Received 3 February 2017
Revised 24 May 2017
Accepted 24 September 2017

From digital currencies to digital finance: the case for a smart financial contract standard

Willi Brammertz
ACTUS Users Association, Rockville, Maryland, USA and Ariadne Business Analytics, Naenikon, Switzerland, and

Allan I. Mendelowitz
ACTUS Financial Research Foundation, Rockville, Maryland, USA

Abstract

Purpose – This paper aims to demonstrate the importance of a cash flow generating standard for individual financial contract level data and the ability to create such a standard.

Design/methodology/approach – The authors analyze the importance for such a standard of software that turns natural language contracts into cash flow generating algorithms; a data dictionary that standardizes contract terms; and access to variables that represent the state of the world (e.g. market risk, counterparty risk, etc.) that affect contractual obligations.

Findings – The ability to realize benefits from the use of such a contract level algorithmic standard depends on the following: making the standard's software open source; fully testing the software to have complete confidence in its accuracy; and enabling the software to use of a wide range of models of various sources of risk (market, credit and behavior risk) to support forward-looking analysis. Such a standard would solve the disconnect that exists in financial firms between the representation of financial contracts for transaction processing and analysis. The ACTUS Financial Research Foundation is building, testing and making available such a standard that represents almost all financial contracts extant in markets.

Practical implications – The adoption of such a standard would reduce the costs of operations of financial firms, provide the computational infrastructure for more effective regulatory oversight, reduce regulatory reporting costs and improve financial market transparency. It would also enable the assessment of systemic risk by directly quantifying the interconnectedness of firms.

Originality/value – This is a new approach to financial analytics that clearly separates the deterministic components of finance, which can be standardized from the stochastic elements that cannot be standardized.

Keywords Standardization, Fintech, Block-Chain, Complete contracts, Smart contract, Smart financial contracts

Paper type Conceptual paper

Uniqueness of financial contracts

The role of contracts in economics has recently received some high-profile attention. The Nobel Memorial Prize in Economics was awarded to two distinguished economists for parsing the challenges of contracts in a market economy[1]. In addition, the rapidly growing interest in finance and information technology – under such rubrics "Fintech" and "Block-Chain" – has focused great attention on financial contracts in general, and "smart contracts"[2] in particular. This attention is not merely an academic interest; it is a practical interest of the finance industry as it tries to develop better ways to organize and manage the business of finance.

The Journal of Risk Finance
Vol. 19 No. 1, 2018
pp. 76-92

1526-5943
DOI 10.1108/JRF-02-2017-0025

In the rush to explore how to take advantage of the potential of Fintech, there has been less attention paid to what is needed in the way of the foundational elements. Block-Chain has received enthusiastic attention. However, the only sustained application of the technology has been for single payment transactions in a crypto currency. It has attracted traders desiring access to a payments mechanism free from the legal and regulatory constraints intended to prevent money laundering; avoid restrictions on financial transfers to the targets of foreign policy sanctions; and make it harder to trade in elicit goods and services. The fact that most financial contracts are long-lived multi-payment obligations means that the real-world experience with Block-Chain and Bitcoin has little to say about how to advance Fintech.

Contracts are critical elements of the operation of a market economy. This observation is all the more true in in finance. What has been missing in these discussions of Fintech and Block-Chain has been the recognition of the uniqueness of the financial contract within the broader world of economic contracts. Understanding this uniqueness is critical to understanding how to represent financial contracts in real world applications of Fintech. Furthermore, understanding this uniqueness is critical to being able to realize the promised benefits of Fintech.

The uniqueness of financial contracts starts with the nature of what is being exchanged under the contracts and goes on from there:

- Most economic contracts consist of an obligation to pay money in exchange for the receipt of goods and/or services. Financial contracts are agreements to exchange cash flows on the part of all the counterparties to a contract without the exchange of any goods or services.
- Cash flows are, in essence, numbers. Such numbers can precisely represent the payment obligations contained in a financial contract.
- Financial contracts typically require the payment of a series of cash-flow obligations that are most precisely represented by a mathematical expression rather than the words of a natural language contract. In fact, financial contracts – the exchange of cash flows – are the only contracts that can be perfectly represented mathematically.
- Because the obligations of financial contracts can be represented mathematically with greater precision than natural language contracts, such contracts are well suited to be represented by self-executing software.

This uniqueness should have enabled the financial industry to be in the forefront with regard to automation when compared to other industries. The reality however is different.

A closer look at the financial contract

"Judex non calculat" is an old roman saying, which can be roughly translated as "Lawyers do not calculate"[3].

This aversion to mathematics may be one reason financial contracts are written in natural language. Another reason is that contracts have other terms and conditions that are not directly related to calculating cash flow obligations and require their representation in natural language. In any event, the current state of affairs is that in financial contracts, clear mathematical expressions that define the exchange of cash are turned into words.

Some of the terms, the "essentialia negotii" are expressed in numbers such as the notional amount exchanged, the interest rate, term of the agreement and amortization of principal.

They are the parameters of formulas hidden in the text. However, at the end of the day, they must be ultimately interpreted as a numerical obligation.

In the following, we will argue that the text of a financial contract that relates directly to computing the exchange of cash flows should be represented by well-defined and openly published algorithms. However, there are other aspects of financial contracts that still need to be represented in natural language. In cases of technical events of default or delayed payment or nonpayment of obligations, human judgment is critical to determining how best to respond. For example, in the case of a debt instrument, the natural language terms of a contract would clearly define rights and obligations in a default. However, what actually transpires requires human interaction and is left to the judgment of the creditor. Once an obligor falls behind in payment obligations, the resolution of the default can involve workout negotiations or litigation, with lawyers and judges playing central roles in the ultimate resolution, a process not coverable by mere algorithms.

There is much more that can be said about a credit default event. Algorithms can be used to provide analytical support for the resolution of a default. However, the length of this article does not allow us to dwell on this issue, and we leave this topic to another article. In this paper, we focus solely on the exchange of cash that, in the modern world, covers the overwhelming majority of the real-world financial transactions.

