Press/MIT Press, 1980.

[MJ16] Bill Marino and Ari Juels. Setting standards for altering and undoing smart contracts. In *Rule Technologies. Research, Tools, and Applications - 10th Int. Symp., RuleML, Proc.*, volume 9718 of *Springer LNCS*, pages 151–166, 2016.

[Moo03] Michael Moore. For what must we pay? causation and counterfactual baselines. *San Diego Law Review*, 40(4), 2003. Available at `https://digital.sandiego.edu/sdlr/vol40/iss4/6`.

[MS81] L. Thorne McCarty and N. S. Sridharan. The representation of an evolving system of legal concepts: II. prototypes and deformations. In *Proc. of the 7th International Joint Conference on Artificial Intelligence, IJCAI*, pages 246–253, 1981.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Available at `https://bitcoin.org/bitcoin.pdf`, Nov 2008.

[NVC] National Venture Capital Organisation NVCA. Model legal documents. Online. `https://nvca.org/model-legal-documents/`. Accessed 5/4/2020.

[Opea] Openlaw. Online: `https://www.openlaw.io`, last accessed 24 Dec 2021.

[Opeb] Openlaw safe video demonstration. Online: `https://www.youtube.com/watch?v=ySIsxEl5yME`, last accessed 24 Dec 2021.

[Opec] OpenZeppelin. Documentation: Ownership. Online `https://docs.openzeppelin.com/contracts/2.x/api/ownership#Ownable`. Accessed 25/3/20.

[PD19] Cristina Poncibò and Larry A. DiMatteo. Smart contracts: Contractual and noncontractual remedies. In DiMatteo et al. [DCP19b], page 118–140.

[Pop93] James David Popple. *SHYSTER: A pragmatic legal expert system,*. PhD thesis, Australian National University, 1993.

[Pop16] Nathaniel Popper. Hacker may have taken $50 million from cybercurrency project. The New York Times, 17 June 2016. See also http://en.wikipedia.org/wiki/The_DAO_(organization).

[Rai14] Gregory Raiten. 500 startups announces KISS. Online. https://500.co/kiss/, Jul 2014. Accessed 2/5/20.

[RM20] Sebastián A. Reyes Molina. Judicial discretion as a result of systemic indeterminacy. *Canadian Journal of Law & Jurisprudence*, 33(2):369–395, 2020.

[San91] Kathryn E. Sanders. Representing and reasoning about open-textured predicates. In *Proc. of the Third Int. Conf. on Artificial Intelligence and Law*, pages 137–144, 1991.

[SC] U.S. Securities and Exchange Commission. Investor bulletin: Be cautious of safes in crowdfunding. Online, https://www.sec.gov/oiea/investor-alerts-and-bulletins/ib_safes, last accessed 29 Dec 2021.

[Sch02] Sanford Schane. Ambiguity and misunderstanding in the law. *T. Jefferson Law Review*, 25(1):167–194, 2002.

[Sch13] Frederick Schauer. On the open texture of law. *Grazer Philosophische Studien*, 87(1):197–215, 2013.

[SET] Setl. Online: https://setl.io, last accessed 24 Dec 2021.

[SSK+86] Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, F. Kriwaczek, Peter Hammond, and H. T. Cory. The British Nationality Act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.

[Sza97] Nicholas Szabo. The idea of smart contracts. https://nakamotoinstitute.org/the-idea-of-smart-contracts/, 1997.

[US 17] US Securities and Exchange Commission. SEC issues investigative report concluding DAO tokens, a digital asset, were securities. Online https://www.sec.gov/news/press-release/2017-131, Jul 2017.

[vdM19] R. van der Meyden. On the specification and verification of atomic swap smart contracts. Online: https://arxiv.org/abs/1811.06099, 2019. Abstract appears in IEEE International Conference on Blockchain and Cryptocurrency, 2019, pp. 176–179.

[vdM21a] R. van der Meyden. A formal treatment of contract signature. *IEEE Transactions on Services Computing*, 2021.

[vdM21b] R. van der Meyden. A game theoretic analysis of liquidity events in convertible instruments. arXiv https://arxiv.org/abs/2111.12237, Nov 2021.

[vdMM21] R. van der Meyden and M. J. Maher. Simple agreements for future equity – not so simple? manuscript, http://www.cse.unsw.edu.au/~meyden/research/SAFEnss.pdf, 2021.

[Vera] Online https://www.verifyinvestor.com, Accessed 15/10/2020.

