August 2017

ISDA Linklaters

Whitepaper

Smart Contracts and Distributed Ledger – A Legal Perspective

CONTENTS

EXECUTIVE SUMMARY

Smart contracts and distributed ledger technology (DLT) are increasingly being seen as a way for the derivatives industry to realise operational efficiencies and cut costs. With these new technologies potentially transforming how derivatives are executed and managed through the entire lifecycle, it seems the derivatives market is on the cusp of significant modernisation.

But these technologies are at the relatively early stage of development, and there is still a lack of agreement on what a smart contract is, what role it can play in the derivatives market, and how it might interact with existing legal standards and documentation. Could a smart contract ultimately replace an existing legal contract in its entirety, or will it only automate the execution of certain actions specified within the contract?

This paper sets out what smart contracts and DLT are, how they might be used in a derivatives context, and what the legal considerations are. Key points include:

- There is a difference between smart contract code, which refers to code that is designed to execute certain tasks, and a smart legal contract, which refers to elements of a legal contract being represented and executed by software.
- Certain operational clauses within legal contracts lend themselves to being automated. Other non-operational clauses – for instance, the governing law of a contract – are less susceptible to being expressed in machine-readable code. Some legal clauses are subjective or require interpretation, which also creates challenges.
- A possible near-term application of a smart contract is for the legal contract to remain in natural legal language, but for certain actions to be automated via a smart contract.
- This would require those actions – for instance, payments and deliveries – to be represented in a more formal, standard way within the ISDA Definitions, enabling them to be read by machines.
- Transaction data could be held on a permissioned, private distributed ledger that would be available to regulators. This would ensure there is a single, shared representation of each trade.
- Industry wide standards are required to ensure smart contracts are interoperable across firms and platforms. ISDA is working to develop these standards.

INTRODUCTION

Derivatives infrastructure has become complex and costly, but new technologies offer the potential to create efficiencies

The regulatory changes instigated by the Group-of-20 nations in 2009 have affected virtually all areas of the transaction process, from pre-trade execution to lifecycle management and reporting. This has added new layers of complexity to an already inefficient derivatives ecosystem, which is putting derivatives participants under considerable strain.

The industry is now at a critical juncture in its efforts to develop and adopt improved derivatives processes. New technologies allow a fundamental reshaping of derivatives infrastructure, which could reduce operational risks, streamline increasingly cumbersome and time-consuming processes, and cut costs. If the industry does not embrace this potential, the derivatives infrastructure stands to become increasingly costly, risky and inefficient.

A recent ISDA whitepaper[1] highlighted that smart contracts and DLT could play a meaningful role in the development of this reshaped derivatives infrastructure.

This paper takes a detailed look at smart contracts, the scope for their potential application to derivatives and ISDA documentation, and the relationship with DLT[2]. It also highlights other potential uses of technology that could facilitate a more efficient legal documentation infrastructure.

DEFINING SMART CONTRACTS

There are different interpretations of the term smart contract, and this can cause confusion

There are many competing conceptions of what a smart contract is. Those from a computer science background often use the term in a quite different way to lawyers. For lawyers, the term 'contract' connotes a very particular legal relationship of obligations, whereas computer scientists tend to think of smart contract more in terms of code.

Reflecting this different usage, Stark[3] presents two distinct schools of smart contracts:

- *Smart legal contracts:* This school resonates most with lawyers. This is where the term smart contract is used to refer to legal contracts, or elements of legal contracts, being represented and executed by software.

[1] The Future of Derivatives Processing and Market Infrastructure, ISDA, September 2016: http://www2.isda.org/attachment/ODcwMA==/Infrastructure%20white%20paper.pdf

[2] As this paper considers further, DLT is the general term used to refer to methods of maintaining distributed ledgers (or records) on computers. The reference to 'distributed' is because the record is held by each of the computers on the network

[3] Stark, J. (2016). Making sense of blockchain smart contracts. http://www.coindesk/com/making-sense-smart-contracts/. Also cited in Clack, C., Bakshi, V. & Braine, L. (2016, revised March 2017). Smart Contract Templates: foundations, design landscape and research directions

- *Smart contract code:* The other school relates less to contracts as a lawyer would understand them, and more to a piece of code (known as a software agent) that is designed to execute certain tasks if pre-defined conditions are met. Such tasks are often embedded within, and performed on, a distributed ledger. For example, one well-known smart contract implementation describes software agents that create cryptocurrency, provide an electronic voting mechanism and offer an electronic blind auction mechanism as smart contracts[4].

These distinctions can cause confusion when the topic of smart contracts is discussed, and there is a risk that lawyers and computer scientists simply talk at cross purposes. But rather than viewing smart legal contracts and smart contract code as two separate domains, the reality is there is a relationship between them. For a smart legal contract to be implemented, it will need to embed one or more pieces of code designed to execute certain tasks if pre-defined conditions are met – that is, pieces of smart contract code. Smart legal contracts, therefore, are functionally made up of pieces of smart contract code – but, crucially, under the umbrella of an overall relationship that creates legally enforceable rights.

As a result, every smart legal contract can be said to contain one or more pieces of smart contract code, but not every piece of smart contract code comprises a smart legal contract.

Highlighting the distinction between smart legal contracts and smart contract code is useful in ensuring a clarity of usage, but it does not result in a basic definition. Recognising there is no single universally accepted definition, it is nonetheless useful as an organising concept to set out a basic description that tries to present a unified view of what the term smart contract encapsulates. One definition that performs this job well is that of Clack, Bakshi and Braine[5]:

> *A smart contract is an automatable and enforceable agreement. Automatable by computer, although some parts may require human input and control. Enforceable either by legal enforcement of rights and obligations or via tamper-proof execution of computer code.*

As Clack, Bakshi and Braine note, this definition has the advantage that it is broad enough to cover both smart legal contracts and smart contract code. It captures what seems to be the fundamental essence of all conceptions of smart contracts: the automation and self-execution (and thereby enforcement) of a pre-set conditional action[6].

A critical point to recognise, however, is that a smart contract is not the same thing as a legal contract. Indeed, an aphorism often repeated is that the term 'smart contract' is a misnomer because, in many cases, a smart contract is neither smart nor a contract.

[4] The smart contract implementation referred to is Solidity. Solidity is a programming language designed to enable smart contract code to be deployed on the Ethereum platform. Ethereum is a decentralised, blockchain-based platform. On its website, Ethereum is described as a platform "… that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference. These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middle man or counterparty risk." See: https://www.ethereum.org/

[5] Clack, C., Bakshi, V. & Braine, L. (2016, revised March 2017). Smart Contract Templates: foundations, design landscape and research directions

[6] For completeness, it is worth giving some specific examples of what, on any typically used conception of the term, a smart contract is not. Some argue that the mere fact a legal contract is viewed and represented electronically (for example, as a PDF or on a screen in HTML) makes it a smart contract. This is not correct, as there is nothing 'smart' about such a contract. Perhaps more debatably, some would argue that an e-negotiating tool (ie, a tool that electronically matches certain pre-set acceptable parameters set by both parties) is an example of a smart contract. While useful, and possibly an electronic tool that will be used in conjunction with smart contract technology in the future, we do not believe such an e-negotiation tool is an example of a smart legal contract, because the key element of a smart legal contract is that it is the performance of the contract itself (rather than its formation) that is automated

To a lawyer, a legal contract has a distinct meaning:

> *A contract is an agreement giving rise to obligations which are enforced or recognised by law. The factor which distinguishes contractual from other legal obligations is that they are based on the agreement of the contracting parties*[7].