How financial institutions work in practice

A financial contract is an agreement to exchange of cash flows that can be expressed mathematically. However, legal contracts have always been expressed in natural language. The reliance on natural langue has obscured the fundamental mathematical nature of the relationship, which has had major consequences for the financial sector.

To understand the nature of the problem, we present a simplified organizational model of a financial institution, using a typical bank for our example.

Such a bank has two key functions or levels – a transaction processing level (TPL) and an analytic level (AL):

(1) *TPL*: It computes on a daily basis both the bank's payment obligations and expected contractual receipts for that day. It is responsible for maintaining the records of what is added to and runs off of the balance sheet on a day-by-day basis and any required payments during the life of the asset or liability. These two functions are generally referred to as deal acquisition and deal processing. For example, a mortgage is made by the bank and maintained on its balance sheet. After it is created, it is entered into the transaction processing systems (TPS) that handles all the payments of the transaction till maturity date.

(2) *AL*: Bank management is essentially the management of numbers, and the analytic level corresponds to the bank management level. Bank management can be roughly grouped into the three views of the bank: risk management, finance and regulatory reporting.

Financial analysis means analysis of the contracts maintained in the TPS. Therefore, it should start with the same cash-flows obligations maintained and managed by the TPL. This perspective gives direct insight into the value of smart financial contracts for more efficient and higher quality bank operations. Because both the TPL and the AL start fulfilling their tasks using the same cash flow generated by the individual financial contracts, the same smart financial contracts – data representing the contract terms and algorithms – can be used by both the TPL and the AL. The reality, however, is far different from this ideal solution.

Transaction processing and financial contracts

On the TPL, the algorithmic nature of the financial contract is well understood and exploited. The implementation of fully automated transaction processing systems started when the rapid growth in financial activities overwhelmed the ability of banks to manually process transactions. This happened mainly from the 1970s to the 1990s.

Today, almost every financial contract is represented by machine-readable code. Contracts are entered into the system at their origination. All payments are automatically calculated and often automatically executed. Straight through processing has been achieved in this process to a far extent.

Many practitioners consider the machine representation of the contract as the true contract. The paper version has hardly any significance as long as the contract is in good standing. This fact alone provides strong support for our view of the algorithmic nature of financial contracts. TPS are, in essence, cash-flow generators **(CFLGs)**.

A CFLG contains certain functions that calculate interest, principal, new rates if the contract includes a variable interest rate, etc. It needs as input the contract terms with which it can generate all future cash flows that include the date, amount and currency for each computed payment obligation.

If the future cash flow obligations depend on changes in the market, such as the market rate needing to reset the interest rate on the variable rate of a swap, a link to these conditions must exist to calculate the new rate.

We call these market terms risk factors as they can change at any time in the future in an unforeseen manner. Market risk factors are interest rates and foreign exchange (FX) rates. However, from a forward-looking analytical perspective risk factors also include credit risk and behavioral risk:

$$\text{In short}: \ \text{TPS} \ == \mathbf{CFLG}(\text{contract terms}, \ \text{risk factors}) \tag{1}$$

The "==" denotes "corresponds to".

If a smart contract is self-executing software that faithfully generates the cash flow obligations of the natural language contract, we can argue that banks have been using smart financial contracts since the 1970s.

The analytic level: how it should be

The AL analyzes the contracts that reside in the TPS. In addition, the AL analyzes the impact of the risk factors on the actual bank position and possible future or planned business.

It is a generally true proposition that *any* financial analysis (FA) depends on the expected future cash flow.

$$\text{FA}^{\text{i}} = \text{f}^{\text{i}}(\text{E}[\text{cfl}]) \text{ where the superscript i refers to the type of analysis} \tag{2}$$

The analysis result i could be a liquidity gap report, a sensitivity analysis, an FRS accounting statement, a liquidity coverage ratio (LCR) calculation, etc.

It would seem logical that a bank would use the same **CFLG** functions for the analysis it uses for transaction processing. After all, the principal difference between transaction processing and analysis is the time dimension and its implications. Transaction processing is done one day at a time using observed risk factors on that day to compute payment obligations. Such current risk factors as interest and FX rates can be retrieved from Bloomberg or Reuters.

Even though the AL should theoretically start with the same algorithms as the TPL, analysis is a more complex challenge. Expected future cash flow depends on the unknown future state of the risk factors, which are market (interest rates and FX rates), counterparty (credit) risk and behavior risk (i.e. prepayment risk). Because analysis is forward-looking, AL needs forecasted future market conditions and forecasted states of the other risk factors including future business developments. In addition, the analytical tasks need the capability to derive from the expected cash-flows basic analysis such as liquidity, income and value for the balance sheet, plus additional functionality for specific reports. Additionally, the analytical engine needs the ability to sort and aggregate by any desired criterion the granular level expected future cash flows to gain greater insight into the condition of the business.

Taking this into account we can extend equation (1) as follows:

$$\mathrm{FA}^{\mathrm{i}} = \mathrm{fa}^{\mathrm{i}}\left(\left(\mathbf{CFLG}\left(\text{contract terms},\ \text{risk factors},\ \mathrm{E}[\text{risk factors}]\right),\ \mathrm{F}^{\mathrm{i}}\right)\right) \tag{3}$$

fa^{i} nests the **CFLG** function with expected risk factors and the F^{i} function. The F^{i} function is a special operator that produces the specific result FA^{i} such as liquidity gaps, balance sheets and so on.

The relationship between **CFLG** and the expected risk factors in the AL space is the same relationship as between **CLFG** and the current risk factors in the TPL space. Also, the construction of the expected risk factors is an entirely orthogonal element that is handled outside the **CFLG** and supplied the same way as the observed risk factors that are used for transaction processing.

The F^{i} function could for example be an LCR operator or any other operator. The F^{i} function always operates on the expected cash-flows. As it operates on the results only, it is independent of the **CFLG** function without any loop back.

