Smart (Legal) Contracts: A Case Study using Simple Agreements for Future Equity

Ron van der Meyden and Michael J. Maher

October 29, 2022

Abstract

The paper gives an overview of a project that has explored how distributed ledger and smart contract technology might be applied to a type of contract that is used in financing early-stage ventures: Y-Combinator's Simple Agreement for Future Equity. The range of questions that needed to be addressed in developing smart contract code for this application is discussed, from understanding financial and accounting aspects of the contract, game theoretic issues, dealing with open-textured legal terms in the source legal documents, smart contract architecture, and terms of an associated "smart legal contract" that gives the smart contract legal standing and covers residual issues that inherently cannot be governed by the smart contract itself.

1 Introduction

Distributed Ledger Technology (DLT), also known as "blockchain", provides a novel form of computational infrastructure for securing ownership records of a range of assets. The emergence of this class of systems was driven by the success of Bitcoin [Nak08] as a decentralized digital currency platform that supports enforcement of simple rules concerning the transfer of value. Subsequent DLT platforms such as Ethereum have incorporated more expressive languages for representing such rules, enabling a realization of Szabo's vision of "smart contracts" [Sza97]: commercially relevant multiparty agreements enforced by code running on computer networks. This development was soon applied to develop "Decentralized Autonomous Organisations" and "Initial Coin Offering" smart contracts, which were used in crowd-fundings raising substantial amounts of cryptocurrency value for projects developing blockchain technology and its applications.

These crowd-fundings were often of legally questionable status. Following a significant amount of fraud, regulators have started to assert their powers over blockchain-based fundraising, and to develop a regulatory stance to this new technology. The focus of the application of DLT is therefore increasingly moving towards regulated forms of assets, for which multi-party trust has historically been managed using legal contracts. This raises the question of how regulated assets can be supported using DLT technology.

Some simple forms of commercial agreement can be completely enforced using smart contracts: a swap of a digital asset represented on a blockchain for that chain's cryptocurrency is one example [Her18]. There is an emerging view, however, that the typical representation of a legal instrument on a DLT platform will require natural language text as well as smart contract code, a hybrid form of representation for which the term *smart legal contract* has emerged [AH22]. On this view, claims of immutability notwithstanding, the legal system will have powers to overrule actions performed on the blockchain. This position was foreshadowed already in Ian Grigg's notion of "Riccardian Contracts" [Gri04].

The methodology of smart (legal) contract development is still nascent, and raises many questions. To what extent can the contracts defining regulated assets be represented in smart contract code? Which of the provisions of commercial agreements can be automatically enforced using code on DLT systems, and which require representation in natural language? What issues, more generally, need to be addressed in developing a smart legal contract for a regulated instrument?

The present paper describes the outcomes of a project in which we have studied these questions by means of a case study of a legal contract that has become widespread in the financing of early stage ventures: Y-Combinator's Simple Agreement for Future Equity (SAFE) [Lev18]. Developed in order to simplify negotiations between seed investors and startup founders, these contracts are like convertible bonds but omit their provisions for payment of interest, simply offering either a return of the investment or a conversion of the contract to shares at the time of a priced equity round. (After a brief introduction to DLT and smart contracts in Section 2, we describe the structure of a SAFE contract, and summarize its terms in Section 3.) Due to the formality of the corporate finance domain, and the relative brevity and simplicity of SAFE's, they are, *prima facie*, a good candidate for a study intended to elucidate the nature of smart contract representation of legal contracts.

The detailed development of a smart contract representation of legal contracts, nevertheless, is non-trivial, and requires the careful consideration of a significant number of issues for the representation to adequately capture the legal dynamics of the source contracts. Indeed, the work of attempting to formalize SAFE contracts using smart contracts raises many issues, but leads, we claim, to an improved understanding of the contract itself. The outcomes of our effort to understand these issues has been captured in a number of papers [vdMM21b, vdMM21a, vdM21, Cou21], which are summarized in the present work.

To begin with, the preciseness and formality of code requires that financial operation of SAFE contracts first needs to be very well understood. As we describe in Section 4, the terms of a SAFE raise unanticipated questions about the meaning of the financial notion of "pre-money valuation" of a company in a priced equity round, which plays a key role in the terms relating to conversion of the SAFE to shares. There are multiple ways in which these questions are understood in practice, raising the question of which should be supported in the smart contract code.

Another issue that impacts decisions concerning the smart contract code is that the SAFE provides that in Liquidity Events (e.g., an acquisition of the startup), the SAFE investor is offered two options concerning the determination of their payout. One expects that an investor will always choose the option that maximizes their payout. However, it turns out that the payout may depend also on choices made by other investors, giving Liquidity Events an inherently game theoretic nature. We summarize our analysis of this game and its impact on coding of the smart contract in Section 5.

Section 6 discusses a number of impediments to straightforward translation of a SAFE contract to code. Related to the issue of precision and formality is that the source legal text contains open textured terms, not straightforwardly formalizable, and potentially subject to shifting interpretations as a result of legal rulings. We describe strategies for dealing with these difficulties in Section 7. A key element of our approach is to code the smart contract representation of the SAFE in a form that allows multiple different understandings to be accommodated. In the actual implementation of code components of the smart legal contract, as with any software engineering project, decisions need to be made concerning the architecture of the solution. In Section 8 give a brief description of our architecture, which has been designed to deal with a number of further issues, including the open structure of the context in which the SAFE operates, and the fact that some versions of the SAFE state declarative constraints instead of giving explicit computations for certain operations.

A further aspect of the financial operation of the SAFE concerns the freedom that equity round investors and the company have to negotiate the terms of equity rounds. In analyzing this aspect, we have found that a simple translation of the SAFE to smart contract code leaves the SAFE investor vulnerable to “gaming” of the SAFE contract by the company and the equity round investor. As described in Section 9, if the equity round is structured into two separate rounds at different prices, the SAFE investor can be left in a situation where their post-round stake in the company is significantly smaller than it would have been in an honestly constructed single round. We argue that this “attack” implies that the SAFE investor cannot rely upon a smart contract alone to protect their interests. The source legal text contains language that gives the SAFE investor legal recourse in such a situation, but its effect inherently cannot be captured in smart contract code. This is one example in which use of a smart legal contract, that retains the original legal terms, is critical.

