OFR WORKING PAPER
OFFICE OF FINANCIAL RESEARCH

15-04 | March 26, 2015
Revised March 27, 2017

Contract as Automaton: The Computational Representation of Financial Agreements

Mark D. Flood
Office of Financial Research
mark.flood@ofr.treasury.gov

Oliver R. Goodenough
Office of Financial Research and Vermont Law School
oliver.goodenough@ofr.treasury.gov
ogoodenough@vermontlaw.edu

Contract as Automaton:

The Computational Representation of Financial Agreements

By Mark D. Flood † and Oliver R. Goodenough‡

March 27, 2017

JEL codes: D86, K12, C63

Keywords: Financial contracting, state-transition system, deterministic finite automaton, theory of computation, contractual complexity

Key Messages:

- Financial contracts are structured internally as state-transition systems
- Discrete finite automata (DFA) are an adequate formalism to represent this structure as finite sets of states, events, and transitions
- DFA representations readily allow for testing of contractual completeness and complexity

Acknowledgments

The authors thank Dan Katz, Jeanne Eicks, Matt Reed, Greg Feldberg, Primavera de Filippi, David Blaszkowsky, Sean Byrne, Harold Boley, Sanjoy Mitter, Andrew Lo, and Lynn Stout. Other helpful comments and background discussions on computation law came from research seminar participants at Cambridge University, Dartmouth College, George Mason University, Harvard University's Berkman Center for Internet and Society, Massachusetts Institute of Technology, Office of Financial Research, Ewing Marion Kauffman Foundation, Gruter Institute, Stanford CodeX Center for Legal Informatics, IBM Research at Almaden, Institute for Pure and Applied Mathematics at the University of California at Los Angeles, Vermont Law School, and American Association of Law Schools' annual meeting. We also thank John Donnelly, Jaryn Fields, Catherine Bourque, and Jacinta Ritchie for able support on this and related research. All remaining errors are the responsibility of the authors alone.

† Office of Financial Research, mark.flood@ofr.treasury.gov

‡ Vermont Law School and Office of Financial Research, oliver.goodenough@ofr.treasury.gov or ogoodenough@vermontlaw.edu

Contract as Automaton:

The Computational Representation of Financial Agreements

Abstract

We show that the fundamental legal structure of a well-written financial contract follows a state-transition logic that can be formalized mathematically as a finite-state machine (specifically, a deterministic finite automaton or DFA). The automaton defines the states that a financial relationship can be in, such as "default," "delinquency," "performing," etc., and it defines an "alphabet" of events that can trigger state transitions, such as "payment arrives," "due date passes," etc. The core of a contract describes the rules by which different sequences of events trigger particular sequences of state transitions in the relationship between the counterparties. By conceptualizing and representing the legal structure of a contract in this way, we expose it to a range of powerful tools and results from the theory of computation. These allow, for example, automated reasoning to determine whether a contract is internally coherent and whether it is complete relative to a particular event alphabet. We illustrate the process by representing a simple loan agreement as an automaton.

I. Introduction

A crucial step in understanding financial networks is comprehending the edges that link vertices together. For many financial networks, these edges correspond to financial contracts in the real world, implying a need for a concise, structured representation of these legal agreements that define formally the legally enforceable relationships connecting financial actors. In this paper, we propose a general framework for such a formal representation, grounded in both legal and computational principles. In addition to defining the legally enforceable relationships that connect financial actors, contracts have increased in importance over the past half-century. An increasing proportion of overall financial activity has migrated to arms-length contracts, exemplified by the growth in derivatives markets, securitization, and high-frequency trading; see Greenwood and Scharfstein (2013), Clark (2011), and Flood, Mendelowitz, and Nichols (2013). This move away from the traditional intermediation provided by banks presents researchers and policymakers with new challenges in the form of liquidity surprises and fire sales, and new opportunities in the form of detailed datasets describing the networked interactions.

A well-written financial contract is a finite-state machine in a formal sense. We demonstrate this proposition by representing a simple fixed-rate loan agreement using a standard computational formalism.[1] The structure of a contract employs legalese to encode a finite set of *states* that can describe the relationship between the counterparties at a given point in the life of the contract. The contract also encodes *transition* rules for shifting the relationship from one state to another based on the realization of certain predefined *events*, such as performance by the counterparties themselves or the occurrence of particular contingencies outside their control.

By formalizing financial agreements according to rules of computability, we expose the contracts to a wealth of powerful computational machinery. This machinery includes programmatic testing for (legal) completeness and (computational) complexity, and tools for simplification, visualization, and even the automated generation of legalese. Conversely, forcing a contract to adhere to the prerequisites of the state machine model — especially the finiteness of states and events, and the independence of states from one another — imposes valuable normative discipline for *how* contracts should be crafted.

Our approach follows Allen (1957) and von der Lieth Gardner (1987), who similarly represent legal constructions with symbolic formalisms. Allen (1957) applies symbolic logic to legal documents generally. Although symbolic logic is more expressive and more general than automata, it is also a lower-level abstraction and inefficient for capturing the legal semantics of financial contracts. Von der Lieth Gardner (1987) considers computational representations of legal knowledge and reasoning generally. She considers an ambitious range of formalisms, including the multiple representation system (MRS) of Genesereth, Greiner, and Smith (1981), and augmented-transition networks once recommended for natural language parsing.

[1] There are many excellent textbook treatments of the theory of computation and finite state machines, including Sipser (2006), Kozen (1997), and Rosenberg (2010). For an overview, see the lectures by Simonson (2013).

Closer to implementation, Grosof and Poon (2004) prototype an e-contract system called SweetDeal, based on a RuleML encoding of the knowledge representation, and a description logic representation of process ontologies. This is a relatively early example of the growing literature on computational contracting; see Surden (2012) for an overview from a legal perspective. Brammertz, et al. (2009) develop a structured formalism for financial contracts that focuses on a description of common intertemporal cash flow patterns to support securities valuation and risk analysis. This is the foundation for the Project ACTUS (2015) implementation pilot. Another recent pilot is the Financial Industry Business Ontology (FIBO), which proposes a formal, standardized model of legal concepts in the finance; see OMG (2014). Still closer to implementation, the automation of accounting, valuation, risk analysis, and trade execution is a practical necessity in the daily operations of trading firms; see Brose, et al. (2014a, 2014b).

Any representation has strengths and emphases. Our formalism is agnostic about the details of implementation, but adheres closely to the fundamental structure and legal semantics of the financial agreement. By making this initial transition from traditional legalese to a computational representation as accurate and lossless as possible, subsequent translations to implementation representations should be easier and more useful. By specifying this foundational representation as a deterministic finite automaton (DFA), we expose the legal structure to a wealth of computational theory.

II. Legal Structure of Financial Contracts

Contracts are central to the economic life of markets. They define detailed relationships between counterparties, itemizing what each promises to do, or not do, under a pre-specified list of key contingencies.

It is convenient (but not necessary) to think of a financial contract as combining a statement of the desired sequence of events and actions with statements of the commitments and legal remedies to be pursued if circumstances should go awry. We refer to the desired execution sequence under a contractual relationship as the "happy path."[2] In our example of a simple loan agreement in Table 1, the sections denoted "Counterparties" and "Basic obligations" compose the happy path. The sections denoted "Default provisions" and "Enforcement" cover the various unhappy paths that the relationship might follow. Taken together, the elements of the loan in Table 1 can create significant complexity by linking a number of variables and factors together in chains of events and consequences, and by delineating varied outcomes depending on the factors involved. For instance, an agreement might make a distinction between, and treat differently, (i) a default under an obligation to make a payment and (ii) a default under a covenant (such as a promise to provide a particular financial report), each playing out differently across the logic of the contract. Complexity almost always increases when a transaction leaves the happy path of expected

[2] The notion of the "happy path" comes from the world of software testing, where it refers to the primary software execution path that provides the core required functionality and does not provide for error handling or other exceptional events. In other words, the happy path is the sequence executed when everything goes as expected (Software Engineering Institute, 2009).

and timely fulfillment and enters the world of default, penalty, and enforcement, a world we will discuss at greater length in Section V.

