CLIFFORD CHANCE

ARE SMART CONTRACTS CONTRACTS?

– THOUGHT LEADERSHIP

DECEMBER 2017

ARE SMART CONTRACTS CONTRACTS?

Will smart contracts prove that the future of contract lies in machine executable software code-agreements, or are they rather an unhappy hybrid – as the quip goes: neither smart, nor contracts?

A new kind of contract

Smart contract technology – the implementation of electronic contracts, embodied in code and performed automatically by computers – is maturing fast. Despite ambitious projects run by start-ups, major financial institutions and others, no smart contract solutions have yet significantly displaced traditional ways of contracting, but that is bound to change. The technological and legal hurdles to implementing a fully-fledged, mature smart chain solution remain daunting. But in the longer term, it is difficult to imagine that the production, execution and performance of legal agreements will not be transformed by the trends of digitisation and automation that are reshaping so many other spheres of human activity.

Why should you care about smart contracts?

As a long-term trend, the development of smart contract technologies may herald the beginning of a general transformation of contracts from paper documents to self-executing code-agreements. But looking to the shorter term, instances of smart contract applications already exist and are being used by early adopters, even if these projects remain mostly experimental. Wide scale projects like Ethereum and the R3 consortium's Corda suggest that the technology is already emerging from infancy. One ambitious smart contract-enabled project that made the headlines in 2016 was the DAO (Decentralized Autonomous Organisation), launched as a fully-fledged crowd funding platform implemented on the Ethereum blockchain. The DAO largely bypassed customary legal and corporate structures to create what was essentially a virtual venture capital fund. The subsequent debacle, when the code was exploited to enable one participant to extract a large part of the funds, indicates that the ambitious project may have been somewhat premature, but it was nevertheless an impressive exercise in testing the potential of smart contracts and a case study for some of the legal issues facing smart contract implementations.

Commercially viable smart contract solutions may emerge around applications that are less radical in scope than the DAO, but rather confined to relatively narrow sets of operations that are well suited to the model of self-executing electronic contracts. Contracts that involve information processing and the execution of online transactions (payments, transfers of assets recorded in electronic ledgers) may be more susceptible to implementation as automated smart contracts than, say, employment contracts that by definition require performance by humans and so must remain impervious to complete automation. This accounts for the particular interest in smart contracts from financial institutions that deal with certain types of standardised contracts in well understood contexts that already involve extensive interactions with electronic systems. For financial transactions that already exist largely online and are already executed by software agents, fully self-executing contracts can be seen as a natural next step.

But are they contracts at all?

The detail of what successful implementations of smart contracts will ultimately look like remains an open question, but we already know enough about the likely issues smart contracts implementations face to be able to identify some key challenges that any solution will have to resolve. Developing a successful smart contract solution will require not only cracking technological problems, but also a uniquely legal question: is a smart contract a contract at all? We all have an understanding, in a general sense at least, of what a contract is, on the one hand, and of what computers do, on the other – but do we have a clear understanding of where the boundary between the two lies? Is there a grey zone between the two where something can be both contract and code? If not, can the gap between the two be bridged? The trick for smart contract developers will be to find a solution that works technologically, that makes sense legally, and that is simple enough that the market is not scared away and can trust in its integrity. We will return to the question of whether smart contracts can be made legally binding once we have explored some key smart contracts concepts – starting with a definition.

Automatability and enforceability

Academic work on how to implement contracts electronically stretches back decades. One of the reasons they are only emerging in the mainstream now is that previously too many aspects of economic life remained too disconnected – too *offline* – for the concept to work. General technological progress and the relentless computerisation of economic activities are pre-conditions to viable smart contracts, but at least one other development has also been key: the emergence of blockchain technologies (implemented most famously in the Bitcoin cryptocurrency), which has given the idea of electronic contracts a new lease of life by opening up new possibilities for genuinely autonomous execution. As a result of this long maturation, there is an extensive research literature on smart contracts and a number of good definitions to choose from. We have chosen as our starting point the definition from a paper by a team of experts from UCL and Barclays (*Smart Contract Templates: foundations, design landscape and research directions* by Christopher Clack, Vikram Bakshi and Lee Braine). This definition is particularly useful for the purpose of our discussion because it is flexible as to the extent to which a smart contract implementation may be fully implemented by computer code or instead combine aspects of traditional contracts and computer code. The definition is as follows:

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

A smart contract, therefore, is an agreement. A traditional contract is at its most basic level an agreement coupled with the intention of the parties to be legally bound by that agreement (omitting for simplicity some of the formalities various legal systems add to that). But a smart contract has two key additional features that distinguish it from a traditional contract. First, whereas a traditional contract is an inert text (or even a mere oral communication) and execution relies on the independent action of the parties, a smart contract is capable of at least partial performance by computers, without the parties' direct intervention. Second, the enforceability of a smart contract can be of two types: either of the traditional legal kind (i.e. a court could enforce it) or of a novel kind – tamper-proof execution of computer code.

Tamper-proof execution

What does the "tamper-proof execution" in this definition refer to? The use of the expression "tamper-proof" reflects the definition's indebtedness to the aforementioned blockchain technologies, which are fundamentally a way of creating tamper-proof electronic record of transactions – i.e. a record which neither the parties nor any third party can modify. What the definition is getting at here is that the enforceability of a smart contract may not necessarily lie in the fact that the output (i.e. its performance) can be enforced by a court, but, as an alternative sense of "enforceable", that it may be effected by an autonomous technological process which, once initiated, cannot be interfered with. If I put a coin into a vending machine slot and enter a valid selection, there should be no requirement for the vendor to step in to ensure that the transaction is performed (the delivery of a can of soda) – it should be *autonomous*; and no further ability of the vendor to interfere with its consummation – it should be *tamper-proof*. If I submit a valid Bitcoin transfer instruction, the transaction is recorded autonomously on the Bitcoin blockchain and the blockchain is not further modified until a subsequent valid transaction is entered into. Similarly, a smart contract solution, to have the property of tamper-proof execution, would have to be capable of ensuring its performance in such a way that it depends only on the completion of an autonomous technological process.

Legal enforceability trumps mere performance

But are these two types of enforceability, legal and technological, equivalent? Or do they rather just re-frame the dual nature of smart contracts as creatures of both contract and code? Looking back to the first question raised about the legal status of a smart contract, it is only if a smart contract constitutes what a court would recognise as a legal contract that the court will enforce it. As for the technological sense of enforceability on its own, if a court would not be willing to recognise the outcome of a technological execution process (i.e. mere tamper-proof execution) as legally effective, then that outcome will be superseded by any contrary legal rule. Consider the following scenario: Alice owns a piece of land, which she wants to transfer to Bob. They happen to be early adopters of a blockchain solution that involves an electronic ledger, on which ownership of real estate interests is represented by electronic tokens. Using that blockchain solution, Alice executes a smart contract that transfers the token representing her piece of land to Bob. On the blockchain, everything goes according to plan: the token transfers from Alice to Bob, and Bob is represented on the blockchain as the new owner of that piece of land. However, it turns out that the law requires a transfer of that kind of estate in land to be effected by deed, signed and witnessed. Unfortunately, the designers of the blockchain/smart contract system did not think of this. The outcome? Absent regulations specifically recognising such electronic transfers as valid, no court will recognise Bob's title to the asset. The court may be unable to modify the result of the transfer on the blockchain (if it is genuinely tamper-proof) but clearly if a dispute were to arise concerning the ownership of the land, the court's view would be determined by its interpretation of the legal rules applicable to the facts and not the face of the blockchain asset ledger. The legal analysis will always trump the technological – because, even for computer-executed contracts, the final arbiter of legal effect will be the courts.