For a contract to be valid, legal systems will impose certain requirements. Under English law, there are four key elements that must (usually) be satisfied: (i) one of the contracting parties must make an offer to contract and the other(s) must accept that offer; (ii) there must be 'consideration' for the offer, this being some form of value that must be exchanged; (iii) the parties must have an intention to form legal relations; and (iv) there must be certainty as to terms of the contract. Other systems of law may have other requirements – for example, under New York law, many legal contracts must be in writing and signed to become binding.

For a smart legal contract, there would need to be a legal contract satisfying the requirements of the relevant governing law, but with some element of that legal contract being electronically automated. With smart contract code, in contrast, there might exist no legal contract at all.

Take the example of a software agent that is formulated so a pre-defined amount of an asset is moved from an account of one person (A) to another (B) if a pre-determined condition is met. This software agent does not create legal obligations between A and B. It does not impose a legal obligation on A to transfer the asset; it simply provides that a transfer will take place if the relevant condition is satisfied. This begs the question of why A would initiate such a software agent if not to satisfy a legal obligation to B[8]. And if the software agent is designed to satisfy a pre-existing legal obligation, then that would seem to be an instance where smart contract code facilitates a smart legal contract, as previously described.

If the word 'contract' in the term smart contract can have different interpretations, what about the word 'smart'? The use of 'smart' in smart contracts refers to the fact that some element of the smart contract is automatic and self-executing in accordance with pre-defined conditions. The smart contract can examine whether certain states have occurred and, if they have, can trigger a pre-determined action. This would fail the Oxford English Dictionary definition of smart as "having or showing a quick-witted intelligence".

Taking a slightly more functional definition, Demis Hassabis, the co-founder of the artificial intelligence company Deep Mind, says: "At its core, intelligence can be viewed as a process that converts unstructured information into useful and actionable knowledge."[9] Again, a smart contract does not possess that type of intelligence. All it does is execute pre-programmed steps.

---

[7] The Law of Contract, G.H. Treitel at 1-001

[8] What would be necessary in order to turn this smart contract code from mere code into binding legal obligations? Under English law, the four key elements for a contract must be satisfied. First, there needs to be offer and acceptance. The initialisation of the software agent does not represent offer and acceptance by itself. However, what if A sends B a message saying, "I offer to you that I will initialise this software agent and the actions of the software agent represent my legal obligations to you. Do you agree?", and B then responds "I agree"? That would satisfy the requirement for offer and acceptance. Secondly, there must be consideration (or value) that passes between the parties. In the example given, there are only one-way obligations: A has received nothing in return for its transfer of an asset to B. In such case, there would be no consideration. But what if the code is modified so it now automatically moves a pre-defined amount of an asset from B to A on the date the code is initialised? This effectively represents the amount B is paying to receive the benefit of the software agent's actions when it moves an asset from A if certain conditions are met. In such a two-way case, consideration would be present as both parties derive value. In that instance, the offer from A would need to be recast as: "I offer to you that I will initialise this software agent and the actions of the software agent in moving assets from my account represent my legal obligations to you, provided that you agree that the actions of the software agent in moving assets from your account represent your legal obligations to me. Do you agree?" If B agrees to this, a legal contract would have been formed. There has been offer and acceptance, there is consideration, there is a clear intention to create legal relations, and the code would give certainty of terms. At this point, the smart contract code also becomes a smart legal contract

[9] Financial Times, April 22, 2017, The mind in the machine: Demis Hassabis on artificial intelligence

DISTRIBUTED LEDGER TECHNOLOGY

Distributed ledger allows for a single shared representation of a trade to be stored centrally

DLT is the general term used to refer to methods of maintaining distributed ledgers on networks of computers. The term 'blockchain' is often used as a synonym for DLT, but a purist (or perhaps a pedant) would note this is not strictly accurate, as blockchain represents one type of DLT. Many of the best-known instances of DLT are based on a blockchain approach[10], but there are other types that are not built on blockchain technology.

What then is a distributed ledger?

It is a digital record that is shared instantaneously across a network of participants. It is distributed because the record is held by each of the users (or nodes) on the network and each copy is updated with new information simultaneously. DLT uses a consensus technique to ensure that every node agrees on the record, with different distributed ledger technologies using different consensus methods. A key advantage of DLT is that there are not multiple competing sets of records that need to be reconciled but just one, albeit maintained on multiple nodes. This one record represents a golden source of data.

The following description gives more colour (this description is of a blockchain distributed ledger but many of the attributes described would also apply more generally to different types of DLT):

> *A blockchain is a globally shared, transactional database. This means that everyone can read entries in the database just by participating in the network. If you want to change something in the database, you create a so-called transaction which is required to be accepted by all others. The word transaction implies that the change you want to make (assume you want to change two values at the same time) is either not done at all or completely applied. Furthermore, while your transaction is applied to the database, no other transaction can alter it.*
>
> *As an example, imagine a table that lists the balances of all accounts in an electronic currency. If a transfer from one account to another is requested, the transactional nature of the database ensures that if the amount is subtracted from one account, it is always added to the other account. If due to whatever reason, adding the amount to the target account is not possible, the source account is also not modified.*
>
> *Furthermore, a transaction is always cryptographically signed by the sender (creator). This makes it straightforward to guard access to specific modifications of the database. In the example of the electronic currency, a simple check ensures that only the person holding the keys to the account can transfer money from it*[11].

[10] A 'blockchain' distributed ledger works by storing data in distinct container data structures known as 'blocks'. Each block is linked back and refers to the previous block in the chain. A blockchain is often conceptualised as a vertical stack of blocks, with each block being piled on top of each other in chronological order. The blockchain methodology came to the fore with the digital cryptocurrency Bitcoin, which uses a blockchain as one of its core elements. The blockchain enables the peer-to-peer, distributed nature of Bitcoin – there being no central record of who holds Bitcoin, but instead a distributed ledger held by all nodes

[11] This description is from the documentation provided by Ethereum with respect to the Solidity scripting language previously referred to. See: http://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html

An additional distinction worth noting is that distributed ledgers can either be 'permissionless' or 'permissioned', which usually also corresponds to being 'public' or 'private'.