Because a contractual relationship typically envisions a single happy path and many unhappy paths, much of the effort in drafting agreements must address the latter. In contracting, as in software development, planning for exceptional situations and component failures are crucial for a robust process. A powerful characteristic of the deterministic finite automata that we describe later in this paper is that DFAs rigorously enforce a simplicity requirement that the history of a process, such as the evolution of a contractual relationship, must be fully captured by the specification of its current state. We argue that well-drafted financial contracts should and do adhere to this rule. This discipline helps keep the overall complexity of the system within manageable bounds.

III. A Computational Representation of a Financial Contract

Using an example, we demonstrate in this section that a well-designed financial contract is a state transition system. Moreover, we can structure such a financial contract as a particular sort of state transition system, namely a deterministic finite automaton.

The deterministic finite automaton or DFA structure of a financial contract is not metaphorical. A DFA is a mathematical formalism with precise requirements, which a contract may or may not meet. We argue that a well-designed financial contract satisfies these requirements, and thus gains access to the substantial body of formal results from the theory of computation. Conversely, a given financial contract observed in the field may fall short of the requirements of the DFA formalism. Such deviations from the ideal represent design compromises, and the distance between the actual and the ideal in any given case might be a measure of the quality of the legal craftsmanship involved.

The subsequent discussion of an example of a streamlined loan agreement simplifies two things. First, our example is condensed to avoid a surfeit of contractual provisions beyond the basics needed to illustrate our core argument and demonstrate the feasibility of the approach. Fully formed, practical examples of financial transactions, such as the swaps master agreement (ISDA, 2002) or the foreign exchange master agreement (Foreign Exchange Committee, 1997) are crowded with particulars that compound the modeling burden while distracting from the central concepts. We leave these more ambitious mappings for future research.

Second, and more fundamentally, we have chosen the most basic computational formalism — the deterministic finite automaton or DFA — to represent the structure of our contract. Computation theory has a set of different computation models, appropriate for different sorts of tasks, such as representing system behavior or parsing source code that conforms to a specific grammar. These models differ in their expressiveness, meaning the degree of intricacy of the structure that they can capture. The fact that the DFA is sufficiently expressive to represent our streamlined contract suggests that contract law and drafting have evolved to embody computational logic at this relatively simple level of sophistication for managing

Table 1: A Streamlined Loan Agreement

Agreement

This loan agreement dated June 1, 2014, by and between Lender Bank Co. ("Lender") and Borrower Corp. (Borrower), will set out the terms under which Lender will extend credit in the principal amount of $1,000 to Borrower with an un-compounded interest rate of 5% per annum, included in the specified payment structure.

1. The Loan

At the request of Borrower, to be given on June 1, 2014, Lender will advance $1,000 to Borrower no later than June 2, 2014. If Borrower does not make such a request, this agreement will terminate.

2. Repayment

Subject to the other terms of this agreement, Borrower will repay the loan in the following payments:

(a) Payment 1, due June 1, 2015, in the amount of $550, representing a payment of $500 as half of the principal and interest in the amount of $50.

(b) Payment 2, due June 1, 2016, in the amount of $525, representing a payment of $500 as the remaining half of the principal and interest in the amount of $25.

3. Representations and Warranties

The Borrower represents and warrants, at the execution of this agreement, at the request for the advance of funds and at all times any repayment amount shall be outstanding, the Borrower's assets shall exceed its liabilities as determined under an application of the FASB rules of accounting.

4. Covenants:

The Borrower covenants that at the execution of this agreement, at the request for the advance of funds and at all times any repayment amount shall be outstanding it will make timely payment of all state and federal taxes as and when due.

5. Events of Default

The Borrower will be in default under this agreement upon the occurrence of any of the following events or conditions, provided they shall remain uncured within a period of two days after notice is given to Borrower by Lender of their occurrence (such an uncured event an "Event of Default"):

(a) Borrower shall fail to make timely payment of any amount due to Lender hereunder;

(b) Any of the representation or warranties of Borrower under this agreement shall prove untrue;

(c) Borrower shall fail to perform any of its covenants under this agreement;

(d) Borrower shall file for bankruptcy or insolvency under any applicable federal or state law.

A default will be cured by the Borrower (i) remedying the potential event of default and (ii) giving effective notice of such remedy to the Lender. In the event of multiple events of default,

the first to occur shall take precedence for the purposes of specifying outcomes under this agreement.

6. Acceleration on Default

Upon the occurrence of an Event of Default all outstanding payments under this agreement will become immediately due and payable, including both principal and interest amounts, without further notice, presentment, or demand to the Borrower.

7. Choice of Law

This agreement will be subject to the laws of the State of New York applicable to contracts entered into and performed wholly within that state.

8. Amendments and Waivers

Any purported amendment to, or waiver of rights under, this agreement will only be effective if set forth in writing and executed by both parties.

9. Courts and Litigation

Any legal action brought to enforce, interpret or otherwise deal with this agreement must be brought in the state courts of the State of New York located in New York County, and each of the parties agrees to the jurisdiction of such courts over both the parties themselves and over the subject matter of such a proceeding, and waives any claim that such a court may be an inconvenient forum.

10. Time of the Essence; No Pre-Payment

Timely performance is required for any action to be taken under this agreement, and, except as may otherwise be specifically provided herein, failure to take such action on the day specified will constitute a binding failure to take such action. Payments shall only be made on or after the dates specified in Section 2 or on or after such other date as may be required under Section 6; pre-payments made on earlier dates shall not be accepted.

11. Notices

Notices provided for in this agreement will be given by an email to the email addresses set out below and will be effective upon receipt.

[Lender email here] [Borrower email here]

Accepted and agreed:

LENDER BANK CO. BORROWER CORP.

By: _____ By: _____

Title: _____ Title: _____

[NOTE: Statute of Limitations on debt obligations in New York is 6 years] Draft of July 23, 2014

Source: Authors' analysis

actual relationships.[3] The agreed relationship must ultimately be interpreted by the counterparties, and potentially by the courts or other arbitrators, in a broader context that may have changed substantially over the course of the relationship. Clarity and simplicity of the contract are important virtues; ambiguity and unnecessarily complex sophistication are not. We return to this fundamental point in the conclusion of this paper.

The Formalism and the Contract

We propose to represent a financial contract as a deterministic finite automaton. This is an exercise in parallel specification. The first specification is the streamlined contract, embodied in legalese, shown in **Table 1**. The second specification is a DFA representation, in the tables and figure below, of the same structure of rights, obligations, actions, and contingencies.

Formally, using Sipser's (2006) notation, the DFA specifies a computation as a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$:

1. A finite set of states, denoted Q
2. A finite set of input symbols (events) called the alphabet (Σ)
3. A transition function ($\delta : Q \times \Sigma \rightarrow Q$)
4. A start state ($q_0 \in Q$)
5. A set of accept (end) states ($F \subseteq Q$)

The contract defines a finite, mutually exclusive, and exhaustive collection of states that describe the possible conditions of the relationship between the counterparties. At any point in the life of the agreement, the relationship is in exactly one state, $q \in Q$. Which particular state is operative at a given point will depend on what has occurred, i.e., the sequence of observed events, $e_1, e_2, \ldots \in \Sigma$ up to that point. Organizing a contractual relationship around discrete states, such as "performing," "cancelled," or "defaulted," is what good drafters do, perhaps subconsciously.

