Smart Contract Templates: legal semantics and code validation

Christopher D. Clack
The Centre for Blockchain Technologies
Department of Computer Science
University College London

Abstract

Smart Contract Templates support legally-enforceable smart contracts, using operational parameters to connect legal agreements to standardised code. The standardised code is derived from legal documentation and performs some or all of the provisions of that contract. For financial contracts such as derivatives agreements the legal documentation may be extensive and the standardised code to perform the contract may be substantial. An important issue is how to validate whether the smart contract code will correctly perform the provisions of the legal contract. This requires an understanding of the semantics of legal text, and is an important step towards supporting industry adoption of legally-enforceable high-value smart contracts.

Keywords: Smart Contract, Smart Contract Templates, Semantics, Automation, Distributed Ledger

1 Introduction

Smart Contract Templates[1,2,3,4] aim to support the management of the complete lifecycle of "smart" legal contracts, including legal document templates created by standards bodies and the use of those templates by counterparties during the negotiation and agreement of contracts. An important aspect of Smart Contract Templates is that they also facilitate automated performance of the contract.

Smart legal contracts, based on Smart Contract Templates, could potentially be implemented as software agents running smart contract code[5] that may operate on a wide range of technology platforms, including distributed ledger platforms.[6,7,8,9,10]

This paper focuses on the requirement to validate the smart contract code generated by Smart Contract Templates, i.e. to determine whether the code will faithfully perform the contract. This turns out to be a complex problem that involves understanding the semantics of legal text (both the meaning of legal words – "lexical semantics" – and the logical meaning of the contract clauses, including tacit assumptions and implications), circumventing semantic misunderstandings between the two disciplines of law and computer science, and elucidating hidden meaning (such as knowledge relating to the legal contract that would not be apparent to a non-lawyer, even if a full lexical semantics were available). The discussions within this paper are of broad relevance to academics and practitioners, and reasonably straightforward language is used so that these discussions are accessible not only to financial institutions but also to, for example, lawyers, regulators, standards bodies, and policy makers.

Acknowledgements: This paper follows two previous papers on Smart Contract Templates coauthored with Lee Braine (Barclays) and Vikram A. Bakshi (Barclays).[11,12] Initial concepts for this paper were first presented at The Third R3 Smart Contract Templates Summit in June 2017.[13] All extracts from the ISDA 2002 Master Agreement are reproduced with the permission of International Swaps and Derivatives Association, Inc. The author is grateful to the anonymous reviewers for their helpful comments, and to Kate Gibbons (Clifford Chance LLP) for feedback and insight on an early draft.

2 Background

Interest in smart contracts exists for both low-value and high-value business agreements, and for both bespoke and standardised agreements, yet the term "smart contract" is often used in contradictory ways. For some, a "smart contract" does not refer to a contract in a legal sense but instead refers to computer code that automates business processes without the need for recourse to the courts of law to resolve disputes. For others, a "smart contract" is the automation of (parts of) a legal contract, and the enforceability of that contract in the courts of law is extremely important.

In the former case, the computer code might be taken to be the full expression of the parties' intentions, and enforcement of these intentions might (or might not) be achieved by technological force such that once the code has started to run it cannot be stopped or modified. In the latter case, the computer code (which may or may not form part of the legal contract) may need to pause for human input, or may need to be paused or modified as the result of human input – these matters are discussed in more detail below. Stark[14] calls these respectively *smart contract code* and *smart legal contract*, and observes that a *smart legal contract* would normally be performed by the running of *smart contract code.*

Smart contracts might be used in both less-regulated and more-regulated business areas, with a range of automation from only trivial aspects to full automation of the entire contract. There is a broad spectrum of applicability for smart contracts and a similarly broad spectrum of approaches to the building, managing, and performance of smart contracts.

When this broad spectrum is coupled with the problems of conflicting terminology (discussed later in this paper) that arise from the overlapping disciplines of law, banking and computer science, there is great potential for confusion. To harmonize the different uses of the term, Clack et al,[15] provide a portmanteau definition of a "smart contract" 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.*

This definition is sufficiently broad to encompass most work in the area of smart contracts. The word "smart" is interpreted in relation to automation, and the fact that the computer code has a degree of autonomy – it can make some (but perhaps not all) decisions, whether simple or complex, without recourse to human control. The word "contract" is interpreted broadly to mean either a legal contract enforceable in the courts of law or an agreed schedule of actions and decisions that, although not enforceable in law, cannot be tampered with once started.

Tamper-proof code might not be suitable for high-value, long-duration, highly-regulated financial contracts. Discretion is a key issue with many such contracts, and there must be support for the smart contract code to request human input. Furthermore, a change in applicable law might require smart contract code to be stopped and perhaps modified before continuing – for example, if a change in legislation has made some encoded actions illegal, or if further actions are required to align the code with new regulatory requirements. Gibbons[16] further observes that changes in market circumstances might require the parties to agree what to do next, which would break the previously agreed chain of actions and decisions.

This paper focuses on smart contracts for high-value, highly-regulated financial agreements, with all of the foregoing complexities, and therefore focuses on *smart legal contracts*. Furthermore, the term *smart contract code* is used to refer to the computer code that is used to perform some or all of the actions of a *smart legal contract*.