Code and contract

This issue of whether a smart contract is an enforceable contract at all stems from a more fundamental issue: computer code and legal drafting are two fundamentally different things. They have a number of similarities that may tempt us to conclude that they are not that different from each other: they use specialised languages with rules as to how that language should be created and interpreted; they have formal structures with a multiplicity of functional components that interact with each other in accordance with a well-defined logic; and they even share the notion of "execution." But scratch the surface and it is readily apparent that these are two completely separate categories of thing: on the one hand, an agreement between human agents, embodied in human language, which humans perform based on their human language interpretation of those words; and on the other hand, code that is ultimately compiled and executed by computer processors as strings of binary machine code at a high level of abstraction and yielding real-world outputs only to the extent that those computers can produce real-world outputs. You can indeed "execute" software code, but the execution of machine code on a computing platform bears no general resemblance to what it means to execute or perform a contract. If smart contracts *as code* are to be executable in the same sense as legal contracts, how can the gap between those two domains be bridged?

CONTRACTUAL OBLIGATIONS	COMPUTER CODE
Embodied in human language text	Embodied in machine-readable data
Drafted using specialised human language	Coded in machine-executable software language(s)
Takes effect on the basis of parties agreeing	Takes effect by being executed electronically
Performance carried out by the parties	Performance by execution by computers/software agents
Performance in accordance with interpretation of human language	Performance (code execution) follows deterministic machine execution process
Parties can agree to disregard errors	Code is performed "literally"
Provisions can remain ambiguous	Ambiguous code cannot be executed or executes incorrectly

Contract and code contrasted

AI: not that smart yet

A tempting shortcut to smart contracts might involve focusing on artificial intelligence (AI), which is rapidly maturing, and relying on the fact that soon computers will be able to process human language on a par with humans, so they will be able to interpret the language of a contract and perform the contract essentially as robotic agents of the parties. But this, for now, is a red herring. AI that has that level of general functional equivalence to human intelligence is a long way off. In any event, none of the current smart contract technology contenders function on that level: they invariably seek to convert certain performable obligations in the contract into something which has a degree of conceptual rigour and formal clarity that is similar to existing forms of computer code and would not allow the level of ambiguity that characterises human language. To put it another way, smart contracts are smart as in "smart watch" (i.e. connected), not as in "intelligent."

Forget about the gap?

One perspective worth considering regarding the gap between code and contract, if only to challenge our preconceptions, is how much it really matters that smart contracts should have the same enforceability as traditional contracts. Enthusiasts of blockchain technologies may argue that the fact that a smart contract would not be recognised as a traditional legal contract is ultimately irrelevant, if it performs much the same function as a traditional contract. Accordingly to that line of argument, our continuing attachment to law courts and paper contracts may just be failure of the imagination. If the law claims supremacy over code, why not leave laws and lawyers behind and freely transact using smart contracts in a stateless cyberspace where legal effectiveness no longer matters? While there may be trade-offs in terms of flexibility, of interpretation or of enforcement, if smart contracts were faster, cheaper and more reliable than traditional contracts might they not gain market share and displace cumbersome paper contracts, perhaps starting with low-value transactions where judicial enforceability is not paramount? Just as some crypto-currency enthusiasts see Bitcoin as displacing traditional state-backed fiat currencies, there is no shortage of techno-utopians who see smart contracts as a way of dispensing with the traditional infrastructure of the law. The fact that as of now traditional currencies show no sign of extinction, while Bitcoin remains something of an experiment, and that few lawyers are losing their jobs to smart contract developers, does not mean that these emerging technologies will not ultimately shape the future of their sectors.

Back to the present

Yet in the shorter term at least, few established businesses will be willing to adopt smart contract solutions that jettison the tried and tested certainties of the legal system. For now, the real commercial prospects lie in bridging the gap between code and contract, on finding ways to preserve enforceability in the traditional legal sense of "enforceable by a court" (the first meaning of enforceability) while reaping the benefits of automated tamper-proof execution (the second meaning of enforceability). That is where the efforts of many smart contract initiatives are focused and where credible solutions, appropriate for complex regulated environments like the financial sector, are beginning to emerge.

Wrap it up

So how do you bridge the gap between machine executable code and legally enforceable contracts? A simple solution is: you wrap the code up in a contract. Perhaps a day will come when the law will evolve to recognise certain types of code as capable of enforceability in the legal sense without more, but at present the only way we see to be sure that the output or performance of smart contract code has the force of legal obligation is to confer that legal binding effect by means of a more traditional legal contract that serves as a legal "wrapper" for the self-executing code. Of course, this idea of a wrapper is a high-level concept, and the devil will be in the detail of how the wrapper is implemented.