The use of smart contracts and DLT as part of multi-party arrangements that have legal significance raises many general questions of law that we have not attempted to address in our work (see [DCP19, AH22] for extensive treatments). We do, however, in Section 10, lay out the demarcation between work done in code, and work done by natural language components of the smart legal contract we propose. Some further legal issues specific to the use of smart contracts for the equity financing domain are also discussed.

A key theme emerging from the work of this project has been to illustrate the way that undertaking a smart contract implementation of a legal instrument forces a greater degree of precision than is usual in the design of contracts. We draw out some general conclusions in Section 12, discussing circumstances in which the effort required for such precision is worthwhile, and giving general remarks on the discipline of Smart Legal Contract Engineering, of which the present work might be considered an exemplar.

2 Background: Blockchain and Smart (Legal) Contracts

Blockchain systems are a type of computational platform running on networks of computers, that coordinate to maintain a shared *consensus* state (a ledger of asset ownership, in many applications), in such a way that no individual node of the network, or even a submajority of the nodes, has the power to corrupt the state and make it differ from the state agreed by the network majority. The innovation of Bitcoin was to create the first broadly adopted digital currency by combining a number of earlier cryptographic ideas into a novel synthesis that for the first time made it possible to securely maintain such a consensus state over an *open* network (one that allows unconstrained entry and exit of nodes of the network). For the purposes of this paper, the details of the operation of the network protocol do not need to be understood, and the reader may conceptualize DLT as simply providing an uncorruptable computer accessible on the internet. The main point relevant below is that use of this computer comes at a cost, making it desirable to minimize the amount of computation that a user asks the platform to perform.

Bitcoin contains a simple form of programming language that enables constraints to be associated to amounts of monetary value, so that the value can be transferred only when the constraints are met in the transaction that makes the transfer (for example, certain forms of cryptographically signed assertion may need to be provided before the transaction can go ahead). This enabled a first realization of an open "smart contract"' [Sza97] platform, in which computer code running on a computer network enforces the terms of multiparty agreements concerning the disposition of certain assets. Bitcoin's programming language has its limitations, which have been removed in subsequent systems such as Ethereum, which enables, in principle, arbitrary code to be used to implement smart contracts.

A very simple example of a multiparty agreement that can be enforced by smart contracts is an "Atomic Swap" of digital assets represented on the blockchain [Her18]. For example, suppose Alice controls digital currency (Ethers) and Bob controls a digital asset (a CryptoKitty non-fungible token (NFT), representing that Bob is the owner of a digital image of a cute cat). Alice and Bob agree to an exchange: for two Ethers, Bob will transfer the NFT to Alice. An Atomic Swap smart contract can be used to enforce "Payment versus Delivery", so that it is guaranteed that either Alice will have control of the NFT and Bob will have control of two of Alice's Ethers, or both parties will still have control of their original assets. The situation where Alice has transferred her Ethers to Bob and Bob has not transferred the NFT to Alice or, conversely, where Bob has transferred the NFT and Alice has not transferred two Ethers, can provably not arise when Alice and Bob make use of an Atomic Swap smart contract [vdM19].

Such capabilities of DLT enable the development of platforms for trading of digital assets, and have led to considerable excitement about the idea of representing and automatically enforcing other forms of contractual arrangement using smart contracts, eliminating the need for the legal infrastructure provided by lawyers, arbitrators and courts. There are limitations to this vision, however. It may be feasible when all the assets involved are digitally represented, but obligations involving actions performed in the physical world obviously cannot be enforced by computer code, because it can neither directly observe nor directly affect the world. In many cases where smart contract technology is used, there remains, therefore, a need for standard forms of contract and legal processes if the agreements are to be enforced.

There is an emerging view, consequently, that there is a need for "smart legal contracts", a hybrid of computer code for DLT platforms and natural language text [AH22]. The precise structure of such hybrids remains a subject of debate, as does their legal status and the need for legal reform to accommodate the new technology.

Still, when all assets involved are digitally represented, the ambition of representing contracts as code remains plausible. However, determining whether this vision can be realized for a particular multi-party agreement, and precisely how, may require a detailed investigation. In what follows, we describe our investigation of this question for the particular case of SAFE contracts.

3 Background: SAFE contracts

SAFE (Simple Agreement for Future Equity) contracts [Lev18] are a class of contract that were developed and published by the Silicon Valley incubator fund Y-Combinator for use in seed financing of startup ventures. The principal motivation for the design of these contracts is to simplify the negotiations between founders and seed investors to decisions on a few key parameters of a standard document, avoiding the need for complex legal work.

Investments in companies typically come in the form of an exchange of shares for the money invested, but this requires a valuation of the company in order to determine a price for the new shares to be issued. Since early stage ventures are generally highly speculative and difficult to value, the valuation can be a sticking point in negotiations. SAFE contracts eliminate the need for a negotiation on valuation by offering the investor, in place of shares, a promise of shares at a future event (a future equity round, typically involving venture capital firms).

The number of shares to be issued to the investor at this future event is determined according to a conditional "conversion formula" that gives the investor a lower price in the equity round if the company has performed well, but also protects the investor against losses in case of poor performance. In essence, SAFE contracts are similar to convertible bonds, except that their provisions for payment of interest in the investment prior to conversion have been eliminated. (Again, the motivation for this is a simplification of the instrument, saving complexities in its tax treatment.)