[Verb] Vertalo. Online: https://www.vertalo.com, last accessed 24 Dec 2021.

[Wad21] Matthew Waddington. Rules as code. *Law in Context*, 37(1):179–186, 2021.

[Wai68] F. Waismann. Verifiability. In R. Harré., editor, *How I See Philosophy*. Palgrave Macmillan, London, 1968.

[Wit53] Ludwig Wittgenstein. *Philosophical Investigations*. MacMillan Publishing Co., first edition, 1953.

[Y C] Y Combinator. *Startup Documents 2016*.

[Y C16a] Y Combinator. *Safe: Cap, no Discount*, https://web.archive.org/web/20180831020232/http://www.ycombinator.com/docs/SAFE_Cap.rtf 2016.

[Y C16b] Y Combinator. *SAFE Primer*, https://web.archive.org/web/20180831020232/http://www.ycombinator.com/docs/SAFE_Primer.rtf 2016.

[Y C18a] Y Combinator. *Post Money Safe User Guide*, https://www.ycombinator.com/docs/Post%20Money%20Safe%20User%20Guide.pdf 2018.

[Y C18b] Y Combinator. *Safe: Valuation Cap, no Discount*, https://web.archive.org/web/20190626002912/https://www.ycombinator.com/docs/Postmoney%20Safe%20-%20Valuation%20Cap%20-%20v1.0.docx 2018.

A Other Aspects of the SAFE

The primary focus of the body of the paper was to develop an implementation of the key part of the Equity Financing clause of SAFE contracts. In this appendix our aim is to determine the extent to which it would make sense to extend the implementation to cover other parts of the SAFE. Where appropriate, we discuss the structure of an implementation, but we leave the development of code for any of these components to future work. Unless otherwise indicated, our discussion in this section is based on the Pre-Money SAFE with cap and no discount, but we note some significant variances in the Post-Money versions of the SAFE where appropriate. Section A.6 deals with "Most Favoured Nation" SAFEs, which introduce some additional issues.

A.1 Execution of Equity Financing Documents

Section 1a(i) of the SAFE contract (see Section 3), requires that the SAFE investor "will execute and deliver to the Company all transaction documents related to the Equity Financing". (We have already noted in Section 4.3 that this potentially gives the SAFE investor some power to disrupt the equity round by withholding their signature, consequently leading to the SAFE investor obtaining an ability to negotiate on the construction of the round.) The documents related to the Equity Financing may be quite complex and may deal with matters such as reincorporation, revisions to be made to the company's governance structure on completion of the round, shareholder rights after the round, and indemnities and procedures relating to conduct of the round. Model documents for an equity round have been proposed by the (USA) National Venture Capital Association [NVC].

Note that this clause states an *obligation* on the SAFE investor, to sign and return the Equity Financing documents. Implicitly, there is a related obligation on the company, to notify the SAFE investor of the Equity Financing and to provide the documents requiring signature. These actions may be essential to the legal status of the round and the obligations exist irrespective of whether the SAFE contract is interpreted as granting negotiation rights to the SAFE investor in the structure of the Equity Financing. In the case of the company's obligation to issue shares to the SAFE investor, it was feasible to enforce the obligation through a smart contract. By contrast, neither the obligation on the SAFE investor to sign the Equity Financing documents, nor that on the company to notify the investor and provide these documents, can be enforced by a smart contract.

In the case of the investor, this is because the contents of the documents to be signed are open ended and not predictable at the time of the issuance of the SAFE contract, so it should be left to the investor to make the decision to sign. The investor may have sound reasons not to sign, for example, if there are reasons to believe that the round is not *bona fide*, but has been constructed to deprive the SAFE investors of rights (see Section 9.3). Even if the signature is digital, there are technical difficulties in having a smart contract control the investor's signature key and sign on the investor's behalf. While there exist *proxy signature* schemes whereby a user is able to delegate the ability to sign messages to another [BPW12], implementing these on blockchain platforms where the contents of the smart contract are accessible to potentially untrustworthy validators (on permissioned blockchains), or even public (on open blockchains), would leak the delegate's signature key and enable parties other than the delegate to create a signature on behalf of the SAFE investor.[39]

In the case of the company, its obligation to notify the SAFE investor and

[39] It may be possible to avoid this problem by means of a legal interpretation of such delegate signatures as being valid only if they can be proved to have been created by execution of the smart contract, together with a definition of the signature verification function that checks this property against the blockchain. However, in this case it is not clear that use of a sophisticated proxy signature is actually required, since simpler properties of the behaviour of the smart contract may serve equally well. We leave exploration of this issue for future research.