Execution: the law and the vending machine

The law is ultimately adaptable and has followed technological changes, albeit not infrequently with a time lag. Under English law, the doctrines necessary to give legal force to at least a broad class of simpler smart contracts are already in place and in some cases these laws may not in fact be particularly new: existing electronic execution rules may be broadly sufficient to ensuring that code-and-contract hybrids can be validly executed as contracts. The fact that smart contracts may exhibit entirely new functionality does not mean that the law would need to develop complex new rules to cater for their execution. To return to our earlier example of the humble vending machine: when I insert a coin into a vending machine, I create a contract, but not because of some sophisticated technical functionality of the machine. A "wrapper" contract is created because the law interprets my inserting the coin as a valid acceptance of the offer made by the owner of the vending machine to purchase the product I have selected. The vending machine may deliver the can of soda autonomously as a real-world output, but the legal contract is created the old-fashioned way (with a little flexible interpretation): by offer and acceptance. With smart contracts, however, reliably creating that wrapper and ensuring that valid offer and acceptance has taken place will likely require an explicit process that incorporates the legal requirements for electronic execution of traditional, non-automated contracts – something like clicking an "I Agree" button before launching – rather than relying on speculative smart contract-friendly interpretations of common law rules. The relative flexibility of English law with respect to electronic execution may make it better suited to smart contract implementations than legal systems with stricter formal requirements, but even so there will likely be limitations to what can be achieved, such as documents that need to be executed as a deed (as Alice and Bob found out in our earlier example).

Maintaining lockstep between legal execution and code execution

This resilience of traditional contract law doctrines in the face of technological change can hopefully be transposed from the simple scenarios such as vending machines and clickable terms and conditions to the more unpredictable environment of multiple autonomous software agents entering into complex transactions. If this is to be done without the help of law reform, it will require careful thought from smart contract developers to ensure that their solutions fall as much as possible within the categories that contract law already understands. In terms of smart contract creation processes, how do you make sure the formalities for the execution of a legal contract move in lockstep with the deployment of the self-executing functionality of the code? To return to our real estate blockchain ledger example – how do you ensure Alice can validly transfer her land to Bob, both on the blockchain and in the "real", legal world? How do you even meet those formalities once software agents start entering autonomously into ancillary transactions with other software agents – without a direct meeting of human minds? If the self-executing process yields an output which is fundamentally different to what the parties anticipated, what happens? If the execution was genuinely tamper-proof, how can it be reversed?

These are questions on which the viability of smart contract solutions will turn.