Many of the well-known distributed ledgers (such as the blockchain that supports the digital currency Bitcoin) are permissionless and public, in the sense that anyone can participate in the network and will be able to see all the data[12]. There is also no entity that could be said to operate the distributed ledger (in the sense of having super-administrator-type rights).

Most work in the financial markets has been on distributed ledgers with membership restricted to market participants in that space, and to other interested parties (such as regulators). These are known as permissioned and private networks. They often have an entity or a group of entities that have some override or super-administrator-type rights over how the distributed ledger operates. This is useful for ongoing management of the distributed ledger, but introduces a degree of centralisation to what is otherwise a distributed network.

In addition, there is a commercial sensitivity around certain data relating to financial transactions. Participants do not necessarily want everyone else to be able to see every piece of data on the network. As a result, types of permissioned/private distributed ledgers have evolved where the data on the ledger is masked in such a way that they can only see certain data relating to them and their transactions (although certain entities, such as regulators, might have access to all data).

SMART CONTRACTS AND DLT

Smart contracts and DLT are distinct technologies, but DLT provides a central platform on which smart contracts can be executed

Smart contracts and DLT are often talked of in the same breath but are distinct technologies – albeit complementary and, in some instances, symbiotically linked[13]. However, the advent of DLT has created a platform on which smart contracts can be hosted and executed. This has brought smart contract concepts into the mainstream.

One of the most powerful inter-relationships between smart contracts and DLT can be seen in answer to the following question: computers that can automatically execute code have been around for ages, so why haven't smart contracts been previously developed?

It is true that the automation of certain elements of existing legal contracts has been feasible for some time. Since the advent of computers in a mainstream context, it would have been perfectly possible (at least technically) for contracting parties to program a computer to automatically execute an event, such as a payment under a contract upon satisfaction of pre-defined conditions (for example, that a stock price hits a pre-defined price).

[12] It is important to note that while all data on a permissionless, public distributed ledger can be viewed by all full nodes, the data might be anonymised in the sense that it cannot be linked to a particular individual or entity. With Bitcoin, for example, it is possible for every node to calculate how much Bitcoin is associated with a Bitcoin address by viewing all historical Bitcoin transactions to and from that address (a Bitcoin address is a string of numbers and letters that is allocated to each user – such as 133gWkwggNqWNLrRAbiJ1k4ke14TGSFVQH), but it is not possible to see the individual identity of the holder of that Bitcoin address because that data is not stored on the blockchain

[13] The precise relationship refers back to the discussion on the different types of smart contracts. Certain instances of smart contact code, for example (and by way of contrast with smart legal contracts), are inextricably linked with DLT and would not be able to have any independent existence. The Ethereum platform cited earlier is one such example because the manner in which the smart contract code works is via 'transactions' occurring on the Ethereum blockchain

Practically, though, this would have meant both parties needing to have programmed their own computers and to have been running separate instances of the program on their own systems.

In addition, one party might not be willing to rely on the other party's code, and so each would code their own version of the relevant provisions. This introduces a risk that the two implementations differ in practice[14].

DLT allows the code to be embedded in the distributed ledger. This means there is only one 'golden' version, which effectively binds both parties. More importantly, once the code is switched on, the parties can take comfort from the fact that it will self-execute automatically and neither party can tamper with that. This is what is meant when smart contracts are described as 'self-enforcing'. There should be no need to resort to the courts to enforce the legal contract for payment because, when the relevant event occurs, failure to pay is not something that can happen within the code[15].

It is important to remember, though, that smart contracts and DLT are distinct and come with their own respective challenges. Confusingly, in the legal context, these often get intertwined.

One example for DLT is the issue of situs. Under various legal regimes, it is necessary to identify the location of an asset or contract to determine the applicable legal jurisdiction for various legal questions relating to it – for example, whether a property right has been created. In the case of dematerialised financial assets where ownership is recorded on a register, it is often the place where the register is held or where the registrar is situated that is deemed to be the situs of that financial asset. However, the distribution of the register across nodes in multiple jurisdictions raises a seemingly intractable problem – under current legal principles at least – as to where the situs should be.

This point is often raised by experts as evidence that DLT and smart contracts suffer from legal issues relating to situs. It is important to note, though, that this legal issue is one relating to DLT rather than smart contracts. To be sure, smart contracts raise their own legal issues (although the degree to which such issues arise depends on the conception of what a smart contract is), but they are distinct issues from those that arise for DLT. There isn't an analogous situs problem for smart legal contracts themselves.

[14] In the financial contracts space, this can be seen in certain bespoke transactions where the payoff depends on the evolution of a complex algorithm over time. The algorithm is set down in detail in the transaction documentation, but both parties often then code their own representations of the algorithm on their own computer systems (ie, because each party has separately programmed it, they will have different code scripts even though they are trying to implement the same thing). This raises the risk that the two implementations differ in their outcomes. Often, parties try and limit the scope for difference by ongoing reconciliations of the outputs of their respective code implementations

[15] This is subject to the obvious qualification that the party making the payment needs to have sufficient funds to make such payment and those funds need to have been made available to the smart contract. In many descriptions of smart contracts, this issue is 'resolved' (in some sense of the word) by saying that the performing party must have pre-funded an account with the maximum funds that might be required to be paid. In such case, the smart contract would simply take the required amount of funds from that account upon its self-execution. Such pre-funding would not be commercially feasible for many financial transactions, but financial derivatives do potentially fit more neatly into this self-executing vision because of the margining of many trades (albeit there is always the risk that the margin provided is insufficient given the time to transfer margin and the potential for market value moves)

OPERATIONAL AND NON-OPERATIONAL CLAUSES

Certain operational clauses within legal contracts are more suitable to automation and self-execution than non-operational clauses

Before considering what smart contracts might mean in the context of a legal agreement (and the ISDA documentation), it is necessary to distinguish between different types of clauses within legal agreements. Not all clauses are susceptible to automation and self-execution. Even where a clause might technically be capable of being automated, it might not always be desirable to automate it.

A legal agreement[16] can be analysed as containing operational and non-operational clauses[17].

Operational Clauses

Operational clauses generally embed some form of conditional logic – ie, that upon the occurrence of a specified event, or at a specified time, a deterministic action is required. Such clauses are at the heart of any financial contract. Examples include:

- A clause that requires an amount to be payable on a payment date equal to the product of a calculation amount, a floating rate (plus or minus a spread) and a day count fraction[18];
- A clause that requires an amount to be payable on an exercise date equal to the number of options exercised multiplied by a strike price differential[19];
- A clause that provides that one party to the contract pays the other an amount equal to the difference between the settlement price and a forward price, with the party required to make such payment being determined by whether the settlement price exceeds the forward price or vice versa[20]; and
- A clause that requires a party to transfer assets on a particular date that have a value equal to the amount by which a required credit support amount is less than the value of collateral provided, subject to certain formulaic haircuts and adjustments[21].