provide the Equity Financing documents cannot be enforced by a smart contract since the contents of these documents will be negotiated with the new investors, so cannot be predicted at the time of creation of the smart contract for the SAFE. The SAFE investors would obtain notification of the equity round if they (or a delegate[40]) are constantly observing the blockchain, and the company calls the `start_equity_round` function, which posts the parameters of the equity round on-chain. However, this still does not mean that the company's implicit obligation to notify the SAFE investor of the Equity financing is enforced, since the company may potentially conduct the equity round off-chain without notifying the SAFE investor. A legal agreement between the company and the SAFE investor is therefore required to ensure that the company does not conduct the equity round off-chain.

One approach for the company to provide the Equity Financing documents to the SAFE investor would be to include them in an additional parameter of the `start_equity_round` function. (Note that this does not imply that the obligation on the company to provide the documents is being enforced, since the company could still include an empty document in this field.) If approval of the equity round proposal by the `Safe` contract requires that the investor has consented to the round, the need for the investor's signature could then be eliminated if there is a legal agreement concerning operation of the smart contract that includes a clause stating that the SAFE investor granting their consent to the equity round will be interpreted as the investor assenting to the terms of the round expressed in these documents. On the automated conversion approach, the need for the SAFE investor's signature would remain. (On chains like Ethereum that charge for memory used by transactions, including large documents in the parameters of a transaction may be expensive or impossible because of transaction size limits, in which case it will be more efficient to include their cryptographic hash. There would then remain an obligation on the company to provide the Equity Financing documents to the SAFE investor in a form that would enable them to check the hash.)

A.2 Pro Rata Rights

Pro Rata Rights (PRR) are rights of an investor to purchase shares in a future equity round, intended to protect the investor from being diluted by future share issuance. PRR typically give the investor the option but not an obligation to purchase additional shares. In practice, only a fraction of the PRR shares need to be purchased.

In the Pre-Money SAFE, the Equity Financing clause obliges the company to issue PRR for the *next round* to the SAFE investor at the time of conversion, but this clause is replaced in the Post-Money SAFE by an optional side agreement (negotiable between the SAFE investor and the company) that grants the SAFE investor PRR for the round in which the SAFE converts.

[40] Service providers who maintain watch on the blockchain and inform users of particular on-chain events are called "watchtowers" in the blockchain community.

The Pre-Money SAFE indicates the PRR are determined using *percentage basis*, which grants a right to maintain the percentage shareholding.[41] This is generally achieved by the company offering a fixed number of shares in the equity round, of which the PRR holders has rights to take up a percentage less than or equal to their percentage holding, with the remaining shares going to the new investors.

PRR could be enforced in equity rounds conducted on-chain, using a smart contract for the PRR, together with an equity round atomic swap contract and controller smart contract similar to that in the existing code. When the equity round proposal is submitted by the company, the controller first checks that it contains options for each of the PRR holders, consistent with the rights encoded in their PRR smart contracts. The round proposal is rejected if this is not the case. Otherwise the equity round atomic swap smart contract is created. The PRR holding investors first interact with this swap contract by paying for the fraction of their PRR that they will take up, followed by consent and payment by the new investors. (A new deadline needs to be added for the PRR stage, to ensure the new investors get adequate time to meet the ultimate deadline.)

However, this approach does not take into account that PRR may create tensions between new investors and those holding PRR. Sometimes the new investors want existing shareholders to exercise their PRR rights to show commitment to the company and validate the new investors' valuation. In other cases, the new investor prefer that these rights are not taken up so that they more cheaply attain their desired stake in the company.

Some richer schemes than the above could also be implemented as a way to deal with these tensions between PRR holders and new investors. For example, making PRR transferrable would enable new investors to compensate PRR holders for giving up their rights. (However, it should be noted that this may conflict with share transfer limitations intended to limit changes of control.)

A.3 Definitions and Other Events: Liquidity, Dissolution and Termination

In addition to the Equity Financing clause, the SAFE specifies investor rights in case of two other types of events: Liquidity Events and Dissolution Events, provided these occur before the SAFE has terminated. (The SAFE terminates when any of the three types of events—Equity Financing, Liquidity or Dissolution—occurs.) The Definitions section defines terms used used in the statements of the Events Clauses; we discuss relevant definitions in this section together with the events.