A number of recent publications discuss smart contracts and distributed ledger technology from a legal perspective, for example white papers authored by ISDA and Linklaters[17], Clifford Chance[18] and R3 and Norton Rose Fulbright[19].

2.1 Smart Contract Templates

The initial case study for Smart Contract Templates was financial derivatives contracts (specifically, Interest Rate Swaps); such contracts commonly use the ISDA Master Agreement and associated standardised documents. A full set of ISDA legal documents might comprise a Master Agreement, a Schedule (with modifications negotiated between the contract parties), definition booklets, and a Credit Support Annex. This document set as a whole, when legally executed, creates an agreement between the parties to set the terms for one or more future trades; each such trade being evidenced by a written or electronic Confirmation document within which are stated the precise financial parameters for that trade.

Smart Contract Templates facilitate alignment of the process of using smart contract code with the process of using the ISDA document set:

- The ISDA Master Agreement and associated documents are viewed as standardised legal contract "templates", with some terms undefined. In a comparable way, Smart Contract Templates provide standardised smart contract code "templates" for the ISDA document set, with some terms as yet undefined (there may be different versions of the code, designed to run on different technology platforms). This standardised smart contract code is developed and comprehensively tested in advance.
- Many undefined terms are subsequently defined within the negotiated Schedule, and the Schedule may also modify some of the terms of the Master Agreement. Once the Schedule has been negotiated and the set of documents has been legally executed, a copy is made of the standardised code template; this copy is extended with definitions for some previously undefined terms, and modified as indicated in the Schedule. Following such additions and modifications, this copy of the smart contract code template is likely to require further testing. The resulting modified smart contract code is not yet ready to run, since the parameters for individual trades are not yet known.
- For each new trade under this agreement, evidenced by a Confirmation document, a new copy of the modified smart contract code is created, with definitions for the remaining undefined terms being provided by the trade parameters held in the Confirmation document. Some additional parameters might also be passed to the code, for example a cryptographic hash that can be used to identify the legal documentation if any dispute arises during automatic performance of the contract. The final version of the code may then be instantiated to run on a distributed ledger platform and perform the appropriate actions defined by the contract.

These processes are illustrated in Figure 1.

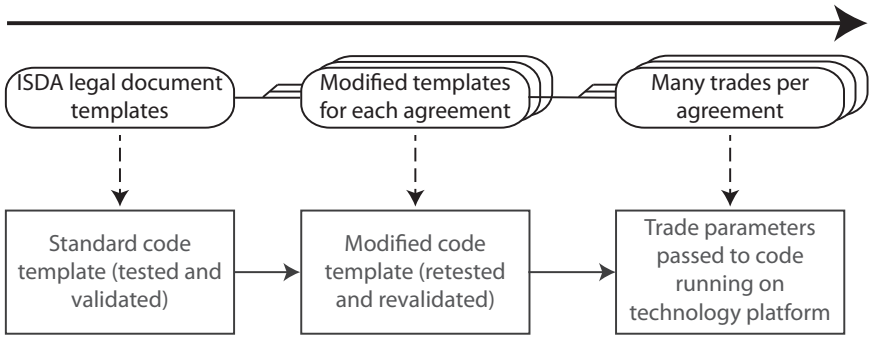

Figure 1: Smart Contract Templates facilitate the alignment of the process of using smart contract code with the legal process of using the ISDA document set. A financial institution may derive many agreements from the ISDA document set, and many trade confirmations may derive from each agreement.

2.2 Languages and representations for smart contracts

As with the definition of a "smart contract", it is important to be clear about other concepts. For example, when "a new language for smart contracts" is mentioned, what does that mean? Is it a new contract-authoring language, or is it a new programming language for writing the smart contract code?

One way to explore the various representations and languages that might be used is to consider the three core activities of "Contract Authoring", "Storage and Transmission", and "Code Generation". A fourth activity can also be added: "Analysis", which assists both contract authoring and code generation.

1. **Contract Authoring**. The traditional way to write a legal contract is to use a human language (such as English) and to write it down on paper using pen and ink. This comprises three aspects: the language (English, specifically "legal English", based on a legal ontology – i.e. a set of legal concepts and categories, giving their definitions and properties and the relations between them), the mechanism (pen, ink, paper), and the human readable form (the final written copy). It is now common to replace the mechanism with a computer running a word-processing program – the "human readable form" would now be either a printout on paper or an image rendered on a computer screen, and a new aspect is added (the "internal representation" of the document held within the computer, which might be for example a variant of XML). The word-processing program might be replaced by a more advanced program that combines word-processing with document-construction capabilities such that a lawyer may choose from a selection of prepared clauses to insert into the contract, yet the same four aspects remain – the language (still "legal English"), the mechanism (a computer program), the human readable form (paper printout or image on a computer screen), and the internal representation (perhaps XML).

   There is much interest in the development of new contract-authoring languages in which to write new legal contracts, or to rewrite existing contracts, using a style familiar to computer programmers (e.g. Legalese[20]). Such a language might combine "legal English" with logical or arithmetic expressions, and might have strict rules on the construction of sentences and paragraphs; alternatively, the language might closely resemble a computer programming language with little if any traditional text. One idea that stems from this approach is that the smart contract code could be more easily (and perhaps automatically) derived directly

from the legal contract, with a better mapping from the meaning of the contract to the meaning of the code and therefore with easier validation of the code. The internal representation might be directly and easily understandable by a computer, and the translation of the smart contract code for a given technology platform might be achieved automatically in a way that provably preserves the semantics of the contract.