The alphabet represents the discrete set of inputs (Σ) that the agreement recognizes. These might correspond to information events or actions taken by the counterparties. A contract makes positive commitments to address those events that rise to the threshold of relevance for the relationship. Our streamlined loan agreement, for example, does not countenance the outcome of the World Series or fluctuations in the spot price of crude oil. Moreover, the contract simplifies matters by discretizing the infinite gradations of possibility that typically describe the real world into a finite set of measurable events. For example, changes in creditworthiness under the International Swaps and Derivatives Association's (ISDA) master agreement are mapped into a set of discrete credit events, such as "failure to pay" or "repudiation," as decided by the appropriate ISDA determination committee. The finiteness of the event

[3] There are several alternative representations with expressiveness equivalent to the deterministic finite automaton (DFA). We discuss tabular, graphical, and regular expression presentations below. In addition, any DFA can be converted to an equivalent nondeterministic finite automaton (NFA), and vice versa, as a programmatic exercise. The NFA representation adds a layer of abstraction, but is typically more compact than the corresponding DFA; see Sipser (2006), section 1.2. More expressive computational models include pushdown automata (Sipser, 2006, 109ff) and Turing machines (Sipser, 2006, 137ff).

space constrains the possible complexity of the agreement. Formalizing these inputs as the alphabet Σ helps clarify these principles.

The transition function describes the change in state of the relationship in response to the arrival of particular events. That is, with the relationship starting in state $q_1 \in Q$, the transition function defines a mapping $q_1 \times e \rightarrow q_2$ that describes that the state of the relationship should change from state q_1 to q_2 in response to the arrival of event $e \in \Sigma$. In principle, any combination of initial states and observed events is conceivable, but in practice, an agreement will simply ignore some events in certain states, so that the relationship remains in the initial state: $q_1 \times e \rightarrow q_1$. For example, if the contract is in the "cancelled" state, then the occurrence of a "declaration of bankruptcy" event will not change the state; the agreement remains cancelled: $q_{\text{cancelled}} \times e_{\text{bankruptcy}} \rightarrow q_{\text{cancelled}}$.

This process of event and transition goes forward until the automaton reaches an "accept" state. The start state, q_0, is the spot where the relationship begins, and an accept state, $q_T \in F$, is a spot where, if reached, the process ends. The sets Q and Σ are both finite, and the transition function is deterministic, with exactly one response (possibly "state unchanged") to a given event. We assert that financial contracts should follow a similar kind of step-by-step logic, matching information against a current state of facts and contract execution specifications to lead to a next state. Normatively, natural language financial contracts should be crafted as computational automata in this sense. That is, contracts are the legal machinery for describing how a relationship will progress in response to key events, and should adhere to the prerequisites of finite automata. This makes available a range of tools and conclusions from computation theory.

IV. Representations of the Contract as a DFA

We illustrate this proposition explicitly in the terms of a DFA with the streamlined loan agreement set out in Table 1. The streamlined agreement specifies one loan and only two repayments. The interest is a simple percentage (5 percent, noncompounding), specified as explicit dollar repayments ($525 and $550) to avoid the need for an ancillary interest rate calculation. There is one warranty, one covenant, and one other event unrelated to payments that can trigger default. The default process shares a set of time frames and notice specifications, and acceleration, if triggered, is still for the sum certain of the outstanding payments, without additional penalty. The DFA implicitly presumes that default events cannot occur simultaneously.[4] We have not provided for collateral or a guarantee.

There are at least three standard forms for representing a DFA: (1) a visual plot of the state-transition network, (2) a tabular (or matrix) listing of the elements of the 5-tuple, and (3) a regular expression. These are formally equivalent, and there are standard procedures for translating without loss of information among the three.

[4] This is a cosmetic simplification. One could add states to the deterministic finite automaton (DFA) to handle "product events" representing the simultaneous occurrence of two or more default triggers. This would proliferate states in the model, but without adding sophistication beyond that expressible by a DFA.

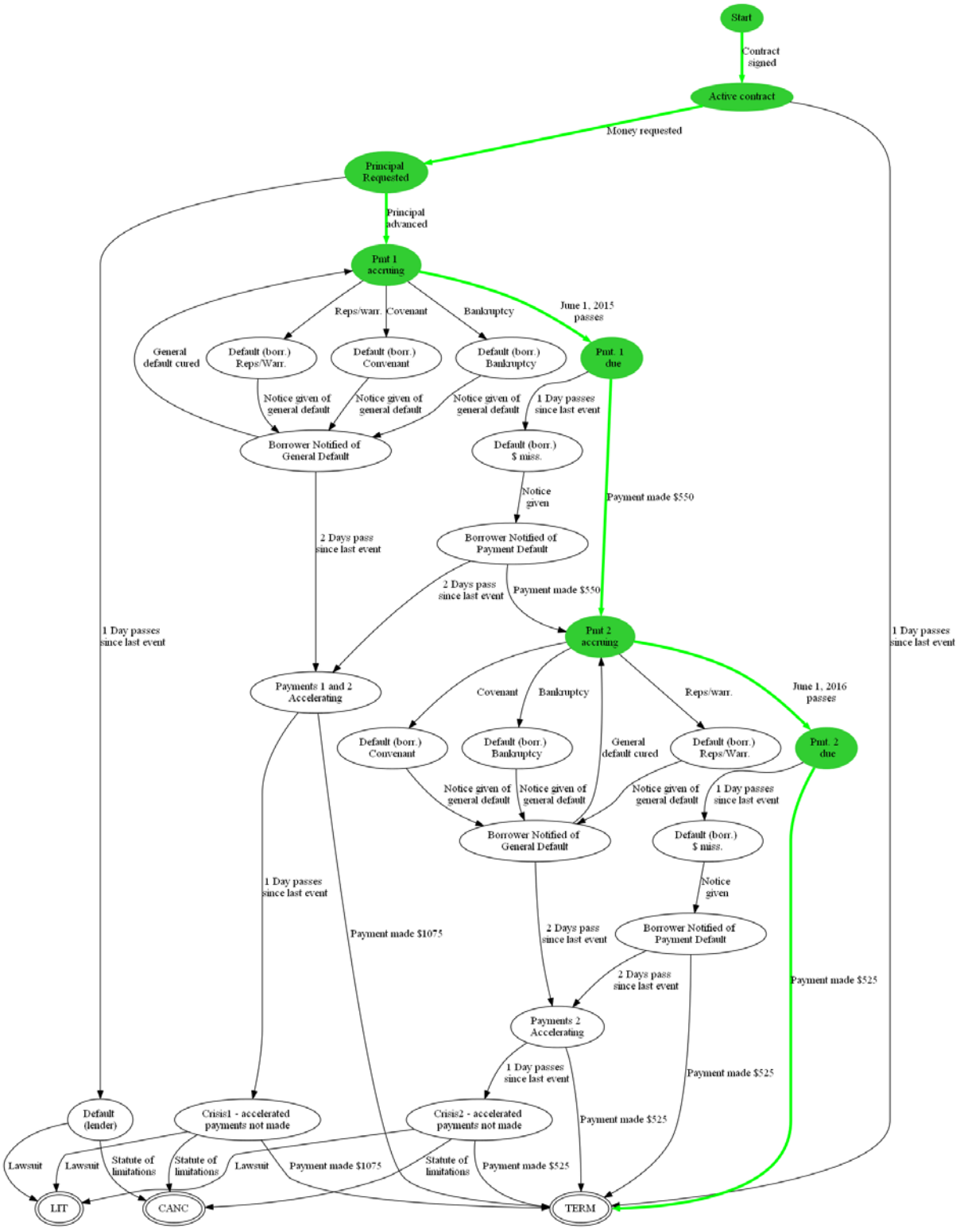

Figure 1: Graphical Representation of the Deterministic Finite Automaton (DFA) for the Streamlined Contract