There are multiple variants of SAFE contracts, differing with respect to the details of the conversion formula. In the original variants of the Y-Combinator SAFEs [Y C], there were 4 versions, depending on a choice of which of two parameters called the Valuation Cap and Discount are included in the conversion formula, along with the Pre-Money Valuation of the company (that is, the valuation of the company agreed by the investors just before the equity round.) The Valuation Cap, effectively, sets a limit on the price at which the SAFE investors money is converted to shares, and the Discount is, effectively, a percentage discount on the price that will be paid by the new equity round investors. Y-Combinator later switched to a new set of "Post-Money" variants [Lev18], in which use of the Pre-Money Valuation is replaced by the Post-Money valuation.

We focus in this paper on a discussion of one of these variants, the Pre-Money SAFE with Cap only [Y C16]. This contract is in the form of a template with just two numerical parameters that are filled in before the contract is signed: the Purchase Amount (the amount being invested by the SAFE investor) and the Valuation Cap. The conversion formula is defined by the following clause:

> (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 notion of "price per share of the Standard Preferred Stock" is treated as a primitive input available at the time of the equity round. The Safe Price is defined by the following clauses:

> **"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.

Beside Equity Financing events, SAFE contracts recognize two other event types that produce some form of return to the SAFE investor: Liquidity Events (an Initial Public Offering, acquisition or merger of the company) and Dissolution Events (in which the company is dissolved). In Pre-Money SAFEs there is no provision for dividend distributions to SAFE investors; these are an additional event type in the Post-Money SAFEs.

The Liquidity Event clauses differ depending on the SAFE variant, but their overall structure considers a SAFE holder to have two options in the case of a Liquidity Event: to Cashout or to Convert. The Cashout option returns to the SAFE holder the Principal Amount that they invested in purchasing the SAFE, or, if there are inadequate funds to be distributed, a pro-rata amount determined from the claims of all the SAFE investors choosing the Cashout option. The Convert option first converts the SAFE to shares, and then gives a pro-rata payout determined from the claims of the Converting SAFE holders and all other share-holders. Investors choosing to Cashout have priority over investors choosing to Convert as well as other shareholders, so the Cashout amounts are distributed first, before the Conversion amounts are determined.

Dissolution Events also yield a payout to the SAFE investor, more simply determined. SAFE investors are treated with priority over shareholders. In the case of the Pre-Money SAFE with Cap only, the payout is simply either the Purchase Price or, if there are insufficient funds for such a distribution to the SAFE investors, they receive an amount determined pro-rata according to their Purchase Amounts.

A SAFE terminates if there is either an Equity Financing, Liquidity Event, or Dissolution Event. In addition to the clauses describing this dynamics of SAFE contracts, there are clauses covering representations made by the company and the SAFE investor that underpin their capacity to enter into the contract and the legal validity thereof, as well as 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.

4 Financial Analysis

When setting out to develop code that represents a contract, it is necessary to develop a correct understanding of the intended interpretation of the terms of the contract, as well as the way that those terms are applied in practice. *Prima facie*, the Equity Financing clause has a straightforward statement. However, the nature of the SAFE contract itself raises issues for the interpretation of some of the terms of art that it contains, that requires clarification and forces some decisions to be made about how to convert these terms to code. We give a sketch of the issues in the present section, and refer to [vdMM21a] for a detailed analysis.

The terms “pre-money valuation and “post-money valuation are terms of art for equity investors. Their meanings are buttressed by a set of equations that relate these terms to other parameters of an equity financing. However, once a company has issued a SAFE (or other convertible note), these equations are no longer consistent [vdMM21b]. This has led to confusion among equity investors and founders who, using different subsets of the equations, arrive at different conversion outcomes.

As a result, multiple conversion methods have evolved (see, for example, [Col]). One method can be viewed as accounting for the SAFE as a liability of the company that is discharged as a result of the equity round. An alternative view is that a SAFE is represented on the capitalization table of the company, corresponding to some (variable, until the equity round) number of shares. (While eschewing the provision of advice on the matter, Y Combinator states that its preferred view is that SAFEs are “equity instruments” [Y C18].) Other methods seem more ad hoc. In summary, in the context of issued SAFEs, the terms of art are now ambiguous.

In some of these potential conversion methods, there is the further issue that the conditions defining the number of SAFE Shares to be issued introduce a circularity. Accounting for the SAFE as a liability requires, in order to value the company to determine a share price, that the SAFE be valued first, but this in turn requires a valuation for the company. This issue can again be resolved by treating the contract as setting up a set of simultaneous equations from which the relevant quantities can be deduced. (Post-Money versions of the SAFE [Lev18] explicitly state a set of simultaneous equations rather than give a formula for determining the SAFE share issuance.)

The question this raises for the task of representing the SAFE contract in code is which of these conversion methods to support in the code, and the extent to which the relevant calculations should be performed off-chain. Representing the solution process for a set of simultaneous equations in code is potentially non-trivial[1] and its cost may be prohibitive on platforms like Ethereum that charge for computation performed on the blockchain.

The solution to these difficulties that we have adopted is to represent the scenario of a company issuing SAFE contracts in a general way that accommodates multiple views. Rather than have the SAFE smart contract explicitly calculate the share issuance in the equity round, we view it as stating a declarative constraint on a description of the equity round proposed by the company. The proposal states the parameters of the equity round: the pre-money valuation, the number of shares to be issued to each new investor and to each of the SAFE investors, and the share price for these issuances. The code governing the company proceeds with the proposed share issuance only if this proposal is approved by the smart contracts representing each of the SAFE contracts.

The benefits of this approach are that the smart contract embodies the meaning of terms like “pre-money valuation” as agreed by the founders and the SAFE investor at the time, and that it leaves potentially computationally complex equation-solving processes to off-chain computation (replacing this by a more efficient computation that checks that the proposal satisfies the SAFE contract). This approach also allows for the smart contract to be coded to permit manual approval by the SAFE investor.

It remains for the SAFE investor and company to select a coding of the SAFE that captures the desired degree of flexibility. For example, they may decide that a particular conversion method applies, but additionally allow for manual approval by the SAFE investor in case required by negotiations in the course of the equity round. In any case, as discussed below, there is a need to allow for negotiated revisions of the contract and legal rulings for variance of the SAFE terms.