The same four aspects remain – the language (new), the mechanism (a computer program), the human readable form (as above), and the internal representation (which might be bespoke) – but now the link from the internal representation to the generation of computer code may be more streamlined. However, at the high value, more regulated, end of the business spectrum it is unlikely that contracts will be entirely constructed anew for each new agreement, and unlikely that the smart contract code will be required to perform all of the semantics of the contract (see below).

2. **Storage and Transmission**. When a high-value derivatives agreement is negotiated, the parties will pass suggested drafts of the ISDA Schedule (for example) between themselves. This requires a standardised, structured representation of the contract that is suitable for storage and transmission between the parties. Traditionally this might have been print on paper, but for electronic transmission a different solution is required. One solution would be to transmit the internal representation of the computer program used for contract authoring, yet there may be many such programs with different internal representations.

   The prior work on Smart Contract Templates has included a set of essential requirements and design options for storage and transmission of smart legal agreements,[21] and this has contributed to a broader community where new initiatives are underway to rethink the structuring and management of legal documents. For example, Common Accord[22,23] creates "codified prose" from legal documents; one important aspect of such "codified prose" being that a legal document is captured in a formal, standard, structure. This aligns with the spirit of Smart Contract Templates, and facilitates the construction and negotiation of legal contracts. Such systems might comprise a contract authoring language, a mechanism (computer program), a human-readable form, an internal representation, and a representation for storage and transmission. As mentioned above, the last two of these might be the same – though the need for a common standard might result in them being different, and the translation into smart contract code might be derived from either.

3. **Code Generation**. There are many different potential technology platforms for smart contracts, and a potential next step might be the development of a "common" programming language in which to write smart contract code. It is likely that such high-level code will initially be generated and tested manually by computer programmers. From this high-level code it would be possible to generate (perhaps automatically) multiple low-level versions of the smart contract code, each in a different low-level language as appropriate for a specific technology platform. Each version of low-level smart contract code might require further testing.

   When a trade is confirmed, the trade parameters must be passed to a running instance of the smart contract code. This data must be conveyed in a standard, structured representation (for example, as FpML).

4. **Analysis**. Many benefits will derive from formal analysis of the legal documentation. For example, during contract authoring and negotiation a formal

analysis of the documentation might highlight missing cases and inconsistencies. Furthermore, as discussed in more depth below, a formal semantic analysis (for example, of the rights and duties of the parties, and of the various time-related aspects of the contract) could be used to generate test cases for use during the generation of smart contract code. The result of such analysis would be a separate semantic representation of the contract, which might be expressed using a formal semantic language.

Figure 2 illustrates summarises the various languages and representations that relate to smart contracts.

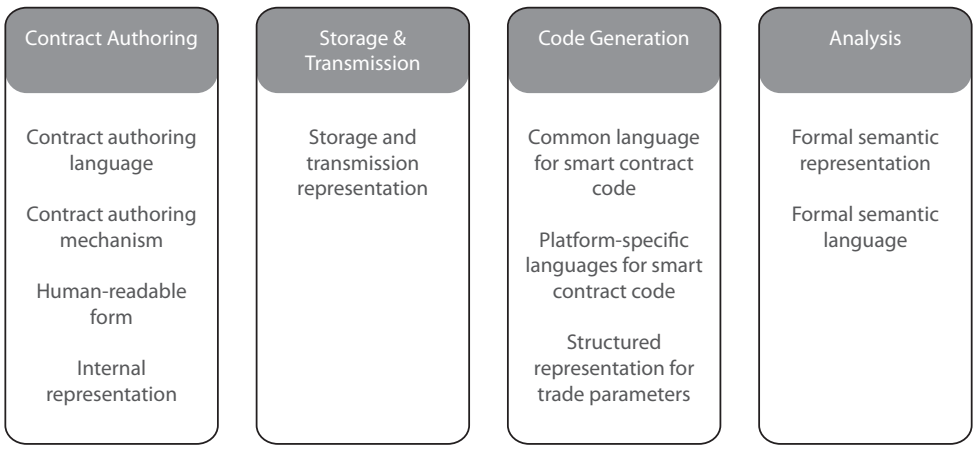

Figure 2: Different languages and representations used for smart contracts during contract authoring, storage and transmission, code generation and analysis

Although it might be possible for a single language or representation to be used for more than one purpose (e.g. to be used as both a formal representation of the semantics of the legal documentation and a language in which to code the smart contract code, or to be used as both a contract authoring tool and a formal representation of the semantics), this is not necessary and it is arguably premature to conflate these issues.

The legal text for high-value and more regulated contracts typically has considerable semantic complexity and it is important that the smart contract code should be faithful to the legal text. Clack et al[24] explain that the legal documentation for high-value financial contracts contains both operational aspects (that define what actions must be performed) and non-operational aspects (that define the rights and obligations of the parties, the timing of actions, the intention of the parties, the way that disputes should be handled, meta-clauses about the contract itself, and so on). Although the smart contract code is likely to be focused on the operational aspects, these are often intertwined with non-operational aspects (see below) and so it is likely that non-operational aspects must also be represented in the smart contract code.