Source: Authors' analysis

Graphical Representation

We present a graphical representation of the state transition network for our streamlined loan agreement in **Figure 1**. The nodes represent the possible states in the DFA. The relationship must exist in exactly one state at each point in time. The labels on the arrows describe elements from the event alphabet that trigger transitions from one state to another. The arrows themselves show the effect of the transition function applied to the state and alphabet elements. The start state appears at the top, and the three accept states of contract fulfillment, cancellation, and litigation appear with double borders at the bottom. (Cancellation includes the case of creating a new contract through waiver or amendment.) The green nodes, together with the sequence of green arrows that link them, indicate the happy path through the relationship.

The representation in Figure 1 is a partial simplification, because it suppresses many transitions in the interest of clarity. In most cases, an event leaves the state unchanged; the event is irrelevant in the context of those states. To be complete, the graph should include transitions for all such events, looping back to the same state. In addition, some events, such as those triggering litigation or cancellation, are relevant in every state, but always transition to the same terminal state. We have omitted these repetitive arrows from Figure 1 to avoid cluttering the visualization. These repetitive transitions do, however, appear in the tabular presentation of the DFA to follow. Note, too, that an input event that represents an agreed waiver or amendment will necessarily result in a new DFA representation with a different set of elements. In the context of the current DFA, these events provoke a transition directly to the terminal cancellation state.

Tabular Representation

The next representation is a tabular specification of the elements in the 5-tuple: $\{Q, \Sigma, \delta, q_0, F\}$. For each state $q \in Q$, most events leave the state unchanged — the event input is irrelevant in that state and no transition occurs. Also, for any state, the input $T \in \Sigma$ (the final row of Table 3) representing a waiver, amendment, or agreed termination of the contract moves us directly to an accept state terminating *this* agreement/DFA. In the case of a waiver or amendment, the ultimate result is a new DFA with a different set of elements. As a convenience, **Table 2** and **Table 3** also cross-reference the section in the natural language most relevant to the state or event.

The tabular presentation emphasizes an important point: The two spaces of states (Q) and events (Σ) are both simple sets. That is, there is no ordering or ranking among the elements of either set; any reshuffling of the rows of Table 2 or Table 3 would not affect the DFA. The only special status is given to start state, q_0, and the accept states, F, declared as separate members of the 5-tuple. The subtle implication of this lack of ordering among the elements of Q and Σ is that the states of the DFA are "memoryless." In technical

Table 2: Contract States (Q)

State	Label	Natural Language Consequences and Correlates (Λ)	Sec
start ☺	Start	Contract is fully specified; key information (payment dates, notice addresses and procedures, choice of law, and dispute process) delivered	7, 9, 11
q0 ☺	Active contract	Contract is fully signed/executed	
q1 ☺	Principal requested	Borrower has requested and awaits $1,000	1
P1 ☺	Payment 1 accruing		
P1d ☺	Payment 1 due		
P2 ☺	Payment 2 accruing		
P2d ☺	Payment 2 due		
Dl	Default (lender)	Lender has failed to deliver principal	5
Acc1	Payments 1 and 2 accelerating	Accelerated payment due is $1,075	6
Acc2	Payments 2 accelerating	Accelerated payment due is $525	6
Db0_1	Default (borrower) payment missed	Borrower has failed to make first payment on time and should be notified	5
Dbcv_1	Default (borrower) covenant	Borrower violates covenant(s) and should be notified	5
Dbrw_1	Default (borrower) representations/warranties.	Borrower breaches representations or warranties and should be notified	5
Dbbkr_1	Default (borrower) bankruptcy	Borrower files for bankruptcy or insolvency and should be notified	5
Nb0_1 Δ	Borrower notified of payment default	Borrower has two days to pay, or all payments accelerate	5
Nbnpd_1 Δ	Borrower notified of general default	Borrower has two days to pay, or all payments accelerate	5
Db0_2	Default (borrower) payment missed	Borrower has failed to make first payment on time and should be notified	5
Dbcv_2	Default (borrower) covenant	Borrower violates covenant(s) and should be notified	5
Dbrw_2	Default (borrower) representations/warranties	Borrower breaches representations or warranties and should be notified	5
Dbbkr_2	Default (borrower) bankruptcy	Borrower files for bankruptcy or insolvency and should be notified	5
Nb0_2 Δ	Borrower notified of payment default	Borrower has two days to pay or all payments accelerate	5
Nbnpd_2 Δ	Borrower notified of general default	Borrower has two days to pay or all payments accelerate	5
xT †	TERM	Contract is fulfilled in accordance with its terms	
xL †	LIT	A legal action is brought to enforce, interpret, or otherwise deal with the agreement in the state courts of the state of New York located in New York County that the results of this action will replace the computation of the contract	9
xC †	CANC	Contract is canceled due to the passing of time beyond the statute of limitations or canceled because of modification or termination by mutual agreement of the parties	8
Crisis1	Crisis1 — accelerated payments not made	Payments accelerated, but borrower has not responded	6
Crisis2	Crisis2 — accelerated payments not made	Payments accelerated, but borrower has not responded	6

☺ *States on the "happy" path of the contract lifecycle*
Δ *Default states*
† *Terminal states*

Source: Authors' analysis

Table 3: Event Alphabet (Σ)

ID	Label	Natural Language Event Specification	Section
A	Contract signed	Contract is signed to bind all parties	
B	1 Day passes since last event	June 1, 2014, passes	1
C	Money requested	Borrower gives request for loan of $1,000	1
D	Lawsuit	A legal action is brought to enforce, interpret, or otherwise deal with the agreement in the state courts of the state of New York located in New York County.	
E	Statute of limitations	June 1, 2020, passes — the Statute of Limitations on debt obligations in New York is six years	
F	Principal advanced	Lender advances $1,000 no later than June 2, 2014	1
G	June 1, 2015, passes	Payment 1 due on June 1, 2015	2(a)
H	Representations/warranties	Borrower's assets exceed its liabilities as determined under an application of the FASB rules of accounting	3, 5(b)
I	Covenant	Borrower fails to make a timely payment of an amount of state or federal tax	4, 5(c)
J	Bankruptcy	Borrower files for bankruptcy or insolvency under any applicable federal or state law	
K	Notice given	Notice given to borrower of a failure to make timely payment of an amount due to lender under this agreement	5
L	Notice given of general default	Notice given to borrower of an event of default other than a failure to make timely payment of an amount due	5
M	Payment default cured	A payment-related event of default is cured	5
N	General default cured	A nonpayment-related event of default is cured	5
O	2 days pass since last event	Two days elapse since last event/notice	5
P	June 1, 2016, passes	Payment 2 is due on June 1, 2016	2(b)
Q	Payment made $550		
R	Payment made $525		
S	Payment made $1,075		
T	Cancel or modify	Contract in this form is canceled because of modification or termination by mutual agreement of the parties	8