We, therefore, can conclude that the same **CFLG** can be used within TPS and within any FA^{i}, save that the **CFLG** function must be able to operate with the forecasted variables for future periods.

From equation (3), we can also see how the deterministic part of finance interacts with the stochastic part. Stochastic elements enter the computations solely through the states of the future unknown risk factors that can be represented by stochastic models. The algorithms that represent the legal contractual obligations are the deterministic component of the financial world.

From this, we can see that there should be no essential difference between how contracts are represented at the TPL and the AL.

This, however, is not the case in practice.

The analytic level: how it is in reality

In practice, we find a wide chasm between TPL and AL in all financial institutions.

The reason why AL <> TPL lies in the unstandardized nature of the TPSs. Despite the fact that there is only a very limited number of different financial transactions in terms of cash-flow patterns, there are thousands of different TPSs out there. There are even multiple TPSs within a single bank that handle the same transactions. In addition, TPSs are generally highly parameterized, which means that the same system can produce very different patterns depending on the given instance.

This means that the same cash flow pattern is generated by many different TPSs in different ways. Not only do they have different programming languages and different

names and meanings for the various contract terms but they also use very different logic. Each system has its own logic and algorithms. Instead of having a unique **CFLG** for a specific cash flow pattern, as in equation (1), we have multiple versions of them.

Despite these differences, the power of the financial logic in the contracts yields results that are considered to be "good enough" for running a bank. The final numbers generated by the different TPSs are not identical, but they are close enough. The differences are a nuisance, but small enough in magnitude to be settled amicably between the counterparties. In the real world business, the differences are subjected to a "tolerance" test. For example, if the difference in the computations of the two counterparties is within tolerance (i.e. $50 or $25) the difference is acceptable. However small the differences are, they do impose significant reconciliation costs on the banks.

While this unstandardized nature is a nuisance to the TPL, it becomes a quagmire on the AL.

A necessary condition on the AL is the ability to aggregate the results of all financial contracts from all TPSs.

From equation (2), it is clear that all these results depend on expected cash flows. From equation (3), however, we can see that expected cash flows depend on the **CFLG** *including* forecasted or expected risk factors.

Risk and financial reports, and, increasingly, even regulatory reports, depend on expected risk factors. Stress tests subject financial institutions to specific stress scenarios, such as a sharp rise in the yield curve. These expected market conditions must be applied consistently across all financial contracts. For this reason, it is not possible to use precalculated results from the **CFLG** of the TPS. It is necessary to calculate the impact of the different risk scenarios at the AL level.

While it is already difficult to reconcile different attributes such as maturity date, principal and interest rate, the problem becomes insurmountable when we have to reconcile different algorithms based on different logic.

When trying to build analysis systems, banks, therefore, faced a serious problem in dealing with the chaos created by the lack of standardization in their TPSs. In an attempt to resolve this problem, the banks created central data stores or data warehouses (DW). Significant resources were devoted to these initiatives. However, they did not solve the problem.

The reason DWs do not solve the problem is that the only data that was *moved* to the DWs were *the terms* of financial contracts: principal, interest rate, maturity date, etc. *The algorithms* used by the TPS to generate the cash flow obligations were *left behind* and were never standardized. Consequently, while the analysis systems were able to retrieve the terms of individual contracts from the DWs, they had to recreate the wheel by building their own algorithmic representations of the contractual obligations within the AL.

The wheel is reinvented time and again. Almost each little analysis has its own **CFLG** function. In this architecture, we end up with myriad **CFLG** functions on the analytic level producing contradictory results and incurring huge reconciliation cost.

Solving the problem

Cash flow generating standard

The problem can only be overcome if we build a system based on standard **CFLG** functions for all individual financial contracts. We can then map the existing real world contracts into this **CFLG** standard. Because the starting point for the analysis is at the level of the individual financial contract, the AL has maximum flexibility. While there is no way to aggregate different algorithms – even if they lead to same final results – due to the complex

nature of algorithms, this can be overcome with standard algorithms. If standard **CFLG** functions are used to produce the expected cash flows, all derived results can be safely aggregated. This is the necessary basis for BCBS 239 and for any efficient high-quality analysis[4].

For each relevant and known cash flow pattern, we need a specific version of **CFLG**, which we can call $\mathbf{CFLG^i}$. The "i" represents the index for a specific cash flow pattern. The standard must be published as an open source code to assure accuracy and a wide spread confidence in the code. These functions then work principally as APIs. This is best explained by a simple example.

Let us call $\mathbf{CFLG^1}$ principal at maturity (PAM). Most bonds follow for example such a PAM pattern. A simple fixed rate PAM needs the following fields to be defined: notional, initial exchange date, maturity date, interest payment cycles and interest rate. With those terms and the PAM-specific **CFLG** algorithm, we can precisely generate the cash flow obligations to be used in transaction processing. In addition, by modeling the stochastic market risk factors, we can also undertake the full range of forward-looking analyses with respect to value, liquidity and income.

The following graph represents the cash flow pattern of a simple fixed-interest rate PAM. The solid red lines are the principal payments. The dotted green lines represent the accrual of interest between interest payments. The solid green lines represent the actual interest payment (Figure 1).

This approach sounds good in theory. However, the PAM represents a very simple set of contractual obligations, and at first blush, it is not clear how representative it is of the proposed solution. There are thousands or even tens of thousands of different complex financial products. How is it possible to come up with a manageable standard to represent so many different types of contracts?