3 Validation of Smart Contract Code

The translation from legal documentation to smart contract code might be achieved either manually or automatically. It has also been envisaged that, in the future, artificial intelligence techniques might be used.[25]

For small, non-standard, loosely-regulated, contracts the translation from legal documentation to code is likely be done anew for each contract. If the legal documentation is small, and straightforward, then it might be possible and beneficial to automate this translation. Note that it may not be necessary to represent *all* of the legal documentation in computer code; it is likely that only a subset will be encoded.

By contrast, this paper focuses on high-value, standardised, highly-regulated financial contracts are likely to require a highly structured translation to create standardised code that will subsequently be used for many trades. It is probable that these contracts will require comprehensive testing of the smart contract code, and furthermore it is envisaged that testing and debugging will be a large proportion of the code development effort. As mentioned in Section 2.1, for Smart Contract Templates most of this effort will initially occur once for each template in the ISDA document set, with some further testing effort being required as a result of the negotiated Schedule for a specific agreement between a specific set of parties. In the near future, it appears likely that the translation into smart contract code and comprehensive testing of that code will be achieved manually. The need for testing and debugging arises for both manual and automatic translation. In particular, it is important to distinguish between:

(i) **Testing** the smart contract code to ensure it has no errors, and
(ii) **Validating** the behaviour of the smart contract code to ensure that it is faithful to the meaning of the contract. Validation is the more difficult task, since it requires the computer programmer to understand the legal documentation.

The need to ensure that the smart contract code behaves in accordance with the legal agreement has been noted by others:

- Al Khalil et al[26] observe that the gap between the lawyer's semantics and the semantics of the operation of the smart contract code may bring unacceptable operational and regulatory risks. They discuss some of the difficulties that arise when attempting to validate smart contract code – i.e. how to reason about the legality of code, and the accuracy of the operation of the code (either manually, or automatically using a tool to check compliance). They argue that trust in smart contracts, by all stakeholders including regulators, can only stem from the ability of lawyers in financial institutions to validate the semantics of smart contract code; the code must be transparent to stakeholders and have empirical fidelity with the legal documentation on which it is based. They argue that a lawyer's knowledge of the explicit and implicit rights and obligations of the parties, and of the applicable regulations governing a financial contract, must be represented in the smart contract code and that there is currently no single representation that can be understood by both lawyers and computer scientists and that would therefore "bridge the gap" between the two realms of expertise. Thus, the issue of semantics (of the legal text, and of the code) is fundamental.
- Harley[27] observes that smart contracts require "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."
- ISDA and Linklaters[28] identify the problem of how a party accepting a contract offer could be sure that the smart contract code will be faithful to the legal documentation.
- Gibbons[29] observes that because contracts are usually not exhaustive (either due to the practicality of bringing a deal to conclusion, or because certain cases have

not been imagined), it is important that the smart contract code should be validated to confirm the extent of its applicability (for example, that the code would pause and request human help when an unexpected circumstance arises).

Validation of smart contract code is not simply a matter of running the code and checking that no errors occur. Nor is it a matter of checking that output values are correct according to various arithmetic formulae and operational logic. Instead, it requires an understanding of the semantics of the contract to provide a set of validation scenarios (for example, some of these might be "what if" scenarios such as "what if party A makes a partial payment?" or "what if the following sequence of events were to occur?" whereas others might be based in practical experience of issues of law such as "if some of the proposed transactions were to become illegal midway through the contract, could the contract be modified or terminated?"). This may require investment into the training of new staff as hybrid experts in both law and computer science.

4 Semantic Issues

If computer programmers are to undertake the validation of smart contract code, then we might presume that this would start with those programmers having a full understanding of the legal documentation. Such understanding, however, is difficult to achieve.

Computer science and law are complex professions, each with its own specialist language of words with specific meanings. Yet often these are common words that also have a lay interpretation. This may lead to some confusion between computer scientists and lawyers:

- For a lawyer, the word "execution" refers to the signing of an agreement, whereas for a computer scientist it refers to the running of computer code.
- For a lawyer, the word "performance" means the enactment of the terms of an agreement, whereas for a computer scientist it refers to how much computer memory is used by a running program, how long a program takes to run to completion, and how this changes as the input to the program changes.
- For a lawyer, the word "termination" refers to the end of a contract's performance, and for a computer scientist it refers to the end of the running of a computer program. Yet these two might not be coterminous.

4.1 Ambiguity

There is often substantial ambiguity in legal text. Sometimes that ambiguity is deliberate: sometimes not. Some simple words and phrases might appear highly ambiguous, or undefined, to a computer scientist whereas to a lawyer the same words might be commonplace with no need for precise definition. Simple examples include:

- The words, "soon", "promptly", and "timely" – to a computer scientist these might appear to be so context-dependent that they have no meaning. For example, a computer scientist might ask whether "soon" is measured in days, hours, minutes, seconds, milliseconds or nanoseconds?
- The word "or" – does this include or exclude the case where both are true?
- The word "deemed" might be used without stating who is responsible for the "deeming".
- The phrase "upon demand" relates to an unknown time of a possible future event

Deliberate ambiguity might occur for example where it is impossible to list all possible future circumstances, or where it would be prohibitively expensive to define the actions to take in all possible scenarios, or where the parties are unable to agree specific words but are content to agree ambiguous words.