[16] In this paper, we generally use the terms 'legal agreement' and 'legal contract' to refer to the terms of the legal agreement between the parties as they are set down in writing in the body of the written agreement. In many jurisdictions, it would be possible for a court to imply additional terms into the legal agreement between the parties (depending on context), and such implied terms, together with the terms of the written agreement, might be said to be the complete legal agreement or legal contract

[17] Clack, C., Bakshi, V. & Braine, L. (2016, revised March 2017) also use this distinction. Al Khalil, F., Ceci, M., O'Brien, L. & Butler, T. (2017). A Solution for the Problems of Translation and Transparency in Smart Contracts instead use a distinction between the operational semantics of a smart contract, its denotational semantics, and its legal, business and regulatory semantics. We have adopted the simpler operational versus non-operational distinction as it is universal and does not carry with it the computer science connotations that talking of operational, denotational and legal, business and regulatory semantics bring

[18] Section 6.1 ISDA 2006 Definitions

[19] Sections 8.1 and 8.2 ISDA 2002 Equity Derivative Definitions

[20] Sections 8.4 and 8.5 ISDA 2002 Equity Derivative Definitions

[21] Paragraph 2 ISDA 2016 Credit Support Annex for Variation Margin (VM) (English law)

Non-operational Clauses

Operational clauses can be contrasted with clauses that do not embed such conditional logic but that, in some respect, relate to the wider legal relationship between the parties. Examples include:

- A clause specifying what law should govern in the event of any dispute[22];
- A clause specifying what jurisdiction any disputes may be brought in[23];
- A clause providing that the written legal document represents the entire agreement between the parties[24];
- A representation that a party's obligations under the legal agreement constitute legal, valid and binding obligations[25];
- A clause that dictates that when making a decision or a determination, the person making the calculation shall do so in good faith and in a commercially reasonable manner[26]; and
- A clause that provides that all transactions entered into under a master agreement form a single agreement between the parties[27].

Such clauses are patently of a different nature to operational clauses, although the boundary line is sometimes difficult to draw.

Operational clauses are readily capable of being expressed as Boolean logic[28]. As a result, they are highly susceptible to being machine-automated or analysed in some way – a point that will be discussed later.

Non-operational clauses are less susceptible to being expressed in pure Boolean logic. However, this does not mean they could not be expressed in some more formal manner that would allow computer software to interact with them in a useful manner. This crosses into the territory of ontologies and formal semantics. Ontologies are a formal way of defining the structure of knowledge of a domain and the relationships between concepts. By modelling the information and relationships that are contained in a contract in a formal manner, it raises a wider range of possibilities for a useful application of computer technology to such a contract.

[22] Section 13(a) ISDA 2002 Master Agreement

[23] Section 13(b) ISDA 2002 Master Agreement

[24] Section 9(a) ISDA 2002 Master Agreement

[25] Section 3(a)(v) ISDA 2002 Master Agreement

[26] For example: the definition of close-out amount in the ISDA 2002 Master Agreement is: "Any Close-out Amount will be determined by the Determining Party (or its agent), which will act in good faith and use commercially reasonable procedures in order to produce a commercially reasonable result." Section 4.14 2006 ISDA Definitions: "Whenever the Calculation Agent is required to act, make a determination or to exercise judgment in any other way, it will do so in good faith and in a commercially reasonable manner"

[27] Section 1(c) ISDA 2002 Master Agreement

[28] Named after George Boole, a 19th century English mathematician who devised a system of mathematical logic designed to reduce complex processes to simple, tractable equations. In his system, variables are either 'true' or 'false'. Boolean logic forms the basis of modern computers and of computer programming languages. In computer programming languages, conditional statements perform different computations or actions depending on whether a Boolean condition is true or false. This often gets expressed in the form of if – then – else statements – ie, if Boolean condition is satisfied then [do something] else [do something else]

Take the example of a standard representation from a party that it is duly organised and validly existing under the laws of the jurisdiction of its organisation or incorporation[29]. This is not a statement of conditional logic, and so would not be susceptible to pure Boolean logic. It is a representation of a legal state. But if there were a sufficiently developed ontology for legal contracts, it would be possible to conceive of a world where a computer could understand what is meant by the terms 'party', 'duly organised', 'validly existing', 'jurisdiction' and 'organisation and incorporation', and could check automatically with relevant company registries whether this representation is correct at the time it is given.

This example is illustrative, however, because it immediately highlights some of the issues that would have to be resolved to arrive at a common ontology. What precisely does 'duly organised' mean? Is it a synonym for 'validly existing'? What is the difference between 'organisation' and 'incorporation' and do we need both terms? Who would develop such ontology and would there be common standards across all contracts, no matter of what type or legal system?

Limits of Formal Representation

Stepping back, the representation of legal terms in more formal logic, whether Boolean logic or some wider representational formalism, does seem to have limits – both on how feasible and how advisable it is.

One challenge is that certain legal terms would seem incapable of being formally represented in a non-ambiguous way (whether as Boolean logic or some wider formalism) because of their ultimately subjective nature. If a contract requires that a determination is made in 'good faith' and a 'commercially reasonable manner', it is clear these terms have a legal meaning, but they are certainly not Boolean (ie, having a discrete yes/no interpretation). Different legal regimes will have different interpretations as to what these terms might mean, and the interpretations are often heavily contextual and driven by facts and circumstances.

But, it might be argued, this is precisely the problem. Wouldn't it be better to draft the contract in such a way that all different permutations are anticipated? Isn't wording like 'good faith' and 'commercially reasonable' merely the last resort of a lazy lawyer? One could easily see how a computer programmer, used to dealing with programming languages that admit no such ambiguity, might take that view.

That would ignore that it is extremely difficult, if not impossible, to conceive all the possible permutations that might occur in the future with respect to a legal relationship between two parties[30].

[29] Section 3(a)(i) ISDA 2002 Master Agreement

[30] This is perhaps the most fundamental challenge a lawyer might pose to a computer scientist regarding the merits of smart legal contracts. The law has evolved a finely-honed system that derives the meaning of clauses not just from the language used, but also from the factual background and the commercial purpose. Courts often exercise judgment as to what the parties intended. If a legal obligation is created through a smart contract where everything is determined by reference to the language, you replace the normal rules of contractual interpretation with literalism. Lawyers might argue this would only work if the drafting were perfect, which is not realistic, and it removes some of the checks and balances that are normally available in interpreting a contract. A computer scientist might respond that the lawyers' fondness for ambiguity is part of the problem, and being forced to define things much more rigorously would be no bad thing. In support of that contention, they might adduce evidence to the effect that computer code, in its many applications, has to deal with many complex scenarios with many permutations, and that it is perfectly feasible to be literal when dealing with complex systems. Regardless of the rights or wrongs of the argument, it is true that the greater the elements of a contract that are coded, the more traditional rules of contractual interpretation are replaced by literalism

For clarity's sake, there is clearly a benefit in spending some time and effort in trying to predict, and explicitly provide for, the most likely permutations. At some point, though, the cost-benefit analysis starts to look less favourable. It is here that a lawyer often reaches for a more sweeping fallback position, such as certain determinations having to be made in good faith and a commercially reasonable manner, or is simply silent on the matter. In either case, the contracting parties are relying on the fact that the courts could be relied upon to provide a contextual interpretation of how the issue should be resolved, informed by the long-standing bedrock of legal principle.