The astonishing truth is that this complexity is artificial. The diversity is superficial, created by overlapping concepts such as marketing and legal considerations. By focusing on the underlying cash-flow patterns, the system collapses into a manageable number of

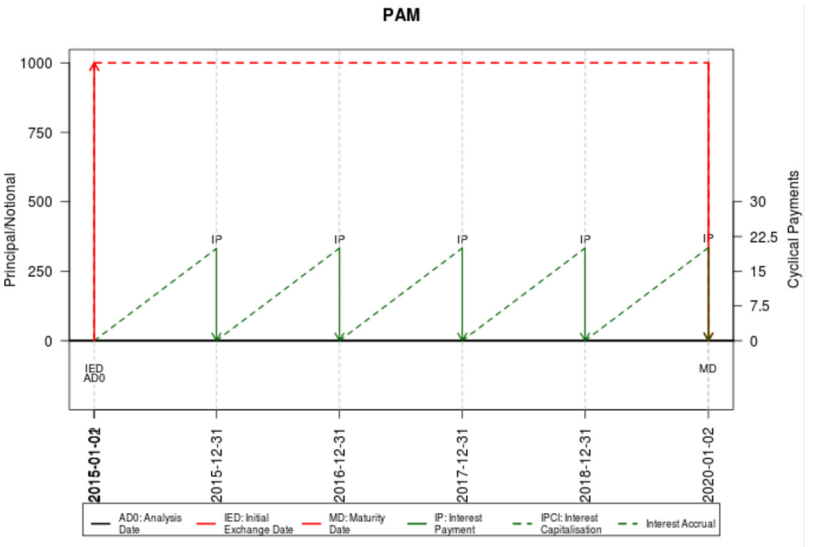

Figure 1.
Events and cash-flows of a PAM contract

standard **CFLG**^{i} algorithms. It will take less than three dozen **CFLG**^{i} algorithms to represent almost all financial contracts. We accept that there will be a small number of rare exotic instruments that are too costly to be standardized, but they will never be a significant part of the market. As soon as an exotic instrument becomes sufficiently significant, the standard has to be expanded to include it. This, however, happens rarely[5].

The breadth of the coverage of the PAM algorithm helps make the case that a manageable number of **CFLG** algorithms are needed. In addition to a simple fixed rate instrument, the PAM algorithm faithfully represents the cash flow obligations of an adjustable rate instrument, a zero-coupon bond, obligations in foreign currencies and instruments with imbedded optionality. Furthermore, the PAM algorithm is the building block for the SWAPS algorithm. A simple plain vanilla interest rate swap has two legs: one is a PAM with a fixed-rate interest obligation; the second is a PAM with a defined adjustable-rate interest obligation. The same approach can be used for basis swaps and for swaps involving foreign currencies.

A second example involves two financial products that banks see as entirely different products and are sold by different parts of a bank. One such instrument is an annuity sold by the retirement/investment silo of a bank. Such an instrument generates a repayment obligation that includes both interest and principle in each payment. Such a contract is represented by the annuity (ANN) **CFLG** algorithm. However, as it turns out, the ANN algorithm also faithfully generates the cash flow payments of a self-amortizing mortgage. Both an annuity and a self-amortizing mortgage have exactly the same cash flow pattern.

The foundation for a **CFLG** standard includes the following:

- identifies and categorizes observed patterns of cfl exchange (**CFLG**^{i});
- builds a ata dictionary (DD) with the contract terms that are needed to run specific **CFLG**^{i} algorithms, where the DD consists of the inputs for the algorithms and includes terms such as notional, contract date, payment cycles, interest rate determinants, day-count method and business-day conventions;
- programs the software for a full set of algorithms that precisely generate the various specific cash flow patterns that underlie the standard; and
- makes the code for each **CFLG**^{i} available as open source code so that all may use it.

ACTUS: the algorithmic contract types unified standard

The ACTUS Financial Research Foundation is building exactly the type of **CLFG** algorithmic standard discussed in this paper. In fact, the effort has already programmed 18 contract types ((CTs), which is the ACTUS terminology for the standard's CFLG algorithms). The completed CTs represent close to 90 per cent of the contracts outstanding in the world of banking. More details about this effort can be found at the ACTUS website[6].

The following graph lays out where the ACTUS CTs' fit into an analytical model for better and a more efficient bank data management and analysis (Figure 2).

The system

The center of the system is the financial contract represented by the terms of the contract and the corresponding CFLG functions that interact with the risk factors. The output of the function are first the contract events that include any happening during the life of a contract, such as a fixing of an interest rate, a payment of interest or principal and an option fixing. From the contract events, it is possible to derive expected cash flows, from which it is possible to derive

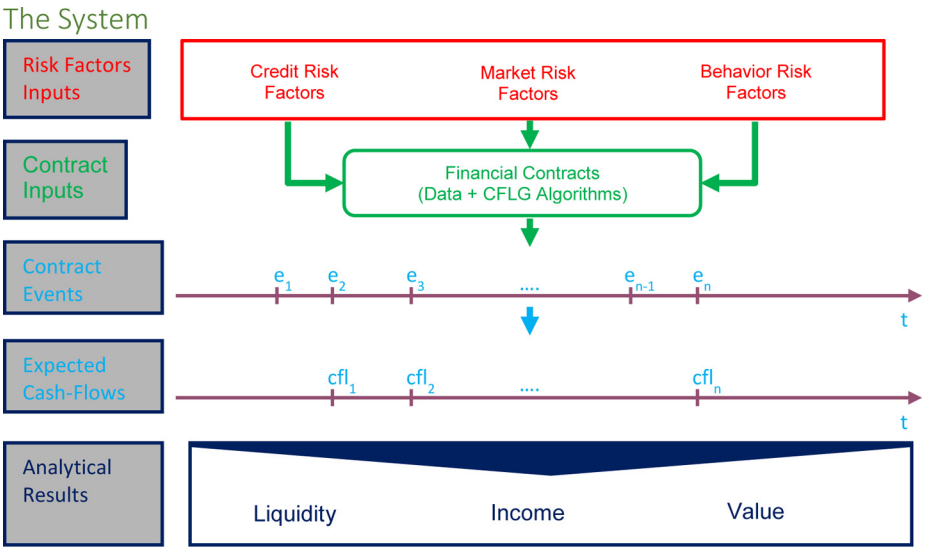

Figure 2.
The system

what we could call the primitive analytical results of finance, which are liquidity, income and value. The system provides time-dependent state variables for each of them.