Liquidity Events are defined to occur if there is a "Change of Control" or "Initial Public Offering". The definition of these concepts contains a significant number of open-textured terms and references to legislation outside of the scope

[41] Alternatives in use are the *dollar for dollar basis* (invest up to the original principal in the next round) and the *fixed sum* right (invest up to a fixed dollar amount). The latter are less common since they create tension by enabling the PRR holder to strictly increase their percentage holding in some circumstances.

of the SAFE. The events that may be classified under these terms may also exhibit significant open structure. For example, condition (i) of the definition of "Change of Control" refers to

> a transaction or series of related transactions in which any "person" or "group" (within the meaning of Section 13(d) and 14(d) of the Securities Exchange Act of 1934, as amended), becomes the "beneficial owner" (as defined in Rule 13d-3 under the Securities Exchange Act of 1934, as amended), directly or indirectly, of more than 50% of the outstanding voting securities of the Company having the right to vote for the election of members of the Company's board of directors ...

This raises an issue similar to that noted in Section 4.2 for the Equity Financing clause: which transactions should be counted as part of the series of transactions? The structure of the transactions is open ended and cannot be predicted in advance. Will these transactions be recorded on-chain? Even if so, we are likely to find further open texture in the text of the legislation. Moreover, the fact that the definition encompasses amendments to the legislation means that its interpretation is dynamic and cannot be statically coded. An on-chain implementation of the SAFE that checks this condition may need either a reference to an on-chain encoding of the Act, or to support a secure update of the coding of the definition in case the Act is amended.

The other conditions are equally problematic. For example, condition (ii) states that any "any reorganization, merger or consolidation of the Company", other than one in which the holders of shares in the company prior to the transaction preserve majority control, counts as a Liquidity event. The structure of such events is again completely open. Finally, condition (ii) states that a Liquidity event occurs in case of "a sale, lease or other disposition of all or substantially all of the assets of the Company". The term "substantially all" here is open-textured, and an automated implementation of this condition would require that we represent the entirety of the company's books on-chain. This suggests that an attempt to automate the detailed conditions for a Liquidity Event is likely to face significant difficulties, and that it is better to apply the strategy of treating the declaration of the conditions for a Liquidity event as an atomic event, with appropriate controls.

The consequences of a Liquidity event are that the SAFE investor has the option to

> either (i) receive a cash payment equal to the Purchase Amount (subject to the following paragraph) or (ii) automatically receive from the Company a number of shares of Common Stock equal to the Purchase Amount divided by the Liquidity Price, if the Investor fails to select the cash option.

The time in which the investor must make the choice is unspecified. Remarks similar to those made in Section A.1 concerning the need for a legal agreement to ensure that a timely notification occurs apply here.

A further paragraph of the Liquidity Event clause states that if there are insufficient funds to pay out all SAFE investors who choose that option, then the available funds are shared *pro rata* among all such SAFE investors, with shares in the company to make up the balance. Automation of this condition requires that all SAFEs are recorded on-chain, and all available funds are under control of the smart contract. A legal agreement is required to ensure that this is the case. There is a proviso that in case of a "tax free reorganisation", the amounts of cash payable may be reduced as determined by the board "in good faith". This proviso is therefore open-textured, and again depends on complex and dynamic external legislation (the tax code), suggesting that it is best handled by a coarse-grained on-chain declaration under appropriate control.

Dissolution Events are defined to be

> (i) a voluntary termination of operations, (ii) a general assignment for the benefit of the Company's creditors or (iii) any other liquidation, dissolution or winding up of the Company (excluding a Liquidity Event), whether voluntary or involuntary.

In the case of such an event, the SAFE specifies that "the assets of the Company legally available for distribution ... as determined in good faith by the Company's board of directors" will be distributed to SAFE investors in priority to all stockholders, up to the amount of the Purchase amount, and pro-rated if insufficient. The conditions for occurrence of a Dissolution event, and the term "good faith" here are open-textured, so again best implemented by a simple declaration, with appropriate controls and backing by a legal agreement. To implement the clause on-chain would also require that all the cash or other assets are recorded on-chain before the distribution is made. A legal agreement is required to ensure this.

The SAFE's requirement to give priority to SAFE holders could conflict with other forms of contract that might have been issued prior to issuance of the SAFE. Absence of such a conflict is covered by a representation of the company that its performance of the SAFE is consistent with other contracts issued (see Section A.4). The process for issuance of a SAFE in our code does not check for consistency of the SAFE with other contracts, indeed, this is likely to be undecidable. The code is best interpreted as correct only when the company issues only SAFEs. This could be expressed in a legal agreement, and is an instance of the Limitation of Fact Patterns strategy of Section 5.1.