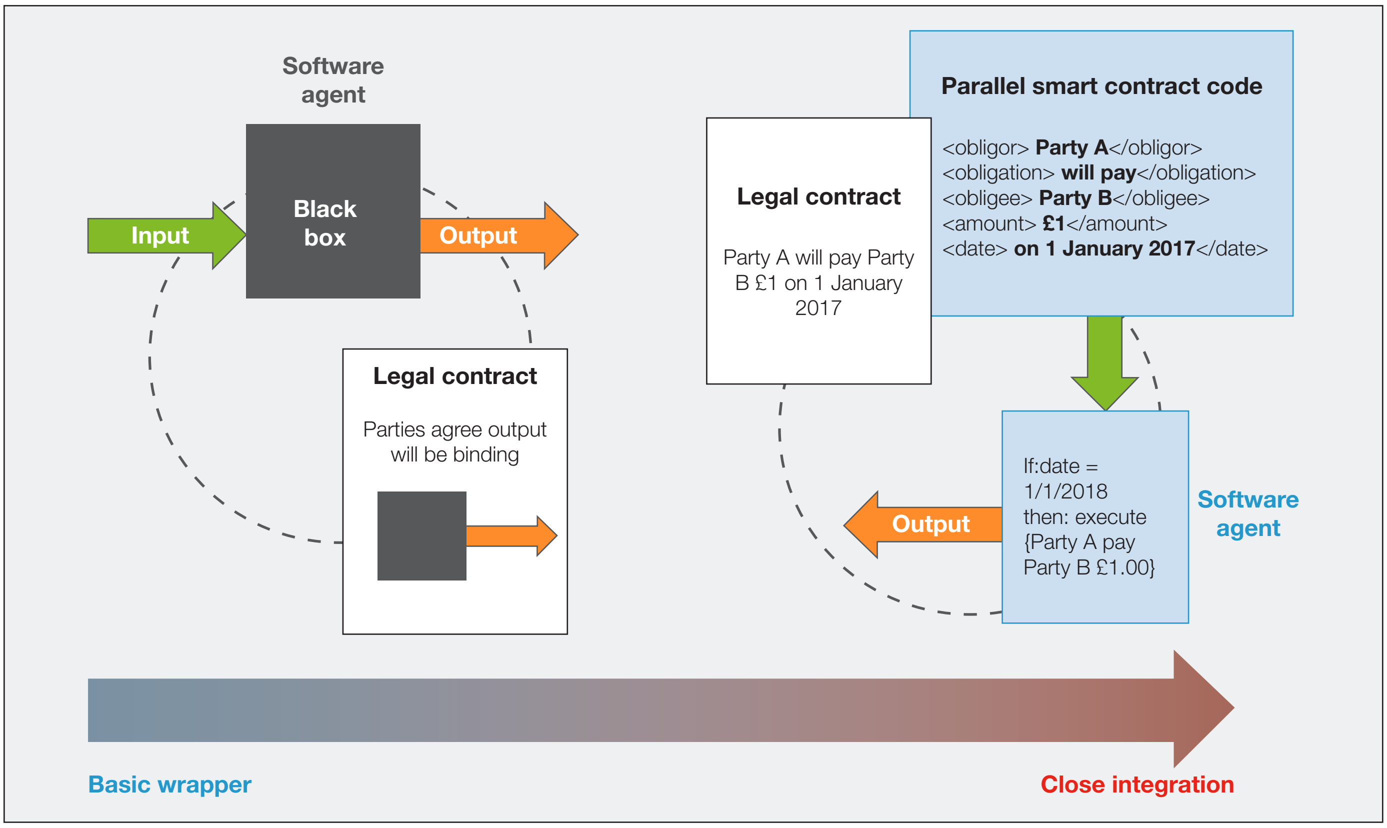
A wrapper around a black box

Even before those questions can be answered, a more general issue regarding the implementation of wrappers needs to be addressed: the question of the relationship the human language contract wrapper bears to the self-executing code itself. At the very simplest level, a wrapper contract is one that says "We, the parties, agree that (i) this is a contract and (ii) the output of this self-executing process is its performance." For this minimal wrapper contract, the code is effectively a black box – the contract says nothing about the expected behaviour of the black box, so whatever the output of the black box, that is what they legally agree is the intended result, whether or not it meets the actual (but undocumented) expectations of the parties. Leaving aside the possibility that such a pure black box approach could be held to be void for uncertainty, it would take a lot of faith in the technology for the parties to make such "blind" undertakings, liable to commit them to the erroneous output of a bug in the black box.

It is also possible to set out in the legal wrapper contract terms relating to the expected output of the code, so that if the code does not perform as expected the parties are not left without remedies. While prudent, this approach may erode the advantages of speed and certainty to be gained from implementing performance of the contract through self-executing code. It also raises questions as to how tamper-proof the execution of the code can be if it needs to be left open to be corrected by the parties whenever there is an issue.