What distinguishes the ACTUS approach is that ACTUS is the only standard under development that provides both the data (contract terms) and the algorithms needed to generate cash flow obligations. Furthermore, the standard is open source and freely available to any interested user, including practitioners, risk managers regulators and academics. The open source nature of the ACTUS algorithms will help insure that the software will be fully tested and deserving of complete confidence when used. In a world of smart contracts, "code is law". The most important lesson learned by the unfortunate experience of Ethereum with the DAO is that any smart contract used for any level of business has to be fully tested and deserving of the highest level of confidence in the accuracy of the code (Coppola, 2016). On the basis of the Ethereum experience, it is clear that bespoke contracts cannot meet this standard. Bespoke contracts actually perpetuate the current situation or make it even worse.

ACTUS proposes:

- build on the specific nature of financial contracts, which is logical/mathematical in its core;
- recognize that the observed patterns of cash-flow exchange are very limited for almost all financial contracts;
- build a fixed set of well-defined and tested **CFLG**^{i} functions that represent almost all financial contracts extant in the financial sector; and
- instead of offering a Turing complete language, as proposed within the Block-Chain movement, there is a need for APIs, where input and output is clearly defined and where the expected cash flows can be thoroughly tested.

The financial market needs a well-tested and understood framework that covers close to 100 per cent of the observed cases. This can be achieved with less than three dozen cash flow patterns.

Discovery, not an invention

In the nineteenth century, each producer of screws and bolts had its own metrics. Consequently, it was difficult to know how much pressure a screw and bolt assembly could hold. The problem that arose from this state of affairs is that steam engines could not be relied on to operate safely and not blow up with use. The solution was the first US engineering standard, which established the design and performance capabilities of nuts and bolts[7]. This standard was an invention that transformed an industry.

In banking, however, we see a different situation. Bankers have followed implicit standards for a very long time. If we go out into the world of finance and look at actual transactions, we find the same cash flow patterns again and again. However, these implicit standards were never turned into fully developed, transparent standards. Therefore, a standard such as ACTUS is not an invention but a discovery. ACTUS just represents these patterns in a clear and efficient manner in software.

Actually, more than 80 per cent of all deals done fit into a half a dozen patterns.

The existence of implicit standards can be illustrated with a simple hypothetical example. Let us assume that two bank enter a long-term contract for a loan. For one of the counterparties, it is an asset and for the other, it is a liability. The cash-outflow for one is a cash-inflow for the other. Other than the designation of the contract as an asset or a liability the contract is the same for both parties. Each bank then enters the contract into its transaction processing system, which almost certainly implies two different systems. The only information entered into the transaction processing systems are the terms of the contract such as principal, interest rate and maturity date. No coding is necessary to represent the contract because the required algorithms are already in the transaction processing system. With this information, the transaction processing systems are able to generate the cash flow obligations of each party.

Both systems represent the deal in their own way using different software, algorithmic logic and data. Both calculate the expected cash flows that need to be exchanged. The biggest difference between what is produced by the two systems is the sign of any particular cash flow.

If the two TPSs are entirely different systems, how can they get the same result? The answer rests with the implicit standards in finance. This means that there is sufficient knowledge in the industry to enable the two systems built by different programmers and implementing different instructions to come up with essentially the same answer. In German, there is a nice term for this: Usanz. This term of art might be translated into English as “the common knowledge of the guild”.

We have said that the two systems produce the same results, save for the sign. This is not 100 per cent correct. Actually, typically, they differ slightly. There is sufficient common knowledge in the industry to get almost the same result, but not exactly the same number. These differences are typically small and are usually amicably resolved. They rarely (if ever) are a reason to go to court. Actually, banks have tolerance limits for such differences. For example, one large bank used to have a tolerance limit on the difference of $50.00, which has more recently been reduced to $25.00.

Although transaction-processing differences are a big nuisance, these differences caused by the different algorithms are not the reason we are promoting a standard (although this alone might be a good enough reason). The more serious problem exists at the analytic level. This is the huge problem that needs to be solved. The ACTUS standard offers the solution, and the growing disruptive potential of Fintech could create sufficient incentive for legacy financial institutions to make the needed efforts to realize the benefits offered by this standard.

Implementation of a system

What about the implementation costs? If this is such a powerful solution, why has it not been implemented a long time ago? Could there be some barriers to implementation?

We will discuss the following five potential implementation barriers:

(1) technological challenge;
(2) adoption of a standard;
(3) universe of financial products;
(4) disruption of the existing (and functioning) production chain; and
(5) implementation costs.

Technological challenge

A faithful implementation of this approach requires that each financial contract is represented individually in the standard.

There could be as many as a few billion financial contracts extant in the world's financial markets. Each bank has between a few hundred thousand and few dozens of millions of financial contracts. Could this pose a problem?

As a system already exists on the TPS level (although non-standardized), we have proof that it works on that level. Performance measurements of the ACTUS standard are highly promising. They indicate that the performance is comparable to the current top performing risk systems. Within a single institution, it will not pose a bigger challenge than the current systems. Actually, the challenge should be seen as less daunting as the same results can be used in many different reports, which with current systems are produced in redundant parallel systems.

On the systemic level, the analytic challenge is bigger owing to the large number of financial contracts and the many scenarios (expected risk factors) that need to be calculated. Monte Carlo simulations, which produce many scenarios, pose an even greater challenge. However, other scientific endeavors face comparable challenges and have overcome them. For example, modern weather forecasting uses a complex algorithmic representation of the weather patterns for the entire globe, which is continuously fed with new data from a large number of observations taken on land, at sea, in the air and from satellites. High performance computing has been up to the task of running such large and complex systems. Similarly, the particle physics experiments at the Large Hadron Collider generate massive amounts of data and, to distribute it, use a significant portion of internet's bandwidth. More than a thousand particle physicists analyze the data.