A.4 Company and Investor Representations

The sections "Company Representations" and "Investor Representations" of the SAFE relate to the factual background of the contract, and concern matters that underpin its legal validity. Generally, the representations relate to matters that are not likely to be represented in detail on-chain, and are probably best left as natural language text in a legal contract to be signed by the parties.

(Section 10.3 discusses interactions between signatures on legal documents and smart contract transactions.)

Should any of the representations made by either the company or the investor be false, legal grounds may exist for the other party to have the contract declared void and claim damages, although they may also choose to renegotiate its terms. Section 11 discusses issues of amendment of smart contracts and handling of legal rulings in a broader context.

More specifically, the company represents the following (we summarize text of the SAFE and comment on the possibility of automation) :

- That the company is properly registered, in good standing and has the power to operate its properties. Some of this information may (in future) become available on a variety of blockchain systems in some jurisdictions, but oracles and/or cross-chain information transfer may be necessary for any automation.

- That the company has the power to execute and perform the SAFE contract: it has been duly authorized and is consistent with the by-laws, statutes, violations or legal actions against the company, or other contracts involving the company. These are likely to be broadly open-textured, so offer only limited opportunity for automation. Verifying consistency with other contracts is likely to be an undecidable problem, so not amenable to automation.

- The company's performance of the contract will not violate any statutes or regulation or result in adverse legal consequences for the company. (Impediments to automation are similar to the previous point.)

- The only consents required for performance of the contract are those internal to company or by securities law. Again, this requires knowledge of the external legislation and regulatory environment, and is not easily amenable to automation.

- The company holds IP rights (trademarks, copyright, patents etc.) necessary to conduct its business. (This depends on the declaration by the company being not just correct but also complete, so is best handled by a legal agreement rather than online verification.)

The SAFE investor representations, in brief, are:

- That the investor has the legal capacity to enter into the contract, accepts its obligations, with exceptions for bankruptcy or other credit and equity laws.

- That the investor is an accredited investor (as defined by the applicable jurisdiction) and understands the risks involved in the SAFE and that the SAFE and the shares to be issued have not been registered and are not transferrable until registration or exempted from registration.

The conditions for these representations are likely to be broadly open-textured, and are best handled in a legal contract. Equities law and regulation may limit the extent to which such representations may be relied upon by the company and place obligations on companies to verify investor accreditation. For example, in the USA, companies are required to verify investor accreditation status, with Safe Harbor provisions stating sufficient conditions for a company to have complied with its verification obligations. There are some moves in some jurisdictions to standardize investor accreditation information and make it available on blockchain (e.g., in Australia there has been recent discussion of the use of Tax Office identifiers as a basis for this, motivated by the Australian Stock Exchange blockchain project amongst others [Cha20] and in the USA, VerifyInvestor, a subsidiary of the tZero blockchain-based security trading platform, offers securities law compliant attestations of investor accreditation [Vera]), so the representations concerning accreditation could be replaced by on-chain evidence. However, matters such as the investor's acceptance of obligations and limitations of the SAFE and their understanding and acceptance of its risks and limitations requires the investor's signature, which is best placed on natural language text in a legal contract.

A.5 Miscellaneous Provisos

The Section "Miscellaneous" of the SAFE contract contains provisos dealing with a diverse range of matters concerning operation and interpretation of the contract.

Proviso (a) states that "Any provision of this instrument may be amended, waived or modified only upon the written consent of the Company and the Investor." From the point of view of smart contract implementations of any part of the contract, this implies that the implementation needs to admit amendments that may occur. We discuss this issue in a broader context in Section 11.

Proviso (b) concerns sufficient conditions for notice required by the SAFE, stating that methods such as courier, email and registered mail are adequate. When implementing the SAFE using a smart contract, this clause could be revised to indicate the conditions for adequate communication via on-chain events. See Section A.1 for related discussion.

Proviso (c) states that certain rights are *not* implied by the (Pre-Money) SAFE, specifically, the rights to vote or receive dividends,[42] or any other rights of a shareholder (including the right to receive notice of meetings) until shares have been issued. Since none of these rights are explicitly mentioned in the remainder of the SAFE, this clause could be considered redundant, but it acts to preempt legal interpretations of such rights as implicit in the SAFE. Beyond excluding from the code implementations for such interpretations, this proviso does not have any effect on the smart contract.