Source: Authors' analysis

Table 4: Transition Function (δ)

Initial State	Event	Resulting State
start ☺	A	q0
q0	B	xT
q0 ☺	C	q1
q1	B	Dl
Dl	D	xL
Dl	E	xC
q1 ☺	F	P1
P1 ☺	G	P1d
P1d	B	Db0_1
P1	H	Dbrw_1
P1	I	Dbcv_1
P1	J	Dbbkr_1
Db0_1	K	Nb0_1
Dbcv_1	L	Nbnpd_1
Dbbkr_1	L	Nbnpd_1
Dbrw_1	L	Nbnpd_1
Nb0_1	Q	P2
Nbnpd_1	N	P1
Nb0_1	O	Acc1
Nbnpd_1	O	Acc1
Acc1	B	Crisis1
Acc1	S	xT
Crisis1	E	xC
Crisis1	D	xL
Crisis1	S	xT
P1d ☺	Q	P2

(Continued)

Initial State	Event	Resulting State
P2 ☺	P	P2d
P2d	B	Db0_2
P2	H	Dbrw_2
P2	I	Dbcv_2
P2	J	Dbbkr_2
P2d ☺	R	xT
Db0_2	K	Nb0_2
Dbcv_2	L	Nbnpd_2
Dbbkr_2	L	Nbnpd_2
Dbrw_2	L	Nbnpd_2
Nb0_2	R	xT
Nbnpd_2	N	P2
Nb0_2	O	Acc2
Nbnpd_2	O	Acc2
Acc2	B	Crisis2
Acc2	R	xT
Crisis2	E	xC
Crisis2	D	xL
Crisis2	R	xT

☺ *Transitions along the happy path of the contract lifecycle*

Only the transitions that result in a change of state are noted here. All un-noted transitions result in the state being unchanged.

Source: Authors' analysis

Table 5: Full Transition Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
start	q0	---	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
q0	---	xT	q1	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
q1	---	Dl	---	xL	---	P1	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
P1	---	---	---	xL	---	---	P1d	Dbrw_1	Dbcv_1	Dbbkr_1	---	---	---	---	---	---	---	---	---	xC
P1d	---	Db0_1	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	P2	---	---	xC
P2	---	---	---	xL	---	---	---	Dbrw_2	Dbcv_2	Dbbkr_2	---	---	---	---	---	P2d	---	---	---	xC
P2d	---	Db0_2	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	xT	---	xC
Dl	---	---	---	xL	xC	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
Acc1	---	Crisis1	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xT	xC
Acc2	---	Crisis2	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	xT	---	xC
Db0_1	---	---	---	xL	---	---	---	---	---	---	Nb0_1	---	---	---	---	---	---	---	---	xC
Dbcv_1	---	---	---	xL	---	---	---	---	---	---	---	Nbnpd_1	---	---	---	---	---	---	---	xC
Dbrw_1	---	---	---	xL	---	---	---	---	---	---	---	Nbnpd_1	---	---	---	---	---	---	---	xC
Dbbkr_1	---	---	---	xL	---	---	---	---	---	---	---	Nbnpd_1	---	---	---	---	---	---	---	xC
Nb0_1	---	---	---	xL	---	---	---	---	---	---	---	---	---	---	Acc1	---	P2	---	---	xC
Nbnpd_1	---	---	---	xL	---	---	---	---	---	---	---	---	---	P1	Acc1	---	---	---	---	xC
Db0_2	---	---	---	xL	---	---	---	---	---	---	Nb0_2	---	---	---	---	---	---	---	---	xC
Dbcv_2	---	---	---	xL	---	---	---	---	---	---	---	Nbnpd_2	---	---	---	---	---	---	---	xC
Dbrw_2	---	---	---	xL	---	---	---	---	---	---	---	Nbnpd_2	---	---	---	---	---	---	---	xC
Dbbkr_2	---	---	---	xL	---	---	---	---	---	---	---	Nbnpd_2	---	---	---	---	---	---	---	xC
Nb0_2	---	---	---	xL	---	---	---	---	---	---	---	---	---	---	Acc2	---	---	xT	---	xC
Nbnpd_2	---	---	---	xL	---	---	---	---	---	---	---	---	---	P2	Acc2	---	---	---	---	xC
xT	---	---	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
xL	---	---	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
xC	---	---	---	xL	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xC
Crisis1	---	---	---	xL	xC	---	---	---	---	---	---	---	---	---	---	---	---	---	xT	xC
Crisis2	---	---	---	xL	xC	---	---	---	---	---	---	---	---	---	---	---	---	xT	---	xC

jargon, each state exhibits the Myhill-Nerode property of being independent of its past.[5] We humans may attend to the narrative of events that brought the relationship to a particular state, but the DFA does not care. Once a state is reached, the history that led there is irrelevant; all that matters is the one-step-ahead process of responding to whichever event arrives next. Any relevant history is encapsulated in the fact of being in that state. As Rosenberg (2010, p. 56) puts it, "the state of a system comprises that fragment of its history that allows it to behave correctly in the future." This requirement of memorylessness implies a discipline that should (normatively) govern the drafters of financial agreements. It restricts the contract to consider only "regular" sequences of events, described below. It also has powerful implications for the computational complexity of the contractual machinery.

The transition function in **Table 4** simplifies the representation by suppressing the "stay-in-place" transitions that return the system to the same state in response to an event.[6] Unlike a parser, for example, which has a responsibility to reject character sequences that are unacceptable, a financial agreement has the flexibility simply to ignore most superfluous event occurrences. This creates a proliferation of self-transitions that would otherwise clutter Table 4 with trivial entries. In other words, the application context for financial agreements is less tightly controlled than for programming-language parsers, and contracts should be relatively permissive and robust to irrelevant event occurrences. Table 4 also suppresses omnipresent events such as the filing of a lawsuit (event D) that could be relevant in any state. For completeness, **Table 5** presents the full transition function as a matrix, where the rows correspond to the states in Table 2, the columns correspond to the event alphabet in Table 3, and the state listed in in each cell is the result of the transition function applied to the combination of the initial state (row) and event (column) values. Unlike Table 4, the full transition function in Table 5 does not suppress irrelevant and omnipresent events.

Regular Expression Representation

Our final representation of the deterministic finite automaton, or DFA, is a regular expression. A regular expression — sometimes called a "regex" — is a compact shorthand notation focusing on the event sequences that the DFA recognizes.[7] We emphasize that the regular expression presentation captures the

[5] The Myhill-Nerode property is essentially a non-stochastic equivalent of the first-order Markov property; in other words, a first-order Markov chain is the probabilistic counterpart of a deterministic finite automaton (DFA). In particular, the characteristic memorylessness of a Markov chain also applies here, even though the DFA is a non-random process. For a detailed discussion of the Myhill-Nerode property and its implications, see Rosenberg (2010), especially chapters 4-5.

[6] Von der Lieth Gardner (1987, chapter 6) refers to self-transitions as "ineffective events." Knowing when to ignore events, rather than reject (for example, by triggering a back-office investigation), is an important judgment call for contract drafters and implementers. For example, expiration of the statute of limitations with respect to the obligations of the borrower (event E in Table 3) before delivery of the principal (event F) is a physical impossibility, so there is little point in developing error-handling procedures for this case. On the other hand, multiple occurrences of event F in quick succession would be a plausible clerical error, and a rejection procedure might be appropriate to handle this case.