From black box to parallel languages

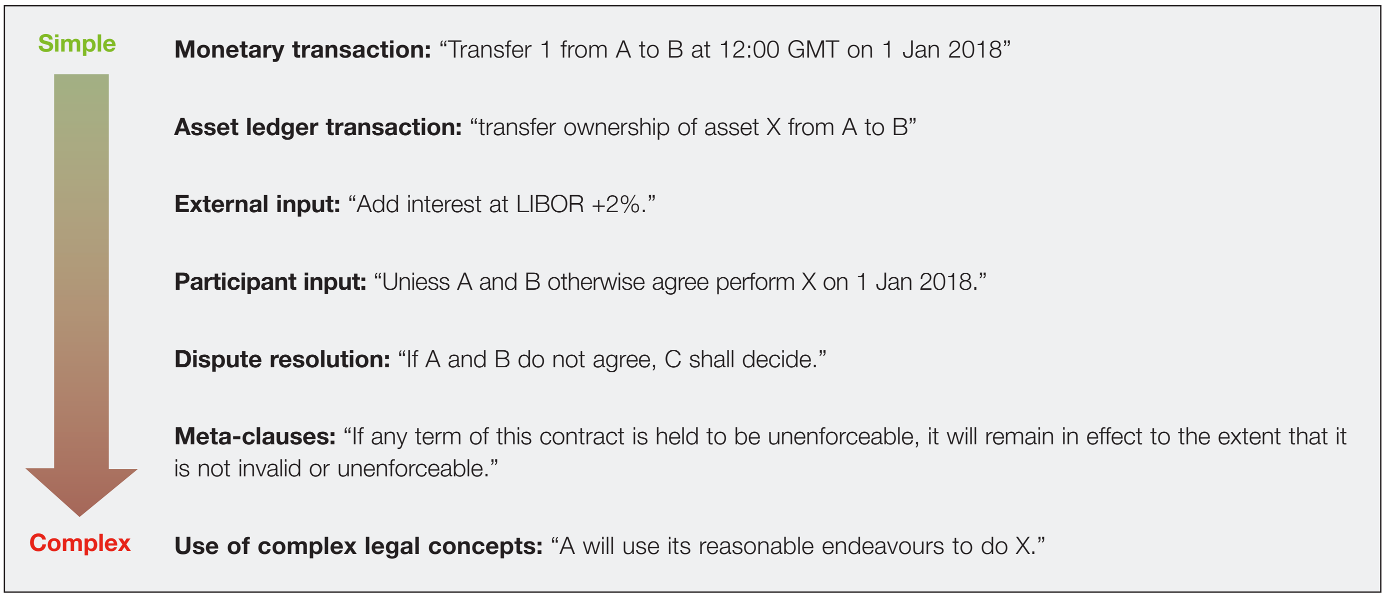
If few parties will be comfortable with a pure "black box" approach that shifts all performance risk onto the technology, yet they want to avoid the code being a purely ancillary instrument of performance (no different, for example, than agreeing in an ordinary contract to make an electronic bank transfer), preserving the benefits of self-execution requires solutions that integrate the code and the legal provisions so that the execution of the smart contract code tallies with the human language reading of the text of the contract. Effectively this requires creating a formal language that works both on the human language level and on the machine level. For reasons we have already touched upon, that machine language cannot for the foreseeable future be equivalent to the whole of ordinary human language, with all of its capacity for ambiguity and contradiction. However, there are relatively simple contexts where the idea of a dual-function, formalised language is really not too difficult to imagine. Take the following instruction: "*If today's date is 1 January 2018, pay £1.00 out of Party A's account to Party B's account.*" Its English meaning is plain but, with a little formalisation perhaps, it does not look like a proposition that would be difficult to parse and execute for a software agent that also has the ability to initiate payments. The more fixed the parameters and the more limited the variations between individual contracts, the easier it should be to create modular "blocks" of contract-code, where standardised paragraphs in this formalised language, which read both as English text and as machine-performable code, implement the relevant function. These blocks could then be assembled like Lego bricks to build a complete smart contract. A promising direction for the development of smart contracts may therefore lie in focusing on limiting the options of the parties and prioritising the implementation of templates for highly standardised documents (for a more detailed discussion of how a parallel machine language might be implemented, with a focus on creating smart contract templates, see a second paper by the same authors we quoted above: Smart Contract Templates: essential requirements and design options).