Despite recent statements that computer programs cannot cope with ambiguity, in fact it is straightforward for programs to perform decision-making and actions based on non-binary logic values (e.g. three-valued logic with values "true", "false" and "unknown"), or to manage "fuzzy logic" where the logic values lie somewhere along a spectrum from "true" to "false". However, there will doubtless be examples of ambiguity in legal text that are either too subtle to describe in, or that the stakeholders would not wish to be included in, the smart contract code.

4.2 Complex legal concepts

Perhaps more difficult than ambiguity is the issue of complex legal concepts (which computer scientists might initially describe as "undefined" or "imprecise" terms). Simple examples include the word "reasonable" (to whom? in what context?) and the phrase "if deemed" (by whom? when?).

Harley[30] cites the phrase "reasonable endeavours" as an example and observes that "*Because this type of concept cannot be set out in a formalized language that can provide a well-defined set of instructions for code to execute, we do not see that [there is] any prospect of implementing these until the advent of more powerful AI than is available today, that can make that context-specific determination of reasonableness*."

ISDA and Linklaters[31] similarly note problems with the precise semantics of phrases such as "party", "duly organized", "validly existing", "jurisdiction", "organization" and "incorporation". They conjecture that a sufficiently developed ontology for legal contracts might resolve some of these problems, with the smart contract code checking automatically with relevant company registries to determine whether certain representations are true at the time given. However, they also point out that constructing such an ontology will be difficult since many subtle semantic issues will need to be resolved and agreed by the community of parties that will use the ontology.

Gibbons[32] observes that the phrase "duly organized" also highlights another possible source of confusion, since two lawyers might have different understandings of this term depending on the jurisdiction in which they qualified.

These legal concepts are clearly a challenge, yet perhaps there is no need to represent them in the smart contract code: as mentioned previously, the smart contract code might only encode part of the legal agreement. Whether it is possible to do so without encoding complex legal concepts is currently unknown.

4.3 Modal verbs

Some legal documentation uses complex modal verbs (i.e. verbs that express necessity or possibility, such as "shall/should", "will/would", "can/could", "must/had to"). If it is required to represent text containing modal verbs in the smart contract code, it will be necessary to define the semantics of that text precisely. Yet the meaning of modal verbs can be context-dependent: the word "will" might imply an obligation or might be a prediction, the words "may" and "can" might either indicate a permission or a possibility, and the word "must" might indicate an obligation, or a logical necessity.

Further complexity arises where modal verbs are combined with negations such as "not", "no", "never" or "nothing", with exceptions and imagined scenarios, and perhaps when expressed in the subjunctive mood to indicate something that is hypothetical,

imagined, wished or possible. This also links with temporal issues, such as text that relates to conditional past or conditional future events or states, or both.

Section 9(h)(i)(3) of the ISDA 2002 Master Agreement is an illustrative example of legal drafting with complex semantics:

> "*Interest on Deferred Payments. If:-*
>
> *(A) a party does not pay any amount that, but for Section 2(a)(iii), would have been payable, it will, to the extent permitted by applicable law and subject to Section 6(c) and clauses (B) and (C) below, pay interest (before as well as after judgement) on that amount to the other party on demand (after such amount becomes payable) in the same currency as that amount, for the period from (and including) the date the amount would, but for Section 2(a)(iii), have been payable to (but excluding) the date the amount actually becomes payable, …"*

5 Semantic Analysis

A formal semantic analysis of the legal documentation would provide a firm foundation for generating a set of validation scenarios, thereby improving the quality of validation of smart contract code. It could also help code development in other ways:

- To clarify the runtime information that will be needed by the smart contract code, thereby improving the quality of that code.
- To clarify obscure clauses and potentially equally obscure code, possibly helping to expose errors in the code.
- To facilitate encoding more parts of the legal documentation than might otherwise be feasible.

Semantic analysis of the legal documentation might also provide further benefits to lawyers during contract authoring:

- To detect missing cases.
- To detect logical inconsistencies.
- To provide "what if" analysis so that lawyers can check complex reasoning.
- To provide a "standardised" human-readable form that could be compared and contrasted with the authored version.

Legal documentation is complex and subtle, and many different types of semantic analysis could be applied. For example, it is possible to analyse the **temporal** aspects of the contract – this includes *inter alia* fixed points in time, the time of an event occurring, the time elapsed since the occurrence of an event, intervals of time, and points in time in conditional futures. Another important type of semantic analysis assesses the **deontic** aspects – this refers to the rights, obligations and prohibitions applying to each party. A third example is the **operational** semantics of a contract – this analyses the required actions, some of which may not be encodeable (or should not be encoded, if they must always be subject to human discretion).

A good policy might be to apply as many different types of semantic analysis as necessary to expose the important semantic aspects of the contract. This may vary between contracts, but it appears likely that at least the three examples given above (temporal, deontic and operational semantics) will be necessary for most contracts.

6 Separability and Interaction

A fundamental question about the semantics of contracts is whether the different types of analysis are syntactically separable, in the sense that it is possible to identify points in the legal text where (for example) a deontic aspect ends and a temporal aspect starts. This would facilitate analysis, and would encourage programmers that there is likely to be a purely operational part of a contract that can be extracted and turned into code.