5 Game Theoretic Aspects

A further issue requiring some analysis to determine what is in scope of enforcement by the smart contract arises from the Liquidity Event clauses

[1]With sufficient expressiveness in the contract terms, solutions could even become uncomputable.

of the SAFE contracts. As noted in Section 3 these clauses give a SAFE investor a choice between two options (Cashout and Covert), each producing a payout amount. However, some SAFE contracts state that the investor will receive the maximum of the payouts resulting from the two options. Obviously, this is the option that would be chosen by a rational investor, but this assumes that this maximum is well defined.

In fact, an analysis of this question reveals that when multiple SAFEs have been issued, an investor's payout for a given choice depends on the choices of other investors. This gives the Liquidity Event setting an inherently game theoretic nature, with maximization of one investor's payout potentially at the cost of the reduction of another investor's payout. To make sense of the notion of "maximum", in this case, requires that we understand the likely outcomes of these competing interests using an appropriate notion of game theoretic equilibrium.

One of the most commonly considered notions of the "solution" of a game is that of Nash equilibrium. A *discrete Nash equilibirium* is a *strategy profile* (a collection of choices of move for each of the players) from which no player has an incentive to deviate, in the sense that a change of move by any one player, while the others do not change their move, does not increase the payoff of that player. If there is a strategy profile that maximizes the payoffs of all players simultaneously, then it is also a Nash equilibrium, so one approach to addressing the problem of maximizing the payoff of all players is to understand the discrete Nash equilibria of the game.[2]

We have conducted an analysis of the discrete Nash equilibria of the Liquidity Event games arising from both Pre-Money and Post-Money SAFE contracts in [vdM21]. This analysis shows that the situation with respect to the existence of Nash equilibria is complicated. If the company issues both Pre-Money and Post-Money SAFE contracts, then there are situations where the Liquidity event game has no discrete Nash equilibria. However, if the SAFEs issued are uniformly Pre-Money SAFEs, or uniformly Post-Money SAFEs, then it can be shown that discrete Nash equilibria exist, and indeed, amongst the discrete Nash equilibria, there is at least one such equilibrium (but possibly exponentially many) that simultaneously maximizes payout

[2]Mixed strategies seem less appropriate in this setting, since the game is played only once. In general, one would also prefer that a financial contract deterministically specifies a payout in a specific scenario, rather than leave this subject to uncertainty. Other forms of game theoretic equilibria, such as correlated equilibria, may be of interest, particularly since agreements of the parties playing a game can be enforced using smart contracts on the blockchain. However, we do not have an analysis of such equilibria of the SAFE Liquidity Event games at this time.

for all SAFE holders. Moreover, this equilibrium can be computed in time that is polynomial in the number of SAFE contracts that have been issued.[3]

It could therefore be recommended that the company should only issue SAFE contracts of a uniform type. It is worth noting that this could in fact be enforced on the blockchain by restricting the operations by which a SAFE contract is issued to allow only SAFE contracts generated from a fixed template to be added. This restricts the generality of the implementation, however. Provided that the company takes care to check that the specific set of contracts that it issues does guarantee that the desired solution of the Liquidity game will always exist, the issue of a lack of solution does not arise. Plainly, a software tool that performs this type of consistency check would be desirable. However, we do not have, at present, a general algorithm that is better than a brute force (exponential time) search for computing an optimal Nash equilibrium under the "promise" that it exists.

A simpler alternative, followed in [Cou21], is to simply accept that the situation is inherently game theoretic, and not attempt to use the blockchain to compute and enforce a particular solution. Instead, the choice of move is left to the SAFE holders to decide. This is certainly desirable from the point of view of maintaining the human autonomy in this choice that is allowed by the original contract. Again, as with Equity Financing events, one can also conceive that investors may negotiate a distribution that varies from the explicit terms of their contracts. To cover this case, there could be an option in the smart contract to approve a specific proposed distribution.

6 Impediments to Formalization

Some of the clauses of a SAFE contract, such as that describing the conversion formula in the case of an Equity Financing (see Section 3) appear to be sufficiently technical that they can be readily translated to code, with a high degree of confidence that the code correctly captures the meaning of the source text.

Other clauses are more problematic, however, because of the limited precision of natural language. Natural language legal text may be *ambiguous* (having multiple meanings), *vague* (lacking sharp boundaries of applicability even in known cases) or *open textured* (of uncertain applicability

[3]The algorithm does require sorting of certain parameters, so has asymptotically non-linear running time, which is potentially a limitation for implementation on a blockchain that charges for computation costs. However, the number of contracts will in practice often be small, and the algorithm can be implemented in linear time with the sorting operation performed off-chain, which makes it feasible even in such settings.

in unforeseen circumstances; some authors use this term also for vagueness) [Wit53, Wai68, Har61, Har58].[4] It has long been understood that such forms of indefiniteness present challenges to the formal representation of legal text in computational systems [McC80, BCS88].

The definition of an Equity Financing is one example of this difficulty in the context of a SAFE contract. It states:

> "**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 contains multiple terms subject to indefiniteness, in various forms.

As we have already noted above, in Section 4, although it might have been expected to have precise meaning, in the presence of SAFE contracts, the notion of Pre-Money Valuation is subject to different interpretations, depending on whether we view a SAFE to be accounted for as a liability or on the company's cap table. Although we have described this as ambiguity, it can also be considered to be an example of *open texture*, in Waismann's original sense of presenting difficulties of interpretation in unforeseen circumstances (a circularity in the problem of valuing the company, caused by the presence of a SAFE.)

A related issue is that the contract is *incomplete* in not stating the relationship between pre-money valuation and share price. Incompleteness of contracts is a well recognized phenomenon, ascribed usually to the fact that drafting contracts so that they cover all possible future circumstances (some very unlikely to occur) may be complex, costly and delay contract formation. The incompleteness here arises, instead, from the presumption, falsified by SAFEs themselves, that this relationship is well understood to be common knowledge background to the contract. This is an instance of the contract not being *self-contained*, but drawing upon external legal and business practice context for its interpretation. Another example occurs in that the contract defines the notion of "change of control", which acts as a trigger for a Liquidity Event, by reference to legislation: the Securities Act.