Software agent
Black box
Input
Output
Legal contract
Parties agree output will be binding
Legal contract
Party A will pay Party B £1 on 1 January 2017
Parallel smart contract code
<obligor> Party A</obligor>
<obligation> will pay</obligation>
<obligee> Party B</obligee>
<amount> £1</amount>
<date> on 1 January 2017</date>
If:date = 1/1/2018 then: execute {Party A pay Party B £1.00}
Output
Software agent
Basic wrapper
Close integration

Wrapping the smart contract code in a legal contract wrapper

Automatability: a spectrum

Not all functionality will be as straightforward as the simple proposition we have just considered. Ultimately, smart contracts will live or die by their ability to perform more efficiently, more quickly and more cheaply things that are currently mediated by human agents. The question of *how much* smart contracts can reasonably be expected to perform autonomously (and how much must be left to human agents) is therefore a question that goes to the root of the potential of smart contract technologies. Let us consider a spectrum (see graph below), ranging from operations that look reasonably straightforward to implement (and in some cases have already been implemented) to operations that look challenging or impossible, given the current state of the technology.



Monetary transaction: "Transfer 1 from A to B at 12:00 GMT on 1 Jan 2018"

Asset ledger transaction: "transfer ownership of asset X from A to B"

External input: "Add interest at LIBOR +2%."

Participant input: "Uniess A and B otherwise agree perform X on 1 Jan 2018."

Dispute resolution: "If A and B do not agree, C shall decide."

Meta-clauses: "If any term of this contract is held to be unenforceable, it will remain in effect to the extent that it is not invalid or unenforceable."

Use of complex legal concepts: "A will use its reasonable endeavours to do X."

Automatability: a spectrum from simple to complex

- **Monetary transaction** – as we have just noted, in an age of routine online payments and cryptocurrencies, it is not difficult to conceive of a process whereby parties can enter into an agreement to make one or more payments that are effected by software agents. For that process to be genuinely capable of "tamper-proof" execution, it may need to be implemented on some type of blockchain or similarly autonomous structure, but the technical and conceptual challenges here are comparatively slight.
- **Asset ledger transaction** – the rise of blockchain technologies has popularised the notion of online ledgers of assets where entries on the ledger are "tokens" for real-world assets – as has been trialled for land registries by public bodies in Honduras, Sweden and Georgia. It is not a big conceptual stretch to get from cryptocurrency-type transactions to tokenised asset transactions, although the key challenge here will be the link between the potentially tamper-proof ledger and the real-world assets which it represents. As we considered in the example of Alice and Bob's blockchain-enabled land transaction, is it possible for a transaction to be valid on the ledger but invalid at law? If so, the implementation may run into trouble.
- **External input** – if smart contracts are meant to self-execute, what happens when you need to bring in inputs from outside that self-executing system? For example, what happens if interest needs to be charged for the purposes of the implementation of some aspect of the smart contract – where do you get the current interest rate from? Because interest rates are set from time to time by central banks, they can't be pre-programmed into the smart contract code. The code will have to reach out an external source and take that interest rate as an input. The sources of such external inputs are referred to in the jargon of blockchain technologies as "oracles", and there are questions as to how to implement these in a way that is resilient and that the parties can trust. As a purely technological question, however, the notion of integrating the ability to receive input from third party sources into smart contract implementations does not appear overwhelmingly difficult – indeed, solutions implementing this are already available.
- **Dispute resolution** – dispute resolution is a concept that seems to fall so squarely within the traditional domain of the legal professions and courts that we might be inclined to conclude that it must be beyond the reach of purely technological solutions. However, if a smart contract can be coded to accept third party inputs, it may not be too much of a stretch to implement functionality whereby the parties can call upon a dedicated function in the smart contract code which requests the input of a specified third party (chosen, perhaps, from a pool of specialised smart contract dispute resolution experts). That third party can then intervene to suspend, terminate or modify the terms and execution of the smart contract and so resolve the dispute. This would require creating exceptions to the model of purely tamper-proof execution to allow for the intervention of the smart contract "arbitrator", but that may be a worthwhile trade-off to give reassurances to the parties that if there are issues with the performance of the smart contract, they will not be left without a remedy.
- **Meta-clauses** – traditional contracts often include "meta-clauses", i.e. clauses that concern the contract itself or other clauses of the contract: principles of interpretation, variations clauses etc, frequently relegated to the "boilerplate" sections at the back of the contract. Here, surely, we are passing beyond the boundary within which current technology can provide a solution? It certainly seems unlikely that high-level principles of human language interpretation can be implemented in code, for the reasons we have already mentioned. Yet functionality similar to that of some common meta-clauses may nevertheless be replicable in code. Take, for example, severability clauses, which provide that if a clause of the contract is found to be unenforceable, it should be interpreted so as to preserve the original intent to the extent possible. Modern coding allows for