When compared to alternative big science endeavors such as the Super Hadron Collider and modern weather forecasting, simulation of the financial sector does not pose a bigger or more challenging problem. Preliminary tests indicate that the modeling of the financial sector could be less demanding than weather forecasting. If we consider the financial sector at least as important as forecasting the weather, the required IT infrastructure is not an insurmountable hurtle.

Adoption of a standard

Establishing a standard is only the first step in realizing its benefits. Ultimately, widespread adoption and use of the standard is essential. While ACTUS is active in standard setting bodies, such as the RMG of ISO 20022, broad adoption must come from industry.

Some in the industry argue against such a standard because it enhances transparency: financial firms have profited from the opacity of financial products. An adoption of a

standard such as ACTUS has the potential to help level the playing field between the sell side and the buy side. We believe that support for maintaining opacity is waning. With the onslaught of Fintech, many bankers are weighting the costs of the absence of such a standard to be much higher than the advantages of opacity.

The fact that ACTUS did not invent such a standard but merely discovered it is another reason for financial firms to support its adoption. The standard replicates what banks are doing now with an excessively costly and imperfect implementation. Adoption of the standard will reduce the cost of operation on the both the TPS and analytic levels.

Creating a complete set of cash-flow generators.

Looking at the universe of financial products, the number of **CFLG^{i}** patterns appears, at first glance, as if it could be very large. However, when looking past the large number of different financial products created in response to marketing, legal and other considerations, the picture is very different. If we focus exclusively on the cash-flow exchange patterns, the number of required **CFLG^{i}**s shrinks dramatically.

As much as 80 per cent of all existing financial contracts might be represented by merely half a dozen patterns. Less than three dozen patterns or **CFLG^{i}**s are needed to represent virtually all-existing financial contracts, including instruments commonly referred to as exotic options (Figure 3).

Appendix explains the meanings of the abbreviated terms that ACTUS gives to the contract types referenced in the diagram above.

Only a small number of highly exotic financial contracts will not be included in the ACTUS standard. The solution for these marginal contracts is to represent them either by bespoke code or to approximate those using existing contract types. In the event that such highly exotic contracts become important in the market, the ACTUS standard will be extended to include coverage of them with new contract types.

How does ACTUS handle the perceived complexity of some financial instruments? The ACTUS standard differentiates between the deterministic and the stochastic. The ACTUS contract types represent the deterministic, clearly understood legal requirements of a

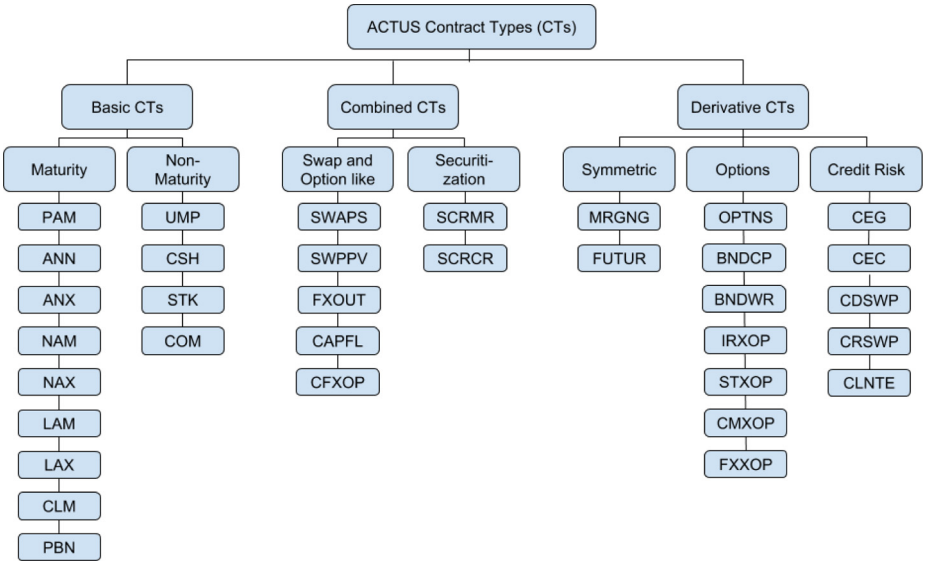

Figure 3.
Overview contract types

contract with respect to cash flows and payoff obligations. The calculation of required cash flows or payoffs is relatively straight forward, even for exotic instruments, if the state of the risk factors is known and the link between the risk factors and the contracts is defined. For example, calculating the payoff of an option is not complex: Pay the difference between the market price and the strike price on the exercise date.

Complexity enters the picture with the interaction between contractual obligations and risk factors, such as market variables. There are different levels of complexity associated with such interactions. For example, interest rates interact with contracts in a well-defined manner when calculating interest payments. However, the combination of behavior and market risks adds an additional level of complexity to other interactions. A standard such as ACTUS should provide the computational infrastructure for such calculations.

Finally, modeling of the risk factors themselves is also necessary for financial analysis. However, the inherently stochastic nature of risk factors does not allow standardization. Therefore, risk factors are beyond the boundaries of what is included in ACTUS. Modeling the risk factors is a task that is the responsibility of analysts, such as risk managers and researchers.

Disruption of the existing production line

The financial sector – banks, insurers, wealth managers and so on – are going concerns. Their TPSs are critical to smoothly running bank operations. They are, in effect, the factory floors of finance. They function well enough to make the bank management reluctant to consider attempting to replace them with new systems, even though they are not as good as they could be.

Consequently, it is not likely that the first implementation of the ACTUS standard will be in transaction processing, irrespective of how well it is developed and the level of future benefits that it promises.

At the AL, the problems are far greater than at the TPL. There is a minimally disruptive way of gaining the benefits for the AL made available by the use of a standard such as ACTUS. The solution is to focus the first implementation efforts on expanding the content and utility of the intermediate data stores or DW. We would give precise definitions to what is stored in the DW for each $\mathbf{CFLG}^{\mathbf{i}}$ using the data dictionary of the ACTUS standard. From a technological viewpoint, such a DW would look very similar to existing DWs. The critical difference is that the DW data would fully support the use of a **CFLG** standard in all applications. Data would be organized in the DW according to the published algorithmic standard, including thorough consistency checks. All analysis can be done using the contract terms in the DW and the algorithmic standard. This approach is non-disruptive and offers consistent standardized support for all analytic tasks.