Other terms of indefinite meaning in the clause are the vague terms "bona fide", and "principal purpose". Even the technical notion of "Preferred Stock" might be considered to be open textured, given that the rules adopted by a company concerning the rights associated to different classes

[4]Even the fact that authors disagree on the meaning of the terms in this classification of indefiniteness illustrates this weakness of natural language.

of stock are open ended, potentially allowing for unusual definitions that make it difficult to decide whether a particular class of stock should count as Preferred.

Also problematic in this definition is the reference to a “transaction or series of transactions”, which leaves it open to interpretation which transactions are part of the single Equity Financing event. If a company sells Preferred Stock, and one week later sells more Preferred Stock, should these events be treated as part of the same Equity Financing events or not?

The *open structure* of the scenario of a company issuing shares also presents challenges to formalization. There is an unlimited variety of potential rules governing shares, governance rules relating to these shares, and company constitutions.

7 Strategies for Dealing with the Impediments

We now discuss a number of strategies that can be applied in developing a smart contract formalization of a SAFE contract in the face of the difficulties of the previous section. When developing a formalization of a contract, a wide spectrum of options remains open with respect to which aspects of the contract to formalize, and how to do so.

One point to note is that specific formalization decisions could be part of the contract negotiation process itself. Ultimately, the parties sign a contract in order to decrease their uncertainty about the future, relying on the legal system to ensure that future events will be constrained as described in the contract. If a smart contract is one that the parties understand, and they are willing to accept and abide by its precise interpretations of natural language expressions, then it may be moot that the formalization does not correspond exactly to the natural language meanings in a document on which the formalization was based. A constraint on the workability of such acceptance, however, is that the parties may be limited in their ability to foresee potential future situations, and in their understanding of the formalization, because of its technical complexity and language. Even experienced programmers have limitations in their capacity to understand the behavior of complex code in all possible scenarios, as evidenced by the pervasiveness of software bugs and vulnerabilities.

One way to tame these complexities is by *limitation of fact patterns.* We may note that smart contract development differs from the broader field of legal knowledge formalization (e.g., for the purpose of developing expert systems for provision of legal advice), in that the fact patterns in which the applicability of a term to be formalized needs to be decided is not always arbitrary, but can sometimes be restricted by the smart contract itself.

One example of this in SAFE contracts relates to the term "Preferred Shares". If the company is allowed to issue Preferred Shares at will (e.g., in lieu of payment to a contractor), the smart contract needs to determine whether that issuance is (part of) an Equity Financing event, in order to trigger conversion of the SAFE. This may be impossible to automate, particularly as questions of intent arise.

However, the smart contract can be designed so that once a SAFE has been issued, the only way that Preferred Shares can be issued is by the company taking an action that declares that the issuance is part of an Equity Financing. This eliminates the complexities of interpretation by restricting the company's scope of action. In effect, it adds to the contract agreement the easily understood condition that the company will not issue Preferred Shares except in an Equity Financing. An issuance not satisfying this constraint would in any event be unusual, as the company would generally prefer to issue Common Stock, so the constraint is likely to be acceptable to the Company. It will also be acceptable to the SAFE investor, since it gives greater certainty about the performance of the contract.

A related limitation of fact patterns in the way we have implemented our SAFE smart contract is by a simplistic treatment of share classification. If we were to allow arbitrarily complex rules defining an unlimited number of different classes of shares, we would potentially require a formalization of the process of determining whether a particular share is a Preferred Share. Instead, each share in our formal representation is, for purposes of the SAFE smart contract, simply declared by the company, on issuance, to be of type Common, SAFE Preferred, or Preferred.[5]

We have noted above the difficulty that determining the number of shares to be issued to a SAFE investor may require solution of a set of simultaneous equations, and may also be subject to negotiated interpretation. Our approach to this difficulty is to represent a SAFE not as a program that automatically computes the number of shares to be issued in conversion of the SAFE, but instead as a program that gives a true or false answer to the question of whether an equity round proposed complies with the terms of

[5]We do think it is desirable to allow for flexible formal encoding of rules relating to share classes, particularly after SAFEs have been discharged, but the work of doing so is orthogonal to the work of encoding SAFEs. The strategy of relying on the company to declare the categorization for the purpose of the SAFE can be combined with the use of a richer representation, and reliance on the legal system to adjudicate the appropriateness of the classification.

the SAFE. The proposal is comprised of information about the money paid by each participant, the shares they are to be issued, and the class of those shares, as well as the understood pre-money valuation of the round. The round is allowed to proceed only if all the SAFE contracts issued by the company indicate that the proposal complies with their terms, and the new investors also assent by provision of payment. (See the next section for more detail.) We apply here a well understood principle in computer science, that it is generally easier to compute whether an answer to a problem is correct, than it is to find such an answer. This approach also allows for ambiguity, and negotiated agreements about a specific equity round can be handled by first revising the SAFE smart contract so that it will accept the negotiated proposal.

As a general remark, in developing a smart contract from a source natural language document containing indefinite terms, such terms can be handled in the smart contract either by full automation or can draw on input from human decision makers. In each case, the choice has implications for the powers of the parties to the contract. Where human decision makers provide input, it is important to carefully consider *which* of the parties to the contract (or external sources of information) should provide this information, and the impact this has on the balance of power between the parties to the contract.