Proviso (d) is similarly negative and states that the SAFE, and any associated rights of either party, are not transferrable without approval of both parties,

[42]The Post-Money SAFE alters the terms by granting rights to receive dividends.

with some exceptions for transfers to entities controlled by the SAFE investor, and rights of the company to make transfers in relation to reincorporation or changes of domicile. The set of entities related to the SAFE investor and the structure of ownership and control is open ended and could be complex, so it is best to handle the exceptions in a coarse grained way, leaving verification of the conditions to the company. Transfers in relation to reincorporation are discussed in Section 11.

Proviso (e) states that in case any part of the SAFE is deemed invalid, the remainder of the SAFE will remain operative. With respect to a code representation of the SAFE, this suggests that there should be a close correspondence between the clauses of the SAFE and the smart contract representation. Given the large divergence of structure between the legal text and its representation in imperative code, it is inherently difficult to achieve this. Thus, the most reasonable means of dealing with this proviso is to create a new smart contract and substitute it for the old. This could be achieved by extending the `SAFE_controller` to allow for such substitutions. However, as other investors may have made their investment decisions based on the contract being revised, careful consideration should be given to governance of this process, to prevent it being used maliciously to make changes that adversely affect the other investors.

Proviso (f) states the jurisdiction governing the SAFE contract "without regard to the conflicts of law provisions" (meaning that the parties are prevented from arguing that the jurisdiction may identify a conflict of laws and apply law from another jurisdiction). This is largely a matter for the legal contract with few interactions with the smart contracts, except possibly where the jurisdiction is an on-chain arbitration service (see Section 10.5 for discussion of a number of approaches to this).

A.6 Most Favored Nation SAFEs

The Pre-Money and Post-Money SAFEs with neither cap nor discount but with MFN clause, have largely identical terms with respect to the Equity Financing clause, except that the Post-Money SAFE detaches the Pro Rata Rights conditions. The conversion clause says that the SAFE investor is converted at the same price as the Preferred shares (or, lowest price of Preferred shares, in the Post-Money version):

> **Equity Financing.** If there is an Equity Financing before the expiration or termination of this instrument, the Company will automatically issue to the Investor a number of shares of Preferred Stock sold in the Equity Financing equal to the Purchase Amount divided by the price per share of the Preferred Stock.

However, this is a default, and an "'MFN" Amendment Provision' could result in modification of this price if the Company issues "Subsequent Convertible Securities".

> **"MFN" Amendment Provision.** If the Company issues any Subsequent Convertible Securities prior to termination of this Safe, the

> Company will promptly provide the Investor with written notice thereof, together with a copy of all documentation relating to such Subsequent Convertible Securities and, upon written request of the Investor, any additional information related to such Subsequent Convertible Securities as may be reasonably requested by the Investor. In the event the Investor determines that the terms of the Subsequent Convertible Securities are preferable to the terms of this instrument, the Investor will notify the Company in writing. Promptly after receipt of such written notice from the Investor, the Company agrees to amend and restate this instrument to be identical to the instrument(s) evidencing the Subsequent Convertible Securities.

"Subsequent Convertible Securities" (henceforth abbreviated to SCS) is defined by

> **"Subsequent Convertible Securities"** means convertible securities that the Company may issue after the issuance of this instrument with the principal purpose of raising capital, including but not limited to, other Safes, convertible debt instruments and other convertible securities. Subsequent Convertible Securities excludes: (i) options issued pursuant to any equity incentive or similar plan of the Company; (ii) convertible securities issued or issuable to (A) banks, equipment lessors, financial institutions or other persons engaged in the business of making loans pursuant to a debt financing or commercial leasing or (B) suppliers or third party service providers in connection with the provision of goods or services pursuant to transactions; and (iii) convertible securities issued or issuable in connection with sponsored research, collaboration, technology license, development, OEM, marketing or other similar agreements or strategic partnerships.

A number of issues are raised when attempting to formalize these clauses. The notification requirement raises issues similar to those already discussed in Section A.1. open-textured concept are again an issue: the definition of SCS contains many expressions that do not have precise definitions: e.g.,"convertible securities","principal purpose of raising capital", and"other Safes". Similarly, the MFN Amendment Provision uses imprecise expressions such as "promptly" and "may reasonably be requested".