[7] For an introduction to regular expressions, see Friedl (2006). For a more technical introduction, see Aho and Ullman (1995, chapter 10). Hopcroft, et al. (2001), section 3.3, for an overview of the Unix syntax for regular expressions. Implementations of regular expression parsers frequently augment the functionality with features such as grouping and backtracking to make them more powerful than "pure" regular languages. We ignore these possible extensions here.

same structural information available in the tabular and graphical representations above (minus the textual labels that describe the states and transitions). Indeed, there are standard procedures for converting to the regular expression representation from the other representations, and vice versa.

More specifically, a finite automaton implies the set of all strings — concatenated sequences from the event alphabet — that the machine will accept. These are the event sequences for which the contract has scripted some appropriate behavior of the counterparties, and which leave the automaton in one of its "accept" states. For example, a given contract might accept a happy-path event sequence of "sign"-"pay"-"quit," meaning sign the deal, then make the promised payments, then terminate the relationship. In contrast, a sequence of "quit"-"pay"-"sign" would be nonsensical, and the contract's DFA should declare its inability to process this sequence of events. In the case of our streamlined contract, we can see that the event sequence "**ACFGQPR**" defines the happy path, while the shortest event sequence resulting in a final state is "**AB**."

The DFA defines a grammar for the language of all acceptable (by the automaton) event sequences recognized by the DFA. A DFA is one of the simplest computational formalisms and only supports some of the least expressive languages, known as the "regular" languages. A useful feature of the regular languages is that the shorthand notation of a regular expression can capture an entire regular language — and thus a DFA — with a single snippet of event-sequence patterns.[8]

We use the method of state elimination to convert the DFA represented in Figure 1 into an equivalent regular expression shorthand for the set of event sequences the DFA accepts. The key operation in state elimination is to replace the event arrows in the DFA with arrows describing event sequences, while still preserving the state-transition logic of the full graph. For example, consider this snippet from Figure 1:

{***Pmt. 1 due***} –[**B**]–> {***Default (borr.) $ miss***.} –[**K**]–> {***Borrower notified of payment default***}

One can eliminate the intermediate state, "Default (borr.) $ miss." without disrupting the overall state-transition logic by replacing the elided state with at joint transition arrow, labeled as an event sequence:

{***Pmt. 1 due***} –[**BK**]–> {***Borrower notified of payment default***}

This is a particularly simple example of state elimination, but the procedure extends in a straightforward way to more involved states in the network.[9]

The regular expression for a given DFA is not unique. We find it convenient to organize the regular expression for our streamlined contract as the union of four key segments, corresponding to: (a) rapid-demise paths; (b) the happy path; (c) unhappy paths (payment and nonpayment defaults) around the first

[8] For an introduction to the translation between deterministic finite automatons (DFAs) and regular expressions, see Sipser (2006), section 1.3. For a deeper discussion, see Rosenberg (2010), especially section 5.2.

[9] For a more detailed introduction to the state-elimination method, see Hopcroft, et al. (2001) section 3.2; or Sipser (2006), pp. 66-76. Hopcroft, et al. (2001) describe two general methods for the DFA-to-regular expressions conversion, namely path induction and state elimination. The two methods are equivalent in the sense of producing equivalent regular expressions.

payment date; and (d) unhappy paths around the second payment. Omitting the derivation of these four expressions, the overall regular expression representation of the DFA is:

A(B|CB[ED])|	*Rapid demise*
ACF(G(BK)?)QPR|	*Happy path*
ACF([HIJ]LN)*(GBK|[HIJ]L)O(S|B[DES])|	*Unhappy* 1
ACF(G(BK)?)Q([HIJ]LN)*(PBK|[HIJ]L)O(R|B[RED])	*Unhappy* 2

Note that the final segment, "unhappy 2," follows the happy path up to state **Q** (payment 2 accruing) and then diverges into the various ramifications of payment and nonpayment default from that state. The happy path segment here includes a wildcard sub-segment, **(BK)?**, covering the possibility of a missed payment that is quickly cured. Similarly, the two unhappy segments include a wildcard sub-segment, **([HIJ]LN)***, indicating that the contract can tolerate an arbitrary number (including zero), *****, of any of the three nonpayment defaults, **[HIJ]**, as long as they are cured in a timely fashion, **LN**.

The regular expression's string of symbols provides a simple and intuitive measure of the complexity of the contract, namely the length of the string.[10] The complexity score should measure the length of a standardized regular expression for the minimized DFA. In the example here, the complexity score is 109. This descriptional complexity measure is more precise than traditional measures of computational complexity. It is well known (see, e.g., Gasarch, 2014) that regular expressions have computational complexity $\boldsymbol{O}(1)$ — i.e., given a family (or language) of event sequences defined by the regular expression, there exists some finite upper bound on the computing time and memory required to decide whether an arbitrary event sequence fits the pattern. This fact is critical for questions of system scalability, and so it is perhaps unsurprising that the legal system has evolved standard practices at the level of individual agreements that guarantee the feasibility of enforcement infrastructure (courts, lawyers, arbitrators, etc.) in the aggregate. In contrast to standard measures of time and space scalability, $\boldsymbol{O}(T(n))$ and $\boldsymbol{O}(S(n))$ respectively, which assert an unspecified finite bound, descriptional complexity as defined here asserts a cardinal measure of complexity. One can state, for example, not merely that two contracts are both computable in finite time, but that one is more complex than the other in a specific sense.

A contract should be as simple as possible, but no simpler. Note that the bulk of the contract's complexity (75.2 percent, to be precise) arises in the two nexuses of unhappy ramifications. That is, the two unhappy substrings, while dealing with potentially unlikely events, account for 82 symbols in total, or 82/109 = 75.2 percent of the overall string length. Unsurprisingly, much of the hard work of managing a relationship occurs not when things are going well, but rather when the process starts to deviate from the happy path.

[10] One might object that, because a deterministic finite automaton's (DFA) state transition network and its regular expression are generally not unique, the complexity score is arbitrary. This objection is ill-founded, however, because there are programmatic techniques to reduce any DFA to a theoretical minimum state-transition network and standard techniques for representing any given DFA as a regular expression.

Much of the value of good contracts and good lawyering derives from the seemingly tedious planning for all the ways that a relationship might run off the rails.

Financial risk and valuation models — the core of financial engineering — tend to ignore these unhappy complications, focusing instead on probabilistic models of the happy path. Implicitly, this relies on an assumption that all unhappy relationships are idiosyncratic, so that the manager of a well-diversified investment portfolio can safely ignore these high-maintenance details. Holdings in many cases, such as bank commercial loan portfolios, consist of a relative handful of large, specialized relationships; this concentration of risk exposures denies the portfolio manager the luxury of simple diversification. Alternative mechanisms, such as securitization and syndication, have evolved to spread the risks in these situations. Formal modeling of these complicated portions of financial relationships as DFAs may make them more measurable and manageable. If so, such modeling has the potential to create significant value by facilitating the pervasive tasks of risk data integration at the very granular level of contractual events.[11]

VI. Conclusion and Directions for Further Research

Taken together, the three canonical representations — graphical, tabular, and regular — of the underlying deterministic finite automaton are equally valid embodiments of the process set out in natural language in the streamlined contract, with the added benefit of being expressly computable. The key is that the state transition structure is sufficiently fundamental to a financial agreement that we can represent it using the standard computational formalism of a DFA without disrupting the contract's organizing principles.

Why does this matter? By identifying the DFA that undergirds a contract, we expose the entire edifice to the tools and techniques developed in the computational and linguistics communities to work on finite automata. Representing the proposition in a computational formalism opens the contract to a number of tests, applications, and manipulations that are much more difficult to apply when the legal logic is expressed in natural language. For example, it is possible to craft the three representations so that they are precisely formally equivalent.