Over time, a likely scenario is that the TPS will gravitate to using the $\mathbf{CFLG}^{\ddot{}}$s for transaction processing as it will significantly reduce maintenance costs and eliminate the need for costly reconciliation.

Once both TPSs and the AL adopt the standard, the distinction between the two levels would disappear. There would be a single system-wide representation of the contract-based exchange of cash-flows. The published algorithms and the contract attributes would constitute the contract and the same terms, and algorithms would be used for any analytics – only the expected risk factors will differ.

Cost of implementation

The main cost of such an exercise – possibly 80 per cent – is the data interface between the contract in the existing TPS and the intermediate data layer (DW).

This exercise is comparable to the introduction of an enterprise risk system or an asset and liability management system. Banks incur such costs and repeat such exercises at regular intervals. Even the introduction of a specific regulatory requirement such as comprehensive capital analysis and review might trigger a similar effort.

If a standard such as the one we propose is not adopted, banks will continue to incur such costs on regular intervals, resulting in high operating, regulatory compliance and risk management costs: a burden that will only continue to grow. The implementation of a standard such as our proposal would be the "last great mapping".

As a consequence, the cost of operations of financial institutions would drop dramatically, possibly to a sufficient level to enable legacy institutions to compete in the Fintech space.

The importance of a smart financial contract standard for Block-Chains.

Smart financial contracts are a necessary innovation to realize significant advances in the financial sector, independent of the technology applied. However, with respect to Block-Chain technology and other distributed ledgers implementations, a powerful standard such as ACTUS is even more important. Smart contracts will only be trusted if they rely on a smart contract standard that is openly published, fully understood and rigorously tested.

The alternative to a smart contract standard is a Turing complete language. However, as mentioned earlier, the freedom it offers comes with significant risk in a world in which code is law. Furthermore, the benefit of universal coverage offered by a Turing complete language is more an illusion than a reality. As mentioned above, as much as 80 per cent of all financial transactions can be represented by a mere half dozen standardized smart contracts. Furthermore, the universe of virtually all financial contracts extant in the market can be represented by less than three dozen smart contracts. In the event that a rare or particularly exotic instrument is not covered by the standard, nothing precludes the creation of a bespoke solution. However, for smart contracts to be widely used, they must be completely trusted by the counter-parties, a benefit not available for bespoke solutions.

Incorporating a smart contract standard, such as ACTUS, into the Block-Chain offers benefits on several levels. It would enhance the efficiency and quality of financial transactions, avoiding the pitfalls of the TPSs discussed above. In addition, it would enable a new level of market transparency based on the smart contracts' support for financial analysis. Each participant in the financial system would not only know precisely what was being transacted, but also how to analyze it. A smart contract is able to generate the state-contingent cash flow needed for the full range of analysis: risk, finance and regulation. Therefore, every interested party, whether a counter-party (such as a bank or investor) or a regulator, would have the same access to this critical input needed for financial analysis.

There are currently two approaches for using smart financial contracts with Block-Chain: either "on-chain" or "off-chain". On-chain approaches would record the actual smart contracts on the Block-Chain. Off-chain approaches would have the smart contracts reside alongside the Block-Chain, with only the payments generated by the smart contracts recorded on the chain. In theory, both approaches can work. However, specialists currently working on this still developing technology see the second approach as the preferred solution for efficiency reasons. Which way this will ultimately be resolved is still an open question.

The most relevant of the CTs have been implemented and are available.

Notes

1. The research recognized with the Noble Memorial Prize centers on the importance of contracts to the workings of an economy. It includes a wide range of issues that go to the ability of contracts to contribute to the realization of desired outcomes, including incentives and performance; agency; incomplete contracts; risk/return tradeoffs; implications for ownership structures; obstacles to cooperation and how to overcome them; and others. This body of work does not include the specifics of financial contracts discussed in this paper. For a summary of this body of work, see "Scientific Background on the Sveriges Riksbank Prize in Economics Science in Memory of Alfred Nobel 2016. OLIVER HART AND BENGT HOLMSTROM: Contract Theory". The Committee for the Prize in Economic Sciences in Memory of Alfred Nobel, October 10, 2016
2. The origin of the term "smart contract" goes back to the mid-1990s and is credited by multiple sources to Nick Szabo, a computer scientist and cryptographer. He defines a smart contract as "A computerized transaction protocol that executes the terms of a contract".
3. For a complete discussion of this topic, see (Brammertz *et al.*, 2009).
4. BCP 239, for example, demands consistent risk aggregation. We argue that consistent aggregation demands a standard, as proposed here.
5. As a consequence of the 2008 financial crisis, the number of active cash-flow patterns in the market has dropped significantly. Exotic options have practically disappeared.
6. www.ACTUSfrf.org
7. In 1864, William Sellers, an engineer, inventor and manufacturer proposed the adoption of a standard for nuts and bolts, which ultimately became the "American Standard Threads". See (Sellers, 1864).

References

Brammertz, W., Akkizidis, I., Breymann, W., Entin, R. and Rustomann, M. (2009), *Unified Financial Analysis: The Missing Links of Finance*, John Wiley & Sons.

Coppola, F. (2016), "A painful lesson for the ethereum community", *Forbes.com*, 21 July 2016.

Sellers, W. (1864), "A system of screw threads and nuts", *Journal of the Franklin Institute*, Vol. 47, p. 344.

