Anthony Macey Follow
All views are my own, it is unlikely anyone, least of all my employer, agrees with what I might have to say.
Jul 8, 2016 · 5 min read

What's in a Name?—The Disambiguation of Smart Contracts

2014 and Gavin Wood takes the stage at the auditorium in Barclays Whitechapel Accelerator space to describe a brave new world of autonomous organisations and law embodied by code.

Even as the endorphins rushed at the possibilities, the ever present cynical side of me felt that there were a lot of unanswered questions in this utopic world of computer driven actions, primarily one word kept coming to mind; *arbitrage*.

The reason for this nagging thought is because of the divide between *something that has occurred* and the *legal outcome of what has occurred*.

Law is not a hard list of rules to be followed in binary fashion like they would be interpreted by a computer system, law is around the governance of people and their actions which makes them entirely contextual to the individuals involved.

Contracts, by their very nature, are embodiments of a best endeavours embodiment of an agreement. Despite best endeavours there will be disagreements, this is due to the context of the parties involved in the contract.

Traditional Contracts—A multi-party agreement

A traditional contract is paper based and generally finalised as a result of negotiation between the parties to the contract. The contract can only be executed without complication if both parties agree unequivocally that the conditions for the contract have been met.

If either party disagrees, the contract can be challenged. Lawyers will interpret the clauses of the contract and will engage in a process of dispute resolution, which parts of the contract have been fulfilled and in who's favour.

"Smart Contracts"—Code execution

Many consider a "smart contract" to be *just code*. In some ways it is similar to a traditional contract as the parties that agree to this on both sides have to agree to the conditions that the contract will execute under. When it becomes ambiguous as to whether the conditions have been met is one of the main issues facing "smart" contracts as sometimes this would be disputed.

The solutions for this include, but are not limited to;

- Trusted Oracles
- An option for arbitration
- An acceptance of "the code is the law"

Everything in Between—The not so smart, but not entirely traditional contract

Nick Szabo's The Idea of a Smart Contract in 1997 speaks to the evolving idea of a smart contract. Using the, now regularly featuring, example of a vending machine, value is provided for a service and an automated function then delivers the goods as per the conditions of the contract you have entered in to; that of the request to purchases goods for a set price.

What the example does not speak to is the external requirement of a third party arbiter if the machine should fail to function in the expected way, however this may not be an oversight as the deterministic computation of code *should* mean that the code executes in exactly the same way every single time without ever failing. There are no mechanical parts to fail, theoretically, until you need to introduce external information (for example a price, an exchange rate, a time) or *action* the introduction of information (for example a change in price, exchange rate or time).

In Szabo's follow up paper A Formal Language for Analyzing Contracts, he speaks of "partially self-enforcing and protected execution as smart contracts."—The key word here being *partially.*

Ian Griggs' seminal work on The Ricardian Contract and the Seven Layers of Financial Cryptography speaks to some of the challenges above, long before the modern definition started to circulate and this introduces the idea of automation where possible while allowing for the deeper complexities of legalese to be interpreted by lawyers. The notion of linking legal prose to computer parameters is fundamental to providing an execution of key parts of complex legal contracts with simple issuance and actions to complete.

The complexity is from the *interpretation of the contract within law* and not just the coding of the contract itself.

A DAO and The DAO—The danger of conflating definitions and terms

The principle of running a company as a decentralised autonomous organisation is indeed an interesting one, as arguably a company is little more than a series of rules, conditions and contracts on what should happen on a given occurrance.

The DAO should not be confused with a DAO, it would be like someone starting a company called "The Webserver" before the term was commonly adopted; a failure infects the term itself, and potentially the technology, even though that is not necessarily where the issue lays.

The legal standing of the DAO and the legal finality of the way in which the software executes is an important one. The contributors to the DAO entered in to a code based application of the technology. The counterparties agreed on the principle of the software and how this was supposed to function, the details of which were shared prior, during and post the funding.

So why should the "stolen" Ether be considered as such?

By the very nature of the "smart" contract, the counterparties agreed to the code that would manage the funds and therefore an exploit of that functionality is an exploit of the agreement—the code equivalent of a legal oversight.

Law still applies to this event in the real world, it is not outside the law. This raises many questions around liability and accountability of

the implementation of the system and who could be sued to recover the lost value.

The key is much of this is currently untested in court, however that is not to say that there are not applicable precedents in existing law.

So what's the point? Where is the disambiguation bit?

Contracts run along a spectrum. For me there are probably 4 or 5 different types of "contract";

1. Executable computer code—This is *not* a contract, it is just some code that will follow instructions
2. Executable computer code with a legal rights and obligations—This is closer to being a contract and provides a legal interpretation of how the code is *expected* to act
3. A legal contract with full execution of code implementation—subtly different to the previous example as the legalese is not being used to describe the logic, but the logic is looking to reflect the legalese
4. A legal contract with partial code implementation—You take an existing contract, identify the parts that can be automated using code, and then implement just those parts. The majority of the contract is in legal prose and the underlying parameters are focused on instantsiation of the contract rather than execution of the contract on a given
5. A legal contract with negotiation—This has nothing to do with executable logic, other than the ability to edit and negotiate in a live environment in a shared environment, think Googledocs for Contracts, something like Clausematch is a good example of this

(The Barclays' Smart Contract Templates team have been primarily focused on fourth & fifth of these and exploring the research behind the third).

The above is an attempt to break down the sliding scale of so-called smart contracts as I believe that the way in which the term is often used is inaccurate and stifling for innovation. By articulating what is actually meant more clearly, a much better appreciation of the

capabilities and governance underpinning different types of implementations should be available for consumers and regulators alike and will hopefully lead to further discussion and exploration of this topic, and I would welcome your thoughts on the above.