complex error-catching, which detects instances where the code encounters an execution error and switches to a "Plan B" to try to preserve the intended functionality of the code notwithstanding the failure. There is no reason in principle why similar coding could not be built into smart contracts. This is not exactly the same concept as a traditional contract severability clause, but it would play largely the same role in trying to preserve the intention of the parties by improving the resilience of the code in the face of errors that result in defects in its implementation.

- **Use of complex legal concepts** – the meaning of certain commonly used legal concepts is subject to a wide range of interpretations that can only be settled on a case by case basis. Take, for example, a commitment by a party to use its "reasonable endeavours" to achieve a certain outcome. That wording is specifically chosen, as a matter of drafting, as an alternative to a strict obligation, to indicate that there are circumstances in which that party will not be expected to perform because to do so would go beyond a certain threshold of "reasonableness" given the particular context of the obligation. It is open to the parties to specify exactly what will and will not be reasonable, but the choice of the term "reasonable endeavours" is a way of postponing that determination, so that the standard of what is or is not reasonable can be resolved based on a concrete set of facts when an issue arises. Because this type of concept cannot be set out in a formalised language that can provide a well-defined set of instructions for code to execute, we do not see that any prospect of implementing these until the advent of much more powerful AI than is available today, that can make that context-specific determination of reasonableness.

The expanding domain of smart contracts

From this overview of some key issues raised by smart contracts, we have seen that the current state of the technology suggests that effective smart contract solutions will emerge in certain well-defined contexts that lend themselves to at least partial automation of the performance of the contract. We expect to see a continued profusion of innovation from what we might term the "pure technologists", of the kind who embraced Bitcoin in its early days, or more recently projects like Ethereum and the DAO. Yet smart contract development will not be confined only to experiments in the Wild West of technology. On the contrary, as we have seen, major projects are already well under way from technologically progressive established businesses in prudent and risk-aware sectors such as financial services. The main reason for this is that the environment in which banks and other financial institutions operate is already so deeply computerised that smart contracts, for the right applications, are not ultimately a leap of faith but rather just the next step in a relentless process of automation.

Clearly, not all domains will be equally suited to smart contracts, but even for contracts that cannot be performed entirely by machines, as contracting parties grow accustomed to the efficiencies of automated performance, it may be that they will come to prefer dealing with contracts that are at least partially smart, so that routine payment mechanics or other easily automatable tasks can be taken over by machines, leaving the human agents to focus on the more challenging tasks. The key challenge for smart contract technologies will continue to be finding ways of resolving the dual nature of smart contracts as creatures of both human language and of machine code. As more and more areas of economic activity undergo processes of digitisation and automation – for example through the spread of blockchain and other distributed ledger technologies – they will become easier to automate, expanding the domain of self-executing smart contracts and transforming the business of the law.

CONTACTS

Brian Harley
Registered Foreign Lawyer
Hong Kong
T: +852 2826 2412
E: brian.harley@
cliffordchance.com

Kate Gibbons
Partner
London
T: +44 20 7006 2544
E: kate.gibbons@
cliffordchance.com

Paul Landless
Partner
Singapore
T: +65 6410 2235
E: paul.landless@
cliffordchance.com

Peter Chapman
Senior Associate
London
T: +44 20 7006 1896
E: peter.chapman@
cliffordchance.com

Laura Nixon
Senior Associate
London
T: +44 20 7006 8385
E: laura.nixon@
cliffordchance.com

Christopher Duff
Senior Associate
London
T: +44 20 7006 8793
E: christopher.duff@
cliffordchance.com

Nikhita Suria
Lawyer
London
T: +44 20 7006 6214
E: nikhita.suria@
cliffordchance.com

Jeremy Walter
Partner
London
T: +44 20 7006 8892
E: jeremy.walter@
cliffordchance.com

CLIFFORD

CHANCE

J20173107171314