A further difficulty is the interpretation of the phrase "amend and restate this instrument to be identical to the instrument(s) evidencing the SCS". It is not completely clear what is meant by this, and"identical" is unlikely to be precisely what is meant, since at least the principal and beneficiary of the SCS are likely to differ in SCS and the new instrument to be granted to the SAFE investor. Substituting these parameters of the MFN SAFE for those in the SCS is likely to be involved in the amendment. However, it may not be clear what the corresponding parameter of the SCS is. For example, what corresponds to the principal parameter if the new instrument involves a (contingent) sequence

of principal payments delivered over time? Other parameters of the SCS may not correspond to any parameter of the SAFE. For example, what if the SCS specifies a proxy of the SCS investor for some actions, or a designated party for arbitration of disputes? Further, what if the jurisdiction differs? Should this be modified to the jurisdiction of the SAFE? This might not make sense, if some of the terms of the SCS were constructed with a specific alternate jurisdiction in mind.

If the modification is to be automated, it would seem that the smart contracts for the Subsequent Convertible Securities that the company is permitted to issue need to be constrained to smart contracts with a known interface, that enables the modification to be constructed by substitution of a limited set of parameters. A further issue is that the source code of the smart contract needs to be available, so that it can be checked that it will behave as expected by the investor. On the Ethereum chain, only the bytecode will be available, which may make it difficult to determine automatically that the code is for the required interface, and to make the required substitution.

If the smart contract is paired with a legal agreement, the latter needs in any case to be provided to the SAFE investor off-chain, so source code for the smart contract can be provided to the SAFE investor as part of this communication. The SAFE investor will want to examine both to determine whether or not to notify the Company that they are taking up their option to switch to the new instrument. Some negotiation may be necessary to come to an agreement concerning the appropriate modification. This makes it reasonable to implement the on-chain activity using an on-chain proposal and acceptance for the modified contract (typically only the final agreement would be put on-chain). Either party could be the proposer, since both would need to agree.

B Privacy and Platform Issues

Smart contracts can be implemented on a variety of platforms that differ in their approaches to privacy. In this appendix, we consider the appropriateness of a number of platform approaches for SAFE smart contracts.

B.1 Policy

Privacy Policy concerning company cap tables is jurisdiction dependent. Table 1 gives an example of a privacy policy for the cap table data held by a private company. The policy assumes that the cap table itself stores data concerning ownership of shares and SAFEs using investor pseudonyms, which can be linked to investor personal information (name and address) in a separate table. The company itself is permitted to access all information, e.g., it needs shareholder identity for communications with shareholders, as well as regulatory compliance. Regulators may need detailed shareholder information in order to support policing of rules concerning, e.g., foreign ownership and money launder-

Visible to:	Company	Investor	Regulator	Public
Investor Name	✓	*	✓	×
Investor Address/email	✓	*	✓	×
Investor Psuedonym	✓	✓	✓	×
Shares Issued	✓	✓	✓	×
SAFEs Issued	✓	✓	✓	×

Table 1: Privacy Policy of Cap Table Data (Private Company)

ing.[43] As the company is private, the general public is not necessarily entitled to any information about the company. Individual investors (i.e., shareholders, SAFE investors and potential new investors from whom funding is being sought) have an interest in understanding the broad structure of the cap table, e.g., for purposes price determination in the event of an equity round, only the total number of shares and SAFEs, and their detailed types, are needed. The cap table could also be released to them with shareholder identities masked using pseudonyms, which could give some information about concentration of control. Knowledge of pseudonyms for investors could allow anonymity of some shareholder data to be breached, e.g., the founders might be easily identified as the largest shareholders. However, founder shareholding is information that the company might release in any event since it addresses investor questions concerning founder incentivization. There may also be circumstances in which it is necessary to release information to investors about other investors, e.g., in case they need to negotiate with each other concerning the details of an equity round. The table acknowledges this by using $*$ to indicate permission to access data in some circumstances.

B.2 Public blockchains

Public blockchains, e.g., Bitcoin and Ethereum, are open for use by any person, who may operate validator nodes or simply participate by contributing transactions. Users are represented using pseudonymous identities (public keys). Each user may have many pseudonyms. All data recorded on the blockchain is publicly visible – this includes bytecode of smart contracts.

The Solidity smart contracts we have developed similarly represent investors using pseudonyms. This allows the cap table data to be split as a relation (Name,Address, Pseudonym), stored by the company off-chain, and a relation (Psuedonym, SAFE issued, Shares issued), stored on-chain in the `Company` smart contract. This would yield the visibility table for the implementation given in Table 2. (A salted hash of personal information could be added to the smart contract to enable integrity verification of this information, while keeping it

[43] For example, in Australia, proprietary companies are required to keep the Australian Securities and Investment Commission informed about their top 20 shareholders. The shareholder register is only semi-private, since the public is entitled to inspect it, although the company may charge an inspection fee to non-shareholders.