More fundamentally, it is very difficult (and perhaps not advisable) to strip a legal system of obligations of all ambiguity. The nuances of complex relationships can be difficult to define, let alone to reduce to paper, and words can mean different things to different people. As an English judge pragmatically noted:

> *The words used may, and often do, represent a formula which means different things to each side, yet may be accepted because that is the only way to get 'agreement' and in the hope that disputes will not arise*[31].

A more practical limit to seeking a wider formal representation is that, even where it would be theoretically possible to represent a legal clause in a more formal way, it does not necessarily mean it should be. There are quite likely to be clauses where there is simply no benefit in trying to subject the clause to such formalism, because there is nothing useful that a computer is likely to be able to do with it. Take the single agreement provision in Section 1(c) of the ISDA 2002 Master Agreement. The contractual relationship is represented by a combination of the ISDA Master Agreement, and the schedule, credit support documentation and confirmations that are entered into under the Master Agreement from time to time. These documents are together stated to form a single agreement between the parties. The idea is to ensure that each party has a net exposure to each other. This is one of the most critical legal provisions in the entire agreement, but it is difficult to conceive of any benefit in trying to represent this provision in a more formal way.

Finally, it is worth noting that, even if the contract is self-executing in the sense that the pre-specified action happens automatically, it does not necessarily mean the relevant action will be legally final in all circumstances. For example, it may turn out that performance has become illegal, or that a payment can be set aside on insolvency grounds. It will not always be possible to make an automatic assessment as to whether an act was illegal or an entity was insolvent at the time of the relevant action. The applicable information might not yet be available or, more fundamentally, certain laws might have a retrospective effect. Admittedly, this is also the case with traditional contracts, but it is worth explicitly noting that smart legal contracts do not (and cannot) change this.

DIFFERENT MODELS OF SMART LEGAL CONTRACTS

Set out below are two contrasting models of smart legal contracts dubbed the external model and the internal model. It is assumed the most likely implementations of smart contracts in the near term will relate to operational clauses and Boolean logic, rather than to non-operational clauses.

[31] Lord Wilberforce in Prenn v Simmonds [1971] 1 WLR 1381 at 1385

External Model

Smart contract code could either be entirely separate from the legal contract, or the legal contract could incorporate some form of code in certain areas

In the external model, the legal contract would remain in its present form (ie, a natural human language document), but, external to the legal contract, certain conditional logic elements of the legal contract would be coded so the required actions happen automatically when the relevant conditions are satisfied.

The code would not be part of the legal contract; all it would do would be to provide a mechanism for the automatic performance of a contract written in a natural human language. If there were any difference between what happens when the code executes and what the legal contract requires, the legal contract would take precedence. In the external model, therefore, the code would not itself legally bind the parties and would not remove legal ambiguity. Accordingly, the parties would need to be comfortable that the code accurately reflects their obligations in the natural human language legal agreement.

In many respects, at least for lawyers, the external model is only a small step further than the operational mechanics derivatives counterparties already have in place. Indeed, there are areas where such automation already takes place – for example, daily collateral flows are already automated in the manner described above in certain margining arrangements. That is not to belie the potential impact a widespread adoption of the external model might have operationally, but merely to note that it would likely be a fairly inconsequential step for lawyers.

In the external model, therefore, it is not the contract itself that is 'smart', but rather the code building blocks that would accompany it and would be used to execute it.

Internal Model

In the internal model, much of the legal contract would likely remain in its present form but, critically, with certain conditional logic elements of the legal contract rewritten in a more formal representation than the current natural human language form. A computer would then take that more formal representation and execute the conditional logic automatically.

The written contract would, as a result, look like a hotchpotch of approaches. Certain clauses would be drafted in natural human language, as is the case today. But other clauses would effectively be set down on the page in some form of code, or other formal representation. Alternatively, instead of setting down the code or formal representation within the written contract itself, the written contract could refer to an identified piece of code stored elsewhere and could state that such code is to be given legal effect between the parties[32].

[32] There are several examples currently where long-term contracts have various payments calculated by reference to the output of some form of model that has been pre-agreed between the parties and that is kept as a piece of software by all parties. The contracts provide that whatever outputs the model produces are to be taken as determinative of the parties' obligations. For example, if the model contains a formula to calculate an amount that might be payable from Party A to Party B, the legal contract will have a simple clause where Party A promises to pay Party B the amount produced by the model

What is Formal Representation?

For a purist (at least from a computer science point of view), a formal representation might be computer code in a high-level programming language[33]. Such a representation would need to follow the strict syntax of that programming language. As programming languages differ, the same piece of conditional logic would result in different code for different languages.

The multiplicity of programming languages used in practice may make it unpalatable to pick any one language. In addition, it may be difficult for the average lawyer to pick up a piece of code in a programming language and understand exactly what it is seeking to do[34]. Even if a lawyer is prepared to learn the programming language, these have not been designed with legal drafting in mind, which would make it difficult for a lawyer to translate a traditional drafting approach into code.

One way to solve these issues would be for a new programming language to be devised (perhaps based on extensions to a current language) that is designed to more intuitively follow the flow and terms of legal drafting. Lawyers would need to learn this programming language to be able to draft smart contracts. There are current examples of languages that have been devised to facilitate smart contract code (such as Ethereum's Solidity), but these are not the most intuitive for lawyers and are also often closely associated with actions on a particular DLT rather than being interoperable across many DLTs. However, it must be recognised that most live instances of smart contracts (or at least, smart contract code) use these specific languages, and these have a certain amount of traction. Whether they can evolve to facilitate smart legal contracts in a manner that meets the needs of the legal community remains to be seen.

An alternative approach would be for the more formal representation to be a bridge that attempts to straddle the natural language legal drafting that lawyers use, and the strict formalism and full syntax of a programming language.

[33] What do we mean by a high-level programming language? To understand this, we need to distinguish between machine code, low-level programming languages and high-level programming languages.

For a computer to do something, it needs to receive its instructions in machine code. Such code is in the form of binary data, but is normally coded in a more readable decimal, octal or hexadecimal form that is translated into a binary data form. Machine code operates directly at the level of the computer's central processing unit, dealing with things like registers, memory addresses and call stacks. Because machine code is operating at the CPU level, different processors or processor families will have different machine code instruction sets. Programming directly in machine code is difficult, time consuming, laborious and prone to error.

Because of this difficulty, programming will normally take place in a programming language. Programming languages can be divided between low-level programming languages and high-level programming languages.