However, it appears that the different semantic aspects are not generally separable. For example, temporal and operational aspects are often linked – consider the phrase "*upon demand*" which combines the occurrence of an action (one party makes a "demand") with the time at which the action was taken (the word "upon").

Furthermore, operational, temporal and deontic aspects are often linked. Consider the following phrase taken from Section 9(f) of the ISDA 2002 Master Agreement: *"…a single or partial exercise of any right, power or privilege will not be presumed to preclude any subsequent or further exercise, of that right…*" which combines operational aspects ("a single or partial exercise", "exercise"), temporal aspects ("subsequent") and deontic aspects ("of any right, power or privilege", "will not be presumed to preclude", "of that right").

To illustrate the importance of semantic analysis for the validation of smart contract code, consider that if the legal text in the above example were included in that code, two validation scenarios might be adduced:

1. The code should be capable of identifying each exercise of such right, power or privilege as a separate event.
2. The code should be able to exercise such right, power or privilege exactly once on each occasion, even if previously exercised.

Many clauses have a complex mix of deontic, operational and temporal aspects. The reader is invited to consider the complexity of separating the deontic, temporal and operational aspects of Section 2(d)(ii) of the ISDA 2002 Master Agreement:

> "*Liability. If:-*
>
> *(1) X is required by any applicable law, as modified by the practice of any relevant governmental revenue authority, to make any deduction or withholding in respect of which X would not be required to pay an additional amount to Y under Section 2(d)(i)(4);*
>
> *(2) X does not so deduct or withhold; and*
>
> *(3) a liability resulting from such Tax is assessed directly against X.*
>
> *then, except to the extent Y has satisfied or then satisfies the liability resulting from such Tax, Y will promptly pay to X the amount of such liability (including any related liability for interest, but including any related liability for penalties only if Y has failed to comply with or perform any agreement contained in Section 4(a)(i), 4(a)(iii) or 4(d)).*"

7 Non-Operational Semantics of Smart Contract Code

It would greatly encourage the programmers of smart contract code if it were possible to identify the purely operational parts of a contract. These could then be encoded without requiring the programmer to consider complex deontic or temporal aspects.

However, this expectation falls at the first hurdle when we consider that in a contract all actions derive from an obligation or right (a deontic aspect) of some form or other. Parties to contracts generally don't take any action other than those they are obliged to take or have the express right or permission to take.

Furthermore, many actions have embedded temporal aspects (trivially) and may have deontic aspects embedded within the operational text. For example, if the smart contract code for the ISDA 2002 Master Agreement were required to check all payments, and if the code were to detect a failure to pay, then Section 5(a)(i) states that it *is obliged* (deontic) to send notice of failure to the other party (operational), and *is obliged* (deontic) to check again (operational) in *one Local Business Day* (temporal).

Further examples of deontic aspects linked to operational aspects include discretionary actions (discretion is a deontic aspect), and any code whose purpose is to detect (operational) a breach of an obligation (deontic).

A final example from the ISDA 2002 Master Agreement, Section 2(c) illustrates the complexity that must be embraced if it were decided to include support for intraday netting in the smart contract code:

> "*Netting of Payments. If on any date amounts would otherwise be payable:- (i) in the same currency; and (ii) in respect of the same Transaction, by each party to the other, then, on such date, each party's obligation to make payment of any such amount will be automatically satisfied and discharged and, if the aggregate amount that would otherwise have been payable by one party exceeds the aggregate amount that would otherwise have been payable by the other party, replaced by an obligation upon the party by which the larger aggregate amount would have been payable to pay to the other party the excess of the large aggregate amount over the smaller aggregate amount. …*"

One of the interesting characteristics of the above example is that it appears to require that an action (operational) will dynamically replace previous rights and obligations (deontic) with new rights and obligations (deontic).

8 Pragmatics

A formal understanding of the semantics of legal words and phrases is unlikely to be sufficient. It will also be necessary to understand legal "pragmatics", where this term is used informally to indicate a shared context between the writer and reader that may not be evident from a semantic analysis. This shared context can alter the overall meaning of the written text.

The pragmatics of legal contracts captures the shared understanding between the writers and the readers of these contracts, where both writers and readers are (or are assumed to be) lawyers. This shared understanding includes *inter alia* pre-suppositions, implications (e.g. whether posted collateral gives rise to regulatory obligations), and the role of law. The matter of the role of law is partly overt (contracts often contain statements on governing law and jurisdiction) and partly hidden (in English law contracts are subject to implied terms, some of which may be mandatory and others that can be excluded when reasonable).

The context shared between expert lawyers is unlikely to be understood by expert computer scientists (and *vice versa*) and this "Pragmatic Gap" may be a formidable problem for the validation of smart contract code.

Harley[33] presents an example of the Pragmatic Gap where a smart contract stores on a blockchain a record of the transfer of ownership of real-estate assets, but neglects to perform the legally required actions to register the change of ownership in the human world – and thus a court of law would not accept the change of ownership.