A key to establishing this mutual equivalence lies in the application of techniques for minimizing the finite automaton. One of the contributions of the Myhill-Nerode Theorem is that there exists a *unique* smallest finite automaton that will accept a given language of event sequences defined by the regular expression. We did not formally minimize the DFA for our streamlined contract in this way, preferring instead to maximize the common-sense legal semantics. It will be instructive to see how far our legally optimized representation is from the theoretical minimum. This gap is likely to widen as we apply the approach to more realistic agreements. It is important to recognize that legal contracts are ultimately devices for coordinating human activity and much of their effectiveness derives from their enforceability. The lender is willing to relinquish the principal, in part because she knows authorities exist to enforce repayment if necessary. These authorities involve human interpreters — judges, juries, arbitrators, etc. — who must be

[11] The Basel Committee on Banking Supervision (2013) identifies risk data aggregation as a key challenge for financial institutions.

able to parse the agreement in the crucial tasks of dispute resolution. Otherwise, the contract has failed to meet one of its most important requirements.

As the analysis moves to more complex and realistic examples, we will want to expand the toolkit beyond DFAs to include nondeterministic finite automata or NFAs. NFAs are a more sophisticated class of automata that allow multiple alternate transitions from a state in response to an event.[12] This is a representational convenience that affords significant simplification of the state transition graph in many cases. Note that NFAs are semantically equivalent to DFAs. They support the same set of regular grammars, and there exist standard techniques for translating between DFA and NFA representations. Any state-transition system with a DFA representation can be converted to an equivalent (in terms of its acceptable sequences of events) NFA and vice versa. It is already clear from our preliminary forays into standard financial master agreements that this sort of flexibility will be useful.

Other practicalities present a greater challenge to the use of a "pure" DFA to represent all aspects of a relationship. Real relationships typically have to keep track of small but important facts, such as mailing addresses and names of authorized signatories. As a simple example, the Section 12(b), "Change of Details," of the ISDA (2002) Master Agreement states: "Either party may by notice to the other change the address, telex or facsimile number or electronic messaging system or e-mail details at which notices or other communications are to be given to it." In other words, the contract calls for a fact that is included by reference, and which can be replaced without affecting the rest of the agreement. However, recording and referencing an ancillary fact technically adds a state variable to the computation, potentially violating the Myhill-Nerode property of the DFA. To handle this, the DFA formalism would need to be extended in some way, such as making the ancillary fact an external reference with carefully controlled dependency semantics.

The experience of augmented transition networks (ATNs), described by von der Lieth Gardner (1987), is instructive in this context. An ATN augments a simple DFA by adding a local memory "register," and allows transitions to refer to and condition on the contents of this register. However, without tight constraints on the register's contents and how they may be used, the expressiveness of the ATN is likely Turing complete.[13] This would vastly extend the expressiveness of the representation, and effectively negate the simplicity benefits provided by the Myhill-Nerode property. Further research is needed on appropriately expressive formalisms to handle these practical cases.

The DFA captures the central legal logic of the contract. However, the DFA by itself does not capture the entirety of the agreement. There are, in addition, two important semantic conversions that round out the picture. First, the agreement defines a measurement or feature extraction process to convert from the salient events and occurrences in the real world to the finite microcosm of the DFA. Has one of the events

[12] Multiple responses to the same event appear to be a contradiction in terms. However, the multiple transitions are not to be taken literally. Instead the nondeterministic finite automaton (NFA) is a modeling abstraction with identical expressiveness to the DFA, but which is typically more concise. See Sipser (2006, chapter 1) for further discussion.

[13] Turing-completeness is a statement about the expressiveness of a programming language. Roughly, a language is Turing-complete if any computable function can be written in that language. In other words, the language can describe any computation that might performed by the broadest class of computers, known as Turing machines. Most familiar programming languages, such as C++, Java, or Python are Turing-complete.

specified as salient in the computation actually occurred? For example, the borrower represents that his assets exceed his liabilities under generally accepted accounting principles. The DFA requires this simple determination — yes or no —encoding this fact as an alphabet element. The measurement process to reach this decision will typically involve a multitude of assumptions, interpretations, and judgments of accounting, but the bottom line will have the full clarity of a discrete, binary variable. Requiring the counterparties to maintain this clarity is a valuable function of the contract. In our current example, and contracting practice more generally, the translation of external occurrences into the event alphabet is still mostly handled by the natural-language definitions in the contract. Indeed, this translation pushes much of the discretionary ambiguity of contracting, sometimes pointed to as an asset in the process, away from the state-transition logic and into event definitions. The details and nuances of this measurement task are outside the scope of this paper, but they are an important area for follow-on research.[14]

Second, there is a question of the semantic interpretation of the states and transitions in the automaton. For example, when the automaton is in state "xL", this fact about the DFA has important implications for the parties back in the real world; one of the parties is likely to be filing litigation in a specified jurisdiction, requiring documentation of claims, etc. Some interpretative mechanism is required to understand that the simple marker, xL entails all these messy contextual details. This explicit mapping from the DFA to the external legal context is a sort of formal semantics that requires additional attention in subsequent research.[15]

The DFA is a system diagram setting out the logical structure of the contract. It is not intended to be a procedural flowchart for automating the relationship.[16] Indeed, the DFA would be a relatively cumbersome form for a computation engine, as each step must be set out in order, without the availability of memory or recursive operations to reduce the complexity of the representation. Actual implementation of more complex financial contracts will use these shortcuts. Nonetheless, the DFA is a good starting point for highlighting the conceptual link between contracting and computation.

An especially important extension of the state-transition model applies to financial agreements that interact through the events they consume. A finite-state transducer (FST) is an enhanced DFA that allows transitions to *emit* events. That is, contracts are not always mere consumers of events; contracts can generate events as well. Moreover, the output events for one agreement may be input events for another. A canonical example is a cross-default clause, which specifies that one contract should "listen" for

---

[14] The need for adequate specification of the external events and internal logic of a contract is reflected in the broadly recognized principle of Anglo-American law that courts will not grant enforcement to contracts that constitute "agreements to agree." In Delaware state courts, for instance, "a valid contract exists when (1) the parties intended that the contract would bind them, (2) the terms of the contract are sufficiently definite, and (3) the parties exchange legal consideration." *Osborn ex rel. Osborn v. Kemp*, 991 A.2d 1153, 1158 (Del. 2010).

[15] For an introduction to computational logics and formal semantics, see Huth and Ryan (2004).

[16] At the implementation level, many contracts will also involve some calculation chores, such as determining precise payment amounts, sorting through holiday calendars and day-count conventions, etc. The streamlined agreement elides these considerations by explicitly stating precise dates and dollar amounts. In general, these sorts of calculations would and should typically be the implementation details of some delegated subsystem.

transitions to default states as they occur in another agreement. Cross-default clauses can have systemic implications, because they typically trigger payments acceleration, creating a legal mechanism for the propagation of default across a network of contracts. A standard FST is a 6-tuple that augments the basic DFA by equipping it with an output alphabet, often represented as Γ. The transition function, δ, is similarly augmented, so that a transition is associated with a character from both the input alphabet, Σ, and the output alphabet, Γ.[17]

There are a number of potentially significant implications that flow from this exercise in computational contract specification. At the most basic level, the exercise of stating even a simplified contract as a DFA serves as a proof of concept, demonstrating that we can describe legal rule and consequence structures directly in computational terms. Of course, success here does not prove that this modeling technique would apply to all contracts, including those of much greater complexity. At this stage, however, we see no conceptual barrier to such a task, at least for agreements (which include the vast majority of financial contracts) involving fixed event spaces and transition rules.