Visible to:	Company	Investor	Regulator	Public
Investor Name	✓	×	✓	×
Investor Address/email	✓	×	✓	×
Investor Psuedonym	✓	✓	✓	✓
Shares Issued	✓	✓	✓	✓
SAFEs Issued	✓	✓	✓	✓

Table 2: Visibility of Cap Table Data (Ethereum Implementation)

private.)

This is close to satisfying an instance of the policy in which investors are permitted to observe investor pseudonyms. However, more information is visible to the Public than is permitted by the policy. If making pseudonymous cap table information available to the public is considered acceptable, this may be an acceptable implementation.

Moreover, when there is a legal contract for a SAFE, and a digitally signed copy of this is placed in a field of the `Safe` smart contract, then the investor identity would be leaked, since it is recorded in the contract. Storing only the digital signature in the smart contract, and including random salt in the text, would avoid this, but this would also prevent the smart contract from verifying that the correct text has been signed. In place of the parties names, their pseudonyms could be used in the legal contract text, but this would require that legal contracts in this form to be considered valid in law. Certified data linking the pseudonyms with real identities would also need to be accessible to the legal system, but not to the general public, to make sense of this idea.

We remark that in the event there is a need for arbitration or litigation concerning any matter, the dispute is most likely to be between an investor and the company, since the investors do not have any direct legal relationships. The company will have identifying information of investors for compliance reasons, and the investor will also be able to identify the company and its key officers, assuming the investor has undertaken appropriate due diligence.[44] This implementation is therefore able to avoid the difficulty of applying legal enforcement against an anonymous party.

The cost of using a public blockchain as above is that more information is released to the public than may be desirable. In order to obtain an implementation in a public blockchain setting that more precisely implements the policy in Table 1, it may be possible to use zero-knowledge proof techniques. We leave an exploration of how this might be done, while preserving the required integrity guarantees for the cap-table and execution of the SAFE smart contracts, for elsewhere.

[44] Thus, even though investors may be anonymous to each other in this implementation, the expected setting differs from that of many Initial Coin Offerings on open chains during the speculative ICO boom of 2016, in which all parties, including the company were anonymous to each other. This model, not surprisingly, was associated with high levels of fraud.

B.3 Private blockchains

Private blockchains (also called permissioned, or consortium chains), differ from public chains in that they restrict the right to be a validator to a known set of parties. In this case, it is more efficient for the validators to use a classical byzantine consensus protocol, rather than a protocol based on "proof of work" or "proof of stake", as with open chains. Additionally, access control may also be applied to (non-validator) parties contributing transactions: on-boarding of such parties can be restricted, and they can be required to provide identifying information as part of the on-boarding process. Their interactions with the blockchain data can moreover be restricted to a particular view depending on their identity or role in the system. (Validators still observe the complete state of the blockchain, and are responsible for restricting the information sent to non-validator nodes to be limited to an appropriate view.)

In this setting, it is straightforward to implement the access control policy of Table 1. Heavyweight zero-knowledge techniques are not required. However, it is required that the participants trust the validators with private identity information and detailed knowledge of the cap table of the private company. This suggests that the most appropriate validators may be securities regulators and other related agencies such as registries or exchanges, who are already required to obtain this private data, rather than commercial entities which may be competitors of companies listed on-chain.

Collecting user identity during on-boarding also better supports the need for identity information when dealing with legal contracts and processes like adjudication and litigation. If the validators are known, and all subject to the same jurisdiction as applies to a legal contract between parties to a smart contract, the validators can moreover be forced to make changes to on-chain data as ordered by a legal authority. When validators are subject to multiple jurisdictions, this may be harder, but potentially, cross-jurisdictional treaties may apply to support this. Treaties dealing with securities law (e.g., the "Hague Convention on the law applicable to certain rights in respect of securities held with an intermediary" [Hag]) appear to have only limited adoption at present, however, so we expect that restricting validators to respecting a single jurisdiction may be necessary to ensure that international validators will comply with legal orders.[45]

Private blockchains are therefore be a suitable platform in the near term for our hybrid legal and smart contract approach to SAFE contracts, although they achieve this at a cost of decentralization and "trustlessness". As mentioned above, it will be worthwhile to explore implementations using zero-knowledge techniques on open blockchains to determine whether this tradeoff is required.

[45] The international membership of the Libra Foundation accepting to be bound by Swiss law is an example of such an arrangement.