As hinted above, the Pragmatic Gap operates in two directions: not only are legal pragmatics hidden from computer scientists, but the pragmatics of computer science are also hidden from lawyers. For example, computer scientists have a shared understanding that it is straightforward to implement multi-valued logics (as long as the rules of that logic are defined) and why it is important never to test two real-valued data items for equality, yet this expert knowledge will not be apparent to lawyers when reading smart contract code.

The Pragmatic Gap is further compounded by the "Isomorphism Problem" that exists when different representations of the legal documentation are compared. The human readable representation of a contract is typically not isomorphic to (does not have the same form as, and does not have a one-to-one correspondence with) a formal semantic representation of the same contract. Nor is it isomorphic to the smart contract code. This is because a single semantic expression often derives from many clauses of a contract, and alternatively a single clause of a contract might be expanded into more than one semantic expression.

This Isomorphism Problem makes it difficult to identify a direct relationship between individual clauses of the contract and individual lines of code (in the smart contract code) or individual semantic expressions (in the formal semantic model), and therefore makes it difficult for a lawyer to understand and validate smart contract code.

9 Research Directions

Further research is urgently required in this area, to fully understand the semantics and pragmatics of legal contracts and to explore the previously described Isomorphism Problem. Since Smart Contract Templates support the automation of high-value financial transactions, an appropriate starting point might be to explore the semantics of the ISDA Master Agreement for derivatives contracts. Figure 3 illustrates an initial roadmap for this research: the aim will be a formal description of the meaning of the ISDA Master Agreement, from which validation scenarios could be constructed to support the development of the associated smart contract code.

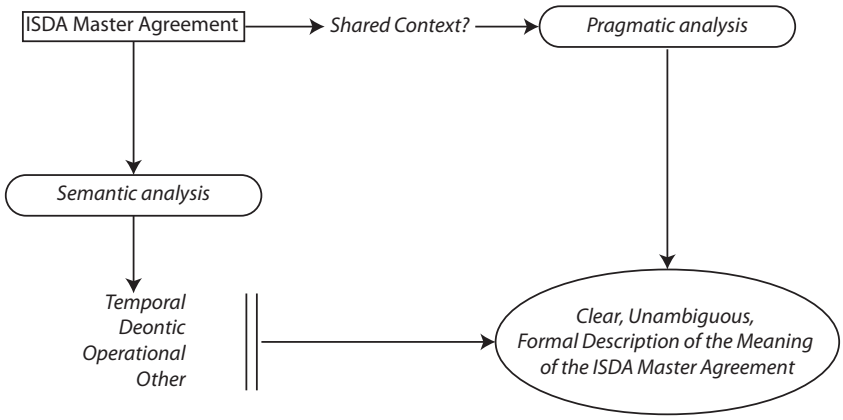

Figure 3: an initial research roadmap for understanding the semantics and pragmatics of the ISDA Master Agreement

10 Summary and Conclusions

Smart Contract Templates support legally-enforceable smart contracts, using operational parameters to connect legal agreements to standardised code. The standardised code (the "smart contract code") is derived from legal documentation (the "smart legal contract") and performs some or all of the provisions of that contract.

An important issue is how to know whether the smart contract code will correctly perform the provisions of the smart legal contract. During development of smart contract code it is important to perform two steps:

(i) **Testing** the smart contract code to ensure it has no errors, and
(ii) **Validating** the behaviour of the smart contract code to ensure that it is faithful to the meaning of the contract.

Many difficulties arise when attempting to validate smart contract code – i.e. how to reason about the legality of code, and the accuracy of the operation of the code (either manually, or automatically using a tool to check compliance). Development of smart contract code for large, high-value, legal agreements is a complex matter – it is not simply a matter of "lifting" operational phrases out of the legal text and turning them into code, because operational and non-operational aspects of the agreement are intertwined. Further, the "pragmatic gap" between expert lawyers and expert computer scientists is sufficiently large that misunderstandings are likely to occur.

A formal semantic analysis of the legal documentation would provide a firm foundation for generating a set of validation scenarios, thereby improving the quality of validation of smart contract code. It could also clarify the runtime information that will be needed by the smart contract code, thereby improving the quality of that code, clarifying obscure clauses, and perhaps making it possible to encode more parts of the legal documentation than might otherwise be feasible.

Semantic analysis of the legal documentation might also provide further benefits to lawyers during contract authoring: to detect missing cases and logical inconsistencies, to provide "what if" analysis to help check complex reasoning, and perhaps to provide a standardised human-readable form.

A thorough understanding of the semantics and pragmatics of legal text is an important step towards building trust by all stakeholders, including regulators, and towards supporting industry adoption of legally-enforceable high-value smart contracts. This paper clarifies the various languages and representations involved in the development of smart contract code, sets out many of the challenges associated with understanding the meaning of legal documentation, and suggests an initial research roadmap for exploring the semantics and pragmatics of the ISDA Master Agreement for financial derivatives.

We are currently pursuing the research roadmap illustrated in Figure 3, exploring three parallel themes: (i) liaising with lawyers and bankers to assist in developing a shared ontology of definitions of words and phrases, (ii) exploring different semantic analyses of the ISDA Master Agreement, and (iii) liaising with lawyers to identify issues of shared context and pragmatics in relation to the standard ISDA documentation.

References

1 Braine L. *Barclays' Smart Contract Templates*. [Presentation] Barclays London Accelerator. 2016. Available from: https://vimeo.com/168844103/ [Accessed 31st August 2017].