Low-level programming languages (also known as assembly languages) are one level up from the machine code itself. There is normally a very strong correlation between the instructions in such language and the machine code language. For this reason, as with machine code, assembly languages are specific to a particular processor. Assembly language is converted into machine code by a program called an assembler. Assembly languages are not readily comprehensible to a non-computer scientist and are still seen as operating very close to the level of the machine.
By contrast, high-level programming languages operate at a higher level of abstraction and tend to use natural human language elements. They will use more abstract computing concepts, such as variables, arrays, Boolean expressions, loops, and so on. They are generally environment independent in the sense that a program written in a high-level language is capable of being used across multiple processor or processor families, as programs can be run that convert code in the high-level language to the particular processor or processor family. Whether a particular programming language is high-level or low-level will depend upon where it lies on a relative spectrum. Non-exhaustive examples of programming languages typically considered to be high-level would be Python, Visual Basic, Perl, PHP, Java, C++ and Ruby. Programs written in a high-level language must be translated into machine code by a compiler or an interpreter

[34] Albeit that certain higher-level languages (for example, Python) are explicitly designed to be more intuitively readable than more traditional programming languages

Al Khalil, Ceci, O'Brien and Butler (2017)[35] propose that a common language would be needed between the legal draftsperson and the computer developer[36]:

> *It is also clear that the interaction and communication between both actors should be governed by a common language. This should not be the controlled natural language of the computer scientist. Rather, the lawyer should author contracts in a controlled legal natural language (LNL) that is logical, clear, unambiguous, and comprehensible by a computer programmer, while being as close as possible to representing the denotational semantics[37]. It could then be employed by the computer programmer as a specification guiding the technical implementation.*

They set out several properties that such a controlled legal natural language would need to possess. These include not alienating the lawyer and being as close as possible to the language of contracts that the lawyer is used to, but still possessing an unambiguous grammar.

In some ways, this seems similar to the development of a new programming language for smart contracts, albeit seemingly operating at a slightly higher level of abstraction and thereby making it more understandable and useable for lawyers.

Code is Contract

With the internal model, the code element of the smart contract becomes an inextricable part of the legal contract. There is no possibility of difference between the code and the relevant part of the written contract because they are one and the same.

However, there are two other potential areas of difference (or translation error) that are normally raised as potential issues:

- The first is illustrated by the question: "how do I know the code as written in the contract reflects my intentions if I cannot read it?" This is really no more than a reprise of the concern that the code, as written in the contract, might not be as readily understandable to an average reader as natural human language is today. One way of resolving this is for the lawyer to have learnt the relevant language used to write the code.

  However, not everyone who might need to review a contract could realistically be expected to have gone through such learning. An analogy can be drawn with what currently happens when contracts use different human languages. A native English speaker who has a contract that is partly written in German may engage someone who understands that language to review that section.

[35] Al Khalil, F., Ceci, M., O'Brien, L. & Butler, T. (2017). A Solution for the Problems of Translation and Transparency in Smart Contracts

[36] It is worth noting that Al Khalil, Ceci, O'Brien and Butler's conception of smart contracts is wider than purely just automating Boolean logic. Their conception also focuses on the capturing of relational data, meta-rules and powers within a legal framework using common logic and ontologies. This falls within the area touched on by this paper when discussing what ultimately might be possible with non-operational clauses, albeit that it was discussed in a more general way and with a deliberate attempt to avoid trying to frame this paper's comments in computer science and symbolic logic terms

[37] Al Khalil, Ceci, O'Brien and Butler use a computer science differentiation between operational semantics and denotational semantics. They explain that operational semantics provide a formal description of the behaviour of a computer program, whereas denotational semantics are concerned with the meaning of a computer program as a function that maps input into output. For legal contracts, they add an extra level that is the legal, business and regulatory semantics that the contract possesses

With code, this problem might be exacerbated by the relatively small number of people who might truly understand the code. This is one area where having industry standard code for particular pieces of conditional logic could help. The code would have been checked by many industry participants at the time it was put together, giving later users comfort that it had been subject to scrutiny and focus.

One counter to the above question might be the following: "how do you know that the natural human language drafting reflects your intentions?" Natural human language has many nuances and shades and can be ambiguous. How one person interprets a clause might be different to another. Admittedly, well-drafted contracts should limit the degree to which such different interpretations might be possible, but the day-to-day business of the courts suggests this is not always the case. Replacing parts of a contract with code may cause some translation issues, but natural human language drafting is itself not free from the risk of incorrectly reflecting a contracting party's intentions.

- The second potential area of difference is illustrated by the question: "how do I know the effect of the code, when executed by a machine, will be what I intend?" In other words, what if there is a glitch somewhere in or between the high-level programming language and the executable machine code that means the code does not do what it was intended to do when executed? What if the high-level programming language coding was accurate in the sense that it was written in the way that should have produced a given effect, but something has gone wrong somewhere down the line? To help guard against this, simulations can be run to observe whether the code produces the expected outputs. It is worth noting that the risk of a glitch somewhere in the programming exists in the use of any computer program.

  A different angle on the same question is whether the code has captured all the different permutations it needs to provide for. This is not really a problem for very simple pieces of conditional logic. But where there is nested conditional logic (ie, multiple layers of conditional logic), it can be difficult to be certain that all the possible variations or permutations have been dealt with, and that the computer doesn't do something odd when faced with an unexpected combination. This may be a valid point, but it is no different to the situation faced by the legal draftsperson today – albeit the legal draftsperson can sometimes use a general catch-all that relies on principles such as good faith and commercial reasonableness.

Automating Everything?

A standard objection to attempting to turn a legal contract, or a part of a legal contract, into an automated, self-executing smart contract is that there will be certain actions that the parties may not wish to happen automatically.

For example, if an event of default occurs under an ISDA Master Agreement, this gives the non-defaulting party the right to terminate[38] outstanding transactions. But the non-defaulting party might decide it does not want to exercise such a termination right at that time. The reasons for that decision tend to be subjective, depending on the commercial and relationship context at the time of the event, the nature of the default, and other external factors (eg, proceedings that may occur under applicable law because of the default). This would not seem to be susceptible to pre-programming.

[38] Section 6(a) of the ISDA 2002 Master Agreement

That does not mean a legal contract cannot have elements of it 'made smart'. It simply means these events would not be automatically triggered (although they could lead to automatic alerts, which would be useful). But a smart ISDA Master Agreement could be coded so it requires a party to make an electronic election to designate an early termination date under the agreement, and it is possible that certain consequences of such a declaration could then be automated.

Oracles

While recognising there is likely to be a need for some subjectivity, certain subjective elements could be replaced by determinations of third-party oracles.

To execute automatically, smart contracts need to be able to interface with data in the wider world. For example, a piece of conditional logic that depends on whether a particular stock price has reached a certain level would require the smart contract to be able to ascertain that stock price. To do this, it can look up the stock price from a separate data source, typically known in the distributed ledger community as an 'oracle'.