Although embodying financial contracts in software has the potential to provide significant benefits, we also recognize that the increases in speed, accuracy and flexibility that this development will provide have the potential to create problems as well. In automotive engineering, for example, a significant increase of power in the motor is usually balanced by increased efficiency in the brakes. The use of one of the simplest computational formalisms (the DFA) in our streamlined example is intentional in this regard. There is a danger of the sorcerer's apprentice problem: that the unwise application of powerful computational tools could encourage inexperienced drafters with only limited understanding of the issues involved to create contracts of unmanageable complexity.[18]

Our analysis of the streamlined loan agreement initiates a larger project of automating financial instruments. An obvious next step will be to undertake such a description for existing agreements actually used in financial markets, such as the ISDA (2002) Master Agreement and the 1997 International Foreign Exchange Master Agreement. While the proof necessarily awaits the exercise, a preliminary review of these agreements suggests that the challenges of restating them as DFA are those of time and patience in the face of complexity, rather than of fundamental differences of kind. Such a step would be a precursor to creating a software version of these instruments; it could also suggest re-drafting opportunities. We can imagine an updated ISDA Master Agreement that is completely computable. In addition to restating existing natural language contracts as DFA, another plausible next step would be a project of new drafting, where the goal is to embody transactional structures straight into the formalism of computation and software, without relying on a natural language precursor.

---

[17] Transducers are widely used in control systems, including computer hardware design, and in computational linguistics. The control applications are closer to our case of computable contracts; see Mueller and Paul (2000). There are two general classes of finite-state transducers — Moore machines and Mealy machines —that differ essentially in whether outputs can (Mealy) or cannot (Moore) depend on the input event that triggered the transition. See Moore (1956), and Mealy (1955).

[18] Computer systems designers worry actively about the problem of state space explosion (i.e., proliferation). See for example Baier and Katoen (2008), especially chapter 2.

References

Aho, A. V. and Ullman, J. D. (1995), *Foundations of Computer Science*, Computer Science Press. http://infolab.stanford.edu/~ullman/focs.html.

Allen, L. (1957), "Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents," *Yale Law Journal.* 66, 833–79.

Baier, C. and Katoen, J. (2008), *Principles of Model Checking*, MIT Press.

Basel Committee on Banking Supervision (2013), "Principles for Effective Risk Data Aggregation and Risk Reporting," January, http://www.bis.org/publ/bcbs239.htm.

Bolton, P. and Dewatripont, M. (2005), *Contract Theory*, MIT Press.

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

Brose, M. S.; Flood, M. D.; Krishna, D. and Nichols, B. (2014a), *Handbook of Financial Data and Risk Information Vol. I: Principles and Context*, Cambridge University Press. http://www.cambridge.org/sg/academic/subjects/mathematics/mathematical-finance/handbook-financial-data-and-risk-information-i-principles-and-context-volume-1.

_____ (2014b), *Handbook of Financial Data and Risk Information Vol. II: Software and Data*, Cambridge University Press. http://www.cambridge.org/sg/academic/subjects/mathematics/mathematical-finance/handbook-financial-data-and-risk-information-ii-software-and-data-volume-2.

Clark, C. (2011), "Whither Equity Volumes?" January 31, 2011. *NYSE Exchanges*. http://exchanges.nyx.com/cclark/whither-equity-volumes.

Flood, M.; Mendelowitz, A. and Nichols, W. (2013), "Monitoring Financial Stability in a Complex World," in Lemieux, V. (ed.), *Financial Analysis and Risk Management: Data Governance, Analytics and Life Cycle Management*, Springer Verlag, 15–46. http://www.springer.com/cda/content/document/cda_downloaddocument/9783642322310-c2.pdf?SGWID=0-0-45-1356754-p174549637.

Foreign Exchange Committee (1997), "The 1997 International Foreign Exchange Master Agreement (IFEMA)," Technical report, Federal Reserve Bank of New York, accessed December 12, 2014. http://www.newyorkfed.org/fmlg/ifema.pdf.

Friedl, J. E. F. (2006), *Mastering Regular Expressions. 3rd ed.*, O'Reilly.

Gasarch, W. (2014), "Classifying Problems into Complexity Classes," *Advances in Computers*, 95, 239–292.

Genesereth, M. R., Greiner, R. and Smith, D. E. (1981), *MRS Manual*, Stanford University. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA123256

Greenwood, R. and Scharfstein, D. (2013), "The Growth of Finance," *Journal of Economic Perspectives* 27(2), 3–28.

Grosof, B. N. and Poon, T. C. (2004), "SweetDeal: Representing Agent Contracts with Exceptions Using Semantic Web Rules, Ontologies, and Process Descriptions," *International Journal of Electronic Commerce*, 8(4), 61–97.

Hopcroft, J., Motwani, R., and Ullman, J. (2001), *Introduction to Automata Theory, Languages, and Computation. 2nd ed.*, Pearson/Addison Wesley.

Huth, M. and Ryan, M. (2004), *Logic in Computer Science: Modelling and Reasoning about Systems. 2nd ed.*, Cambridge University Press.

ISDA (2002), "ISDA Master Agreement," ISDA Bookstore, accessed December 12, 2014. http://www.isda.org/publications/isdamasteragrmnt.aspx.

Kozen, D. C. (1997), *Automata and Computability*, Springer.

MacLeod, W. B. (2007), "Reputations, Relationships, and Contract Enforcement," *Journal of Economic Literature* 45(3), 595–628.

Mealy, G. H. (1955), "A Method for Synthesizing Sequential Circuits," *Bell System Technical Journal*, 1045–1079.

Moore, E. F. (1956), "Gedanken-experiments on Sequential Machines," *Automata Studies*, 34, 129–153. http://books.google.com/books?hl=en&lr=&id=oL57iECEeEwC&oi=fnd&pg=PA129#v=onepage&q&f=false.

Mueller, S. M. and Paul, W. J. (2000), *Computer Architecture Complexity and Correctness*, Springer.

Object Management Group (OMG) (2014), "Enterprise Data Management Council (EDMC) Financial Industry Business Ontology (FIBO) Standard, Version 1.0 - Beta 1," Technical report, OMG, Accessed December 13, 2014. http://www.omg.org/spec/EDMC-FIBO/FND/.

Project ACTUS (2015), "ACTUS: Algorithmic Contract Types Unified Standards," Internet site accessed August 28, 2015. http://www.projectactus.org/.

Rosenberg, A. L. (2010), *The Pillars of Computation Theory State, Encoding, Nondeterminism*, Springer.

Simonson, S. (2013), "Theory of Computation," *ADUni.org*. http://www.aduni.org/courses/theory/.

Sipser, M. (2006), *Introduction to the Theory of Computation. 2nd ed.*, Thompson Learning.

Software Engineering Institute (2009), "'Happy Path' Testing," *Acquisition Archetypes*, Carnegie Mellon University. http://www.sei.cmu.edu/library/assets/happy.pdf.

Stout, L. A. (2012), *The Shareholder Value Myth: How Putting Shareholders First Harms Investors, Corporations, and the Public*, Berrett-Koehler.

Surden, H. (2012), "Computable Contracts," *University of California - Davis Law Review*, 46, 629–39.

von der Lieth Gardner, A. (1987), *An Artificial Intelligence Approach to Legal Reasoning*, MIT Press.