For example, in our treatment, specific shares are simply labelled as Preferred by the company in its equity round proposal, and there is no automated analysis of the rules associated with these shares to determine that this classification is appropriate. *Prima facie*, this might appear to give the company control over this decision. However, the company's proposal is effectively an offer and it remains open to the new investors to reject the offer and refuse to provide payment. Their acceptance of the offer can therefore be understood as their agreement to the interpretation of the shares as Preferred. The SAFE investors are apparently in a more vulnerable position, particularly when the SAFE smart contract automatically approves the Equity Round proposal, since they then do not have an opportunity to assent to the classification. On the other hand, since code is able to enforce the contract constraint that the SAFE Preferred shares are identical to Preferred shares except with respect to the price paid, and the SAFE contract does not explicitly grant the SAFE holder powers to negotiate the construction of the equity round, it can be argued that the smart contract's treatment of the SAFE investor's powers is not materially different from that of the source legal contract. As we discuss below, in our ultimate solution, the SAFE investor also has the protection of a legal challenge based on natural language text in a smart legal contract, so has a second line of defense.

8 Software Architecture

The overall architecture of our smart contract implementation is given in Figure 1 and Figure 2. The architecture makes use of several types of object implemented as smart contracts on the blockchain. A `Company` object represents the state of the company's cap table and accounts. In order to deal with the open-endedness of the company's governance regime during different stages of its development, we use a *controller* proxy pattern, in which access to operations on the `Company` object is mediated by a Controller object, which is able to impose various restrictions on operations that parties (usually, company officers) attempt to perform on the `Company` object. A bespoke controller object can be programmed so as to impose the desired control regime.

In our solution, this controller object is also dynamic. While SAFE contracts have been issued by the company, but before these have been converted in an equity round, access to the Company object is controlled by a `Safe_Controller`, which checks that a requested action is compliant with the SAFE contracts that the company has issued. (For example, an attempt to issue Preferred Shares, not in the context of an equity round, would be rejected.) `Safe` objects represent the SAFE contracts themselves, and respond to requests for authorization of actions on the `Company` object with true/false answers.

The company's proposed equity round is also sent to the `Safe` objects for approval. If they all approve, an `Equity_round_swap` object is created, which guarantees that the round completes if and only if all new investors provide payment, and returns payments in case of failure of the round. After the round, the `Safe_Controller` is replaced by a new controller that encodes the company's governance structure after the equity round, as illustrated in Figure 2. For example, this controller might enforce rules relating to the post-round board structure and voting powers, as well as constraints on the transfer of shares and participation rights in future equity rounds.

A similar transformation of the architecture occurs during Liquidity Events and Dissolution Events. Each of these types of events terminates the SAFE contracts the company has issued, so they have no further connection to the `Company` after they have occurred. (But their past existence will be immutably recorded on the blockchain.)

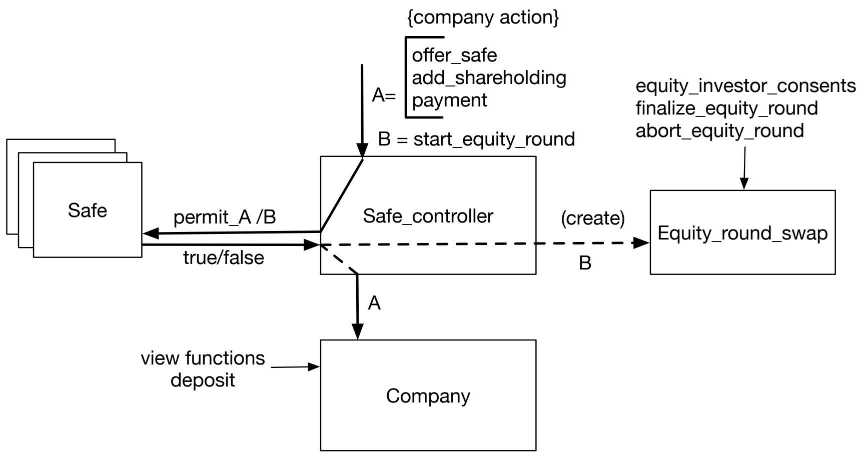

Figure 1: Architecture with `Safe_controller`

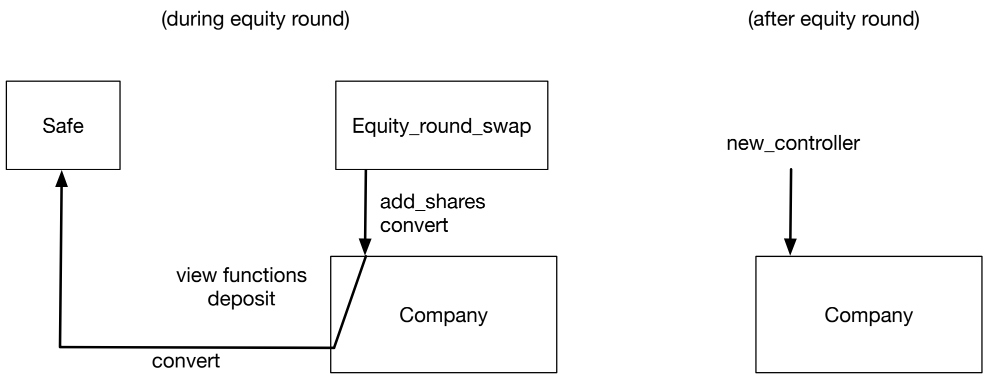

Figure 2: Architecture configuration during and after an equity round

9 A Limitation of the Smart Contract Implementation

As already mentioned, by reference to Atomic Swaps, some smart contract applications are able to completely enforce the requirements of multiparty agreements, obviating the need for legal contracts to protect the interests of the parties. Inasmuch as the smart contract implementation sketched above is able, in the event of an equity round, to enforce application of the formula for the share issuance to the SAFE investor (or an alternate agreement later negotiated and represented in the smart contract), it might appear that the investor's interests in this event are completely enforced by the smart contract. In fact, there is a possible "attack" on the smart contract, that leads to the conclusion that it does not provide equal protections to the original natural language SAFE.

In this attack (described an analyzed in greater detail in [vdMM21b]) the company and equity round investor(s) collude against a SAFE holder in order to deprive the SAFE holder of their rights. They do so by structuring the new investor's investment into two equity rounds rather than one. The attack is applicable when the pre-money valuation of the company falls below the Cap of the SAFE. In the first round, a small amount of the new investor's money is invested at a valuation equal to the Cap. This round discharges the SAFE exactly according to the SAFE's formula, and the SAFE investor believes that they hold shares of value equal to their original investment, so have suffered no loss. In the second round, the new investor purchases shares with the remainder of their money, but at a much lower valuation.