2 Allison, I. *Barclays' Smart Contract Templates stars in first ever public demo of R3's Corda platform*. Available from: http://www.ibtimes.co.uk/barclays-smart-contracttemplates-heralds-first-ever-public-demo-r3s-corda-platform-1555329/ [Accessed 19th April 2016 and 31st August 2017].

3 Clack CD, Bakshi VA, Braine L. *Smart Contract Templates: essential requirements and design options*. Available from: https://arxiv.org/pdf/1612.04496.pdf [Accessed December 2016 and 31st August 2017].

4 Clack CD, Bakshi VA, Braine L. *Smart Contract Templates: foundations, design landscape and research directions*. Revised 15th March 2017. Available from: https://arxiv.org/pdf/1608.00771.pdf [Accessed 31st August 2017].

5 Stark J. *Making sense of blockchain smart contracts*. Available from: http://www.coindesk.com/making-sense-smart-contracts/ [Accessed 20th June 2016].

6 Axoni. *AxCore*. Available from: https://axoni.com/ [Accessed 31st August 2017].

7 Brown RG, Carlyle J, Grigg I, Hearn M. *Corda: An introduction*. Available from: https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/57bda2fdebbd1acc9c0309b2/1472045822585/corda-introductory-whitepaper-final.pdf [Accessed August 2016].

8 Digital Asset. *The Digital Asset Platform - Non-technical White Paper*. Available from: http://hub.digitalasset.com/hubfs/Documents/Digital%20Asset%20Platform%20-%20Non-technical%20White%20Paper.pdf?submissionGuid=19b3704a-4934-4661-9920-2270d03db39 [Accessed 31st August 2017].

9 Ethereum. *Ethereum*. Available from: https://www.ethereum.org/ [Accessed 31st August 2017].

10 Hyperledger. *Welcome to Hyperledger Fabric*. Available from: https://hyperledger-fabric.readthedocs.io/en/latest/ [Accessed 31st August 2017].

11 Clack CD, Bakshi VA, Braine L, ref 3 above.

12 Clack CD, Bakshi VA, Braine L, ref 4 above.

13 Clack CD. *Smart Contract Templates: the Semantics of Smart Legal Agreements*. [Presentation] The Third R3 Smart Contract Templates Summit. June 2017. Available at: https://www.r3.com/slides/third-smart-contract-templates-summit-slides.pdf [Accessed 31st August 2017].

14 Stark J, ref 5 above

15 Clack CD, Bakshi VA, Braine L, ref 4 above.

16 Gibbons K. Personal communication. Clifford Chance LLP; 2017.

17 ISDA and Linklaters. *Smart Contracts and Distributed Ledger – A Legal Perspective*. Available from: http://www2.isda.org/attachment/OTU3MQ==/Smart%20Contracts%20and%20Distributed%20Ledger%20%20A%20Legal%20Perspective.pdf [Accessed 31st August 2017].

18 Harley B. *Are Smart Contracts Contracts?* Clifford Chance. Report. Available from: https://onlineservices.cliffordchance.com/online/freeDownload.action?key=OBWIbFgNhLNomwBl%2B33QzdFhRQAhp8D%2BxrIGReI2crGqLnALtlyZe4BCvqTOr3az7SqWBWY%2Bn5%2Fp%0D%0A5mt12P8Wnx03DzsaBGwsIB3EVF8XihbSpJa3xHNE7tFeHpEbaeIf&attachmentsize=1861746 [Accessed 31st August

2017].
19 R3 and Nortin Rose Fulbright. *Can Smart Contracts be Legally Binding Contracts*. Available from: http://www.nortonrosefulbright.com/knowledge/publications/144559/can-smart-contracts-be-legally-binding-contracts [Accessed 31st August 2017].
20 Legalese. *Legalese*. Available from: http://legalese.com/ [Accessed 31st August 2017].
21 Clack CD, Bakshi VA, Braine L, ref 3 above.
22 CommonAccord. *CommonAccord*. Available from: http://www.commonaccord.org/ [Accessed 31st August 2017].
23 Hazard J, Haapio H. Wise Contracts: Smart Contracts that work for people and machines. In: Schweighofer E et al (eds.) *Trends and Communities of Legal Informatics: Proceedings of the 20th International Legal Informatics Symposium IRIS 2017*. Osterreichische Gesellschaft. ISBN 978-3-903035-15-7; 2017. p. 425-432.
24 Clack CD, Bakshi VA, Braine L, ref 4 above.
25 Harley B, ref 18 above.
26 Al Khalil F, Ceci M, O'Brien L, Butler T. *A Solution for the Problems of Translation and Transparency in Smart Contracts*. Government Risk and Compliance Technology Centre. Report, 2017. Available from: http://www.grctc.com/wp-content/uploads/2017/06/GRCTC-Smart-Contracts-White-Paper-2017.pdf [Accessed 31st August 2017].
27 Harley B, ref 18 above.
28 ISDA and Linklaters, ref 17 above.
29 Gibbons K, ref 16 above.
30 Harley B, ref 18 above.
31 ISDA and Linklaters, ref 17 above.
32 Gibbons K, ref 16 above.
33 Harley B, ref 18 above.