Sources already exist for simple pieces of data, but it is possible to have other entities make more complex determinations. A good example of this already exists in the ISDA documentation framework – namely, the Credit Derivatives Determinations Committee.

Under the 2003 ISDA Credit Derivatives Definitions, notice of a credit event would need to be given to trigger a payment under the transaction. Under the 2014 ISDA Credit Derivatives Definitions, however, a determination by the Determinations Committee that a credit event has occurred would result in an automatic triggering in many cases, subject to certain conditions. A notification from a party under the contract is no longer required. This has had the practical benefit of dramatically reducing the administrative burden on counterparties upon the occurrence of a credit event, as well as the attendant issues that tended to arise, such as whether a credit event notice and related documentation had been delivered correctly.

A smart contract version of a credit default swap would therefore use a determination of the Credit Derivatives Determinations Committee as an oracle.

That is just one example, but it shows the important role that oracles might play. Market participants might decide there are other determinations under the ISDA documentation that would benefit from having a similar oracle.

ISDA DOCUMENTATION

Parts of the ISDA Definitions could be re-written to give a more formal representation that facilitates automation – but standards are needed

Derivatives are fertile territory for the application of smart contracts and DLT because their main payments and deliveries are heavily dependent on conditional logic.

How, then, might smart contracts and DLT be applied to ISDA documentation? A high-level conception is as follows: a distributed ledger could store data on the derivatives transactions between parties in the network; smart-contract logic stored on the distributed ledger could provide for certain actions required in respect of those transactions to happen automatically; and the distributed ledger would reflect that these actions had taken place. The smart contract element would relate to the legal agreement and the execution of operational flows. The DLT element relates to the recording and updating of a single source of data (producing a near real-time golden source of data for derivatives transactions) and the hosting of the smart contract.

The following sets out some granular proposals of how this high-level conception might be put into effect.

- Parties still enter an ISDA Master Agreement and schedule drafted in natural human language, although execution might take place electronically[39].

- Parts of the ISDA Definitions booklets that set out the terms used in defining payments and deliveries would be re-written in a more formal representation that would be tractable by computers (or would refer to code or a more formal representation in a separately maintained code bank).

- At the time of writing, there would seem to be no obvious developed candidates for such a formal representation. Currently, most smart-contract scripting languages have been developed in the context of smart contract code rather than with explicit reference to the needs of smart legal contracts. While they may be high-level languages, they are not particularly accessible or intuitive for lawyers without a coding background. Consequently, there is a need for the development of a consistent, non-ambiguous language for the drafting of smart legal contracts. There are several existing business ontologies, vocabularies or schema that might serve as a useful base from which to leverage, and a good deal of academic research and suggestion, but one of the main bars to widespread implementation of smart contracts in the near future would seem to be the lack of any industry standard for such code. ISDA is looking at utilising existing frameworks in the smart contract space, including those provided by standards created in Financial products Markup Language (FpML), which tie back to the ISDA documentation framework[40]. Several working groups have been set up by ISDA to coordinate the various legal/documentation, regulatory, technological and reporting workstreams in order to future-proof standard documentation and align data standards. ISDA has held roundtable meetings between the various workstreams in order to ensure cross-fertilisation and coordination going forward.

[39] One suggestion of an additional feature is to use a cryptographic hash function to provide a unique identifier for a legal agreement. This would provide certainty that a legal agreement claimed by a party to be an accurate representation of the agreement was indeed the legal agreement originally entered into. A cryptographic hash function is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size. For example, one instance of a cryptographic hash function, the SHA 256 algorithm, produces a different 256-bit number for any input. There is only one output for any given input, and the output is always the same for the same input. See the suggestion regarding unique identifiers in Clack, C., Bakshi, V. & Braine, L. (2016) Smart Contract Templates: essential requirements and design options

[40] FpML is the open source XML standard for electronic dealing and processing of derivatives. It establishes a protocol for sharing information electronically on, and dealing in, swaps, derivatives and structured products

- Prior to deployment of smart contract technology, there might be a transitional step where amendments are made to ISDA Definitions booklets to make certain provisions more prescriptive, so they lend themselves to the application of conditional logic[41]. This would also give an opportunity to revisit certain provisions that currently have subjective elements to decide whether these are necessary or whether they could be recast somehow in a more prescriptive sense, or by reference to a committee decision. As a starting point, ISDA intends to look at the 2006 ISDA Definitions for interest rate and currency derivatives.

- Transaction data would be held on a permissioned, private distributed ledger. The smart contract elements of the ISDA Definitions booklets would be embedded in, and would operate on, that distributed ledger[42].

- Regulators could be given direct access to data stored on the distributed ledger.

The proposals would future-proof the ISDA documentation framework – setting up a structure that would be more capable of flexibly accommodating future evolution. By creating a direct relationship between the drafting of the smart contract provisions in the ISDA documentation and their implementation in the operational ecosystem, parties could realise efficiencies and reduce operational costs.

To proceed, progress needs to be made on developing a formal representation that can be used to re-write the relevant provisions. In tandem, technology providers need to develop DLT implementations that would use such a formal representation. Standards need to be developed to ensure interoperability across different implementations. ISDA can play a role in unlocking the value of the new smart contract technology, participating in the development of such standards, and potentially leveraging FpML to do so. Given the pace of technology, the market must be ready to fully operate in the digital space. If paper transactions become obsolete, all market participants must be primed to adapt their systems and processes accordingly.

ISDA NETTING OPINIONS

A question commonly asked is whether applying smart contract technology to ISDA documentation might render the existing ISDA jurisdictional netting opinions in some way deficient or redundant.

This should not be the case. Smart contracts are not the same as legal contracts. In any application of smart contract technology to ISDA documentation, there would still need to be a legal contract, and it would be that legal contract upon which the netting opinions would bite.

[41] For example, similar to the way in which the 2006 Definitions provided explicit formulae for a number of day count fractions in order to reduce any potential for ambiguity in the interpretation of such provisions

[42] This paper has focused primarily on the challenges of smart contract implementation, but analogous issues arise with implementing a DLT solution for derivatives. In particular, the question arises as to what DLT technology (or 'fabric') would be used. There are currently a number of proof-of-concept implementations of financial contracts on distributed ledgers, and each implement on a particular distributed ledger fabric. Examples include Axoni Core (https://axoni.com), Corda (https://www.corda.net/), Ethereum (https://www.ethereum.org/), and Hyperledger Fabric (https://github.com/hyperledger/fabric/), among others. It is unclear which distributed ledger technology is likely to be adopted in the derivatives industry, or even if multiple technologies might be used. This raises the possibility of whether ISDA might be able to assist the market by providing high-level specifications of how derivative transactions might be embedded in a DLT framework, but leaving the actual implementation to be done on a DLT by DLT basis

COMPLEMENTARY TECHNOLOGIES

There are other technological advances that might be adopted alongside smart contracts that could enhance use of the ISDA documentation. These are not, strictly speaking, elements of smart contracts and DLT, but they would be complementary and might use elements of these technologies as part of their own implementation. This paper does not propose that the derivatives industry adopt all or any of these technological advances, but rather highlights some potential avenues to facilitate discussion.