The net effect is that the new investor pays an effective price corresponding to their actual valuation, and receives an appropriate number of shares, but the SAFE investor is left holding both a smaller proportion of the company than they would have been entitled to in an honestly constructed round, and are deprived of the downside protection that the SAFE is intended to provide. The other shareholders (typically, the founders) are left holding a correspondingly larger share of the company. In effect, the SAFE holder has been cheated of their rights under the contract.

The difficulty here is that smart contracts, when allowing certain events to occur, act in way that we might call "*nuncospective*" (in the now). The fraud may not be apparent at the time of the first round, only to be revealed when the second round occurs. At this later time, the smart contract has been terminated and can no longer provide protections. By contrast, legal contracts can be applied *retrospectively*, when a court rules that events that have occurred in the past were not compliant with the contract terms. Retrospectivity is a powerful feature of a legal system, since it affords waiting for evidence of contract violation to emerge. The cost (in terms of both code design and computation) to provide smart contracts with similar power would seem to be prohibitive. In contrast, with a legal SAFE, the SAFE investor has the option of suing for compensation after the second round. The legal SAFE provides multiple protections against this attack, by allowing that the equity round may consist of a "sequence of events", requiring that it be "bona fide", and have "the principal purpose of raising capital". It is the very indefiniteness of this language, combined with retrospectivity, that allows the attack to be ruled to be a violation of the technical terms of the SAFE.

Note, moreover, that the determination as to whether such an attack has occurred cannot be programmed. It is possible that the reduced valuation in the second round came about because of events such as a competitor entering the company's market, an earthquake, a pandemic, or any number of other reasons. Only human judgement can determine the validity of such a justification, based on consideration of a set of relevant facts that cannot be programmed because they cannot be predetermined. Of course, we could program a right to appeal to a human adjudicator into the smart contract. However, it remains the case that the adjudicator will require a natural language description of the intent of the contract on which to base their determination. A natural language text, therefore, cannot be eliminated.

10 Supporting Legal Contract

Besides the attack of the previous sections, several other reasons also exist that suggest that the smart contract code alone cannot satisfactorily capture all of the effects of the legal SAFE contract. We therefore propose that a "smart legal contract" is required, in which the smart contract code is accompanied by, or integrated with, a natural language contract. We have taken no stance in our work on the precise form of such a smart legal contract, since this and the precise language may be jurisdiction dependent. We just indicate here some of the points that we believe should be covered, and why.

Some relate to the legal standing of the smart contract representation of the state of the company. The company itself will be a legal entity recognized in some jurisdiction. For a record of share or SAFE ownership on the blockchain to represent rights within that jurisdiction, that record needs to be accepted as a valid representation of these rights. The legal contract should therefore contain such text as is necessary to garner such acceptance (assuming that the jurisdiction accepts such representations at all).

To ensure correctness of the smart contract's issuance of SAFE shares in Equity Financing events, or payouts in Liquidity or Dissolution events, it is also necessary that the on-chain record be a *complete* representation of the relevant state of the company when those events occur. To prevent the company issuing (legally recognized) shares off-chain, text is required stating that the company will issue shares only on-chain, and conduct an equity round on-chain only when the representation is complete.

There should also be an obligation on the company to conduct the first equity round after a SAFE issuance using the smart contract, by submitting an on-chain proposal that completely, correctly and honestly represents the structure of that equity round. In particular, this obligation should be stated in a way that protects against the attack of the previous section. Similar obligations apply to Liquidity and Dissolution events. There may also be obligations on the SAFE investor, for example, that they sign paperwork (possibly via an on-chain digital signature) relating to the equity round.

Finally, protection against software errors and human error in use of the smart contract should be provided using a statement concerning processes and arbitration authorities and/or jurisdictions to be applied to arbitration of disputes concerning the operation of the smart contract. An ability to react to legal rulings and discovery of software errors may require that the smart contracts be coded so as to be allow unpredictable changes and compensating transactions to be made, subject to appropriate authorization. However, even where such changes cannot be made, we note that since the on-chain record of the state of the company has been imbued with legal significance by means of an off-chain declaration of the company in the context of a jurisdiction's regulations, that validity can similarly be revoked by off-chain events, and transferred to a new, corrected smart contract record on-chain. (A proviso is that in the event that there are digital assets in the original smart contract, there is process for their transfer to the new one.) We refer to [vdMM21a] for a longer discussion of these issues.

11 Further Considerations

Our focus in this paper has been on issues of representation of the SAFE contract itself. Other issues that need to be addressed in developing a blockchain-based implementation of SAFEs are the user interface and the blockchain platform on which the smart contracts are deployed. See [Cou21] for work on these issues, in which a permissioned blockchain is used to manage data privacy concerns. We consider a permissioned chain setting to be better suited to private company cap table management solutions since the anonymity of investors common in open blockchain systems is inappropriate to regulated settings where investor identity information may be required for compliance purposes.

Our solution has not been deployed in a commercial blockchain application for private equity management, so we do not have information about the market response to our proposals. It is therefore unclear whether the market will perceive the security benefits and efficiencies of a smart legal contract solution for SAFEs a sufficient advantage over the protections provided by a legal contract alone. In general, for assets that trade with a relatively high frequency, there appears to be acceptance that blockchain representation provides benefits and savings. The Australian Stock Exchange, for example, is developing a blockchain-based clearance and settlement system for public equities. The extent to which this acceptance will extend to private equity markets is less clear, and SAFEs themselves are not subject to trading since non-transferability is amongst the contract terms. A number of commercial projects are developing blockchain-based cap table management solutions targeting also private equity markets [Bou, Kor, Ver], but again, we are not aware of any independent studies concerning market traction for such services.