Corresponding author
Willi Brammertz can be contacted at: willi.brammertz@ariadne.swiss

Appendix

Family	Class	Type	Description	Covered real-world instruments
Basic CT	Maturity	PAM: Principal at Maturity	Principal payment fully at Initial Exchange Date (IED) and repaid at Maturity Date (MD). Fixed and variable rates.	All kind of bonds, term deposits, bullet loans and mortgages etc.
		ANN: Annuity	Principal payment fully at IED and interest plus principal repaid periodically in constant amounts till MD. If variable rate, total amount for interest and principal is recalculated to be fully matured at MD.	Classical level payment mortgages, leasing contracts etc.
		NAM: Negative Amortizer	Similar as ANN. However when resetting rate, total amount (interest plus principal) stay constant. MD shifts. Only variable rates.	Special class of ARMÂ´s (adjustable rate mortgages), Certain loans.
		LAM: Linear Amortizer	Principal payment fully at IED. Principal repaid periodically in constant amounts till MD. Interest gets reduced accordingly. If variable rate, only interest payment is recalculated. Fixed and variable rates.	Many amortizing loans
		ANX: Exotic Annuity	Exotic version of ANN However step ups with respect to (i) Principal, (ii) Interest rates are possible. Highly flexible to match totally irregular principal payments. Principal can also be paid out in steps.	A special version of this kind are teaser rate loans and mortgages with annuity features
		LAX: Exotic Linear Amortizer	Exotic version of LAM. However step ups with respect to (i) Principal, (ii) Interest rates are possible. Highly flexible to match totally irregular principal payments. Principal can also be paid out in steps.	A special version of this kind are teaser rate loans and mortgages
		NAX: Exotic Negative Amortizer	Exotic version of NAM However step ups with respect to (i) Principal, (ii) Interest rates are possible. Highly flexible to match totally irregular principal	A special version of this kind are teaser rate loans and mortgages with variable MD
			payments. Principal can also be paid out in steps.	
		CLM: Call Money	Loans that are rolled over as long as they are not called. Once called it has to be paid back after the stipulated notice period.	Interbank loans with call features
		PBN: Perpetual Bonds	Bonds without any maturity date. Interest is paid into eternity if is not terminated.	Consoles, war loans
	Non-Maturity	UMP: Undefined Maturity Profile	Principal paid in and out at any point in time without prefixed schedule. Interest calculated on outstanding and capitalized periodically. Needs link to a behavioral function describing expected flows.	Saving products of all kind, current accounts. In some countries even variable rate mortgages can be represented with this CT
		CSH: Cash	Cash or cash equivalent position	Cash, deposits at central bank
		STK: Stock	Any instrument which is bought at a certain amount (market price normally) and then follows an index.	All straight stocks
		COM: Commodity	This is not a financial contract in its proper sense. However it tracks movements of commodities such as oil, gas or even houses. Such commodities can serve as underlyings of commodity futures, guarantees or simply asset positions.	Oil, gas, electricity, houses etc.
Combined	Swap and Option like	SWAPS: Swap	Exchange of two basic CTÂ´s (PAM, ANN etc.). Normally one is fixed, the other variable. However all variants possible including different currencies for cross currency swaps, basic swaps or even different principal exchange programs.	All kind of swaps. The variety is defined by the underlying CTÂ´s which currently are PAM and ANN in all tis flavors. With each new basic CT the variety rises
		SWPPV: Plain Vanilla Swap	Plain vanilla swaps where the underlying is always a PAM and one leg is fixed, the other variable. Plain vanilla cross currency swaps also covered.	More than 90% of all interest rate swaps follow this simple pattern.
		FXOUT: Foreign Ex-	Two parties agree to exchange two fixed cash flows in different	Any FX-outright transaction. This is also

(*continued*)

		change Outright	currencies at a certain point in time in future.	the underlying of FX-options and FX futures
		CAPFL: Cap Floors	Interest rate option expressed in a maximum or minimum interest rate	Caps and Floor options
		CFXOP: Exotic Cap Floor	Exotic variants of caps and floors	
	Securitization	SCRMR: Securitized Instruments Market Risk	Instruments bundled and traded in tranches without any specific credit risk feature	MBS, ABS, Principal only, Interest only instruments
		SCRCR: Securitized instrument Credit Risk Feature	Instruments bundled and traded in tranches that include specific credit risk feature	CDOÂ´s
Derivatives	Symmetric	MRGNG	A generic margining contract governing the agreement of margining usually present at central clearing houses	
		FUTUR: Future	Keeps track of value changes for any basic CT as underlying (PAM, ANN etc. but also FXOUT, STK, COM). Handles margining calls.	Standard interest rate, FX, stock and commodity futures.
	Options	OPTNS: Option	Calculates straight option pay-off for any basic CT as underlying (PAM, ANN etc.) but also SWAPS, FXOUT, STK and COM. Single, periodic and continuous strike is supported.	European, American and Bermudan options with Interest rate, FX and stock futures as underlying instruments
		BNDCP: Callable or puttable maturity contract	Bonds with a call or put option. If option is exercised, underlying bond ceases to exist.	Callable and puttable bonds or loans
		BNDWR: Bond with warrant	Bonds with a warrant. If option is exercised, underlying bond continues to exist.	Warrants
		IRXOP: Exotic Interest Rate Option	Exotic interest rate options	
		STXOP: Stock Option	Exotic stock options	
		CMXOP: Exotic Commodity Option	Exotic commodity options	
		FXXOP: Exotic FX Option	Exotic FX options	
	Credit Risk	CEG: Guarantees	Guarantee is a credit enhancement contract. It creates a relationship between a guarantor, an obligee and a debtor, moving the exposure from the debtor to the guarantor.	Personal guarantee. Government guarantee. Underlyings of CDOÂ´s.
		CEC: Collateral	Collateral is a credit enhancement contract. It creates a relationship between a collateral an obligee and a debtor, covering the exposure from the debtor with the collateral.	Mortgages include a collateral contract. Any coverage with financial or physical collateral
		CDSWP: Credit Default Swap	All sorts of credit default swaps	
		CRSWP: Total Return Swap	All sorts of total return swaps	
		CLNTE: Credit Linked Note	All sorts of credit linked notes	