E-signing

Every contracts lawyer will have spent many hours dealing with the administrative logistics of arranging for contracts to be signed by the parties. In a world where goods can be ordered online by a simple click, the traditional method of parties executing documents in wet ink on paper is anachronistic.

One of the abiding takeaways for a lawyer experiencing DLT and smart contract demos is not just the automated execution of payments and deliveries under the transaction, but the fact that most seem to allow for electronic legal execution. The terms of a transaction are electronically proposed by one party, electronically accepted by the other clicking on an accept button, and then are embedded in a distributed ledger.

Even if derivatives contracts are not recorded on a distributed ledger or made smart in some way, electronic signatures offer several palpable advantages over their wet-ink predecessors. These include a more robust identification of the person electronically signing, and a definitive, time-stamped record of when the signatures occurred.

One factor currently inhibiting the widespread adoption of electronic signatures is a concern they might not be legally valid in certain jurisdictions. To address this, ISDA has commissioned e-contract opinions from a range of jurisdictions to assess the robustness of parties electronically forming or electronically executing contracts. The aim is to facilitate increased use of electronic signatures.

Looking forward, there is the potential to combine electronic signatures with a distributed ledger and smart contract implementation. One feature of DLT is that each participant has a unique private key they use to initiate transactions on that distributed ledger. This unique private key could act as an electronic signature and produce an indelible record that someone with access to that private key executed the transaction. It also offers the possibility that a piece of smart contract code could then also validate that person's authority against a signing authority list stored on an accessible data source. This would eliminate the need for one party to check the other party's authority lists to ensure the person signing has the requisite authority to execute. This would be easier and more accurate than the manual process followed at present.

In summary, a more widespread adoption of electronic signatures would offer significant procedural efficiencies compared to the current approach. This would be the case even without any distributed ledger and smart contract technology. If derivatives do utilise distributed ledger and smart contract technology, however, electronic signatures would seem to play an integral part in building an effective legal process within such technology.

Modularisation

The current ISDA documentation structure is based on a single relationship agreement between two parties. However, the impact of various regulatory reforms[43] means the legal terms of a new single relationship agreement may now be split among multiple sources, making it difficult to readily view the precise terms of the legal relationship.

Re-consolidation of these agreements could offer benefits in the medium term, by making it easier to on-board new clients and identify and demonstrate the definitive record of existing contracts, eliminating potentially overlapping provisions. It could also be a catalyst for further standardisation of existing documentation terms.

One way to achieve this re-consolidation would be the development of a compiler that could electronically analyse the elections made by the parties and automatically produce a tailored version of the legal terms of the single relationship agreement.

This approach might even be extended to the individual transactions themselves. As the range of derivatives covered by the ISDA documentation grows, and as more derivatives are standardised to promote market stability and consistency, the ISDA definitional booklets inevitably become longer. This can make it difficult for a lawyer to easily see and review only the definitions pertinent to a trade.

An extension of the compiler described above would be to program the compiler so it could produce a compiled long-form transaction confirmation for any individual transaction that would extract from the ISDA definitional booklets those definitions relevant to that transaction and set them out in one place. This would greatly assist in the review of the terms of that individual trade.

A further extension of this approach might be to replace the separate ISDA definitional booklets for different types of derivatives with a modular ISDA library of individual definitions. Such individual definitions could then be combined by the compiler for an individual transaction.

When a transaction is entered into, it would be time stamped. This would mean that when the compiler prepares a compiled long-form transaction confirmation for that transaction, it would use the version of each modular definition that was current at that time stamp.

Such a system would allow the ISDA library to be updated on an ongoing basis rather than requiring periodic updates of definitional booklets or the publication of annexes containing often disparate updates to certain definitions. Currently, the latter creates its own problem in that, to understand the current set of definitions for an ISDA definitional booklet, one must read that booklet alongside the various annexes. The modular system described in this section would obviate the need for this.

A move to such a modular approach might seem a radical jump from the position today. But if elements of the ISDA definitional booklets are coded to facilitate smart legal contract implementation, this would lend itself to the idea that each definition could be viewed as a distinct module and could be kept in a universal repository.

[43] There are several potential instances of this. For example, the regulatory requirements with respect to margin for non-cleared derivatives have led to the development of a range of different credit support annexes and documentation. In any one derivatives relationship, the parties might enter several such annexes. In addition, when parties adopt ISDA protocols during the life of the relationship, this also alters the legal terms of the single relationship agreement, but without requiring an alteration to the hard copy contracts entered into between the parties. That can make it difficult for a reader to fully understand the terms of the relationship at a particular time

CONCLUSION

It is important for ISDA and the wider market develop standards that can be applied by technology providers

This paper sets out one potential, high-level vision of how smart contract technology might be used for derivatives products and ISDA documentation. However, while smart contract technology offers great potential, it is important to recognise that its development is still nascent.

Now is a crucial time in the development of smart legal contracts. If there are to be smart legal contract standards and technologies that can viably be used for derivatives products, it is important that ISDA and the global derivatives industry play a key role in the development of these standards and work closely with utilities and service providers in their development of the applicable technologies.

ISDA's Market Infrastructure and Technology Oversight Committee (MITOC) intends to facilitate this involvement, with smart contracts forming one part of the wider work anticipated by the September 2016 ISDA whitepaper. Working groups have been set up by ISDA to coordinate the various legal/documentation, regulatory, technological and reporting workstreams in order to future-proof standard documentation and align data standards. ISDA will hold further roundtable meetings going forward across the various workstreams to ensure cross-fertilisation and coordination.

A final word as to the likely time frame for change. At ISDA's 32nd Annual General Meeting in Lisbon in May 2017, the prevailing view was that DLT and smart contracts would have a meaningful impact on the derivatives industry within five years. That time frame is undoubtedly ambitious, but the need is pressing and the journey needs to start now.

ABOUT ISDA

Since 1985, ISDA has worked to make the global derivatives markets safer and more efficient. Today, ISDA has over 875 member institutions from 68 countries. These members comprise a broad range of derivatives market participants, including corporations, investment managers, government and supranational entities, insurance companies, energy and commodities firms, and international and regional banks. In addition to market participants, members also include key components of the derivatives market infrastructure, such as exchanges, clearing houses and repositories, as well as law firms, accounting firms and other service providers. Information about ISDA and its activities is available on the Association's web site: www.isda.org.

ABOUT LINKLATERS

As one of the leading global law firms, Linklaters supports clients in achieving their objectives by solving their most complex and important legal issues. Our expertise and resources help clients to pursue opportunities and manage risk around the world.