In the course of our work, we have uncovered some weaknesses of the SAFE contracts we analyzed. A well designed contract should satisfy criteria such as the following.

- Where transitions in the state of the contract rely upon the existence of a solution to a set of equations, there should provably exist a solution in all cases for all scenarios that could arise.

- It is desirable that transitions either be defined so as to be independent of choices made by holders of other contracts, or, where the contract sets up a game theoretic scenario, for a well-defined game theoretic equilibrium to provably exist. Ideally, this equilibrium should be well-justified as striking an optimal balance amongst the interests of the multiple parties.

SAFE contracts do not always satisfy such criteria. There are situations where Post-Money SAFEs set up simultaneous equations that do not have solutions [vdMM21b]. There are also situations where there may fail to exist a discrete Nash equilibrium for the game associated to a Liquidity Event [vdM21]. Our approach to dealing with this difficulty has been to allow for compromise contract transitions to be made on the basis of explicit approval from the contract holder.

When developing a smart contract representation for a particular contract type, one could potentially redesign the contract terms, based on an analysis of first principles requirements and the extent to which smart contract technology offers opportunities to provide stronger security guarantees than are available from current legal contract forms. (Atomic Swaps can be understood as providing such an improvement over legal contracts.) We have not attempted to develop such an improvement of SAFE contracts in our work, and leave the exploration of this issue for future work.

We have argued that a natural language legal contract remains necessary for SAFEs to be represented as smart contracts. To a computer scientist with "Code is Law" ambitions for smart contract technology, the need for a legal contract backing a smart contract might at first be unsatisfying and a repudiation of their ambitions. However, the approach of smart legal contracts can also be understood as fully concordant with the well accepted computer security principle of Defense in Depth, with the legal contract providing a second layer of defense when when the behaviour of the code diverges from the intent of the agreement, whether by software bug or by manipulation of the input to the smart contract.

One of the benefits of formalization of legal contracts as smart contracts is that it forces a focus on issues of contract well-definedness and completeness, and on whether the contract terms in fact provide the parties the guarantees they desire in the face of 'gaming' of the rules by other parties. Detailed analysis of such issues may require considerable effort. However, we suggest that the effort is worthwhile for industry standard documents, where costs can be amortised across multiple users. Certainly SAFE contracts are intended to be a standard contract form, so we believe that ongoing work to improve and better understand such contracts is worth pursuing.

More generally, our work on SAFEs points to the potential significance of a discipline of "Smart Legal Contract Engineering" (a deliberate pun), best carried out by interdisciplinary teams of lawyers, computer scientists and domain experts (e.g., from Finance in the case of SAFE contracts). We, the authors of this paper, are computer scientists, who have conducted this project for purposes of understanding the scope of smart contract technology and the requirements for computational support of smart legal contracts. We encourage legal and finance experts to take up the issues in their domains that have been exposed by our work on computational formalization of SAFE contracts.

12 Conclusion

We have explored the capability of smart contracts to express legal contracts, through a case study of Y-Combinators Simple Agreement for Future Equity. We identified issues in this agreement that indicate that the surface simplicity hides awkward complications around agreement termination. Others have pointed to difficulties when the agreement does not terminate [GC16]. Y-Combinator has addressed several issues in the new Post-Money SAFEs, but the effects linger on in the many extant Pre-Money SAFEs and the continuing use of these SAFEs outside Y-Combinator.

While the above issues are specific to SAFEs, the SAFE also exhibits features that are well-known to create difficulties in formalizing legal documents, in general. We identified some strategies for smart contracts that can limit their impact. However, we also identified protections of the legal SAFE that require open texture and retrospectivity, which seem difficult, perhaps impossible, to emulate in smart contracts. We draw the conclusion that for this application, smart legal contracts, which combine both legal contract and execution by smart contract, are preferable to stand-alone smart contracts. We outlined some requirements for the legal contract to appropriately support our design of a smart contract for SAFEs. However, there is a need for greater understanding of the needs of smart legal contracts, which will require interdisciplinary study.

References

[AH22] Jason Grant Allen and Peter Hunn, editors. *Smart Legal Contracts: Computable Law in Theory and Practice.* Oxford University Press, Oxford, 2022.

[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 3961. Quorum Books, New York, 1988.

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

[Col] Derek Colla. Calculating share price with outstanding convertible notes. `https://www.cooleygo.com/calculating-share-price-outstanding-convertible-notes/`. [Online, Accessed Sep 2019].

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

[DCP19] Larry A. DiMatteo, Michel Cannarsa, and Cristina Poncib, editors. *The Cambridge Handbook of Smart Contracts, Blockchain Technology and Digital Platforms.* Cambridge Law Handbooks. Cambridge University Press, 2019.

[GC16] Joseph M. Green and John F. Coyle. Crowdfunding and the not-so-safe safe. *Virginia Law Review Online*, 102:168 – 182, 2016.

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

[Har58] Herbert 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.

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

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

[Lev18] Carolynn Levy. Safe financing documents 2018. Online `https://www.ycombinator.com/documents/#safe`, Accessed Sep 2019, Sep 2018.

[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.

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

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

[vdM19] Ron 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.

[vdM21] R. van der Meyden. A game theoretic analysis of liquidity events in convertible instruments. arXiv https://arxiv.org/abs/2111.12237, Nov 2021.

[vdMM21a] R. van der Meyden and M. J. Maher. Can SAFE contracts be smart? manuscript, 2021.

[vdMM21b] Ron van der Meyden and Michael J. Maher. Simple agreements for future equity – not so simple? manuscript, http://www.cse.unsw.edu.au/~meyden/research/SAFEnss.pdf, 2021.

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

[Wai68] Friedrich 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 C16] Y Combinator. *Safe: Cap, no Discount*, https://web.archive.org/web/20180831020232/http://www.ycombinator.com/docs/SAFE_Cap.rtf 2016.

[Y C18] Y Combinator. *Post Money Safe User Guide*, https://www.ycombinator.com/docs/Post\%20Money\%20Safe\%20User\%20Guide.